Ecosystems, Complexity, Topology and Evolutionary Computation

Clinton Jon Woodward

A thesis presented for the degree of
Doctor of Philosophy

2010

Abstract

Evolutionary algorithms have been applied to an increasing range of complex problem domains. A challenge for many applications is the discovery of appropriate structures and processes that allow solutions, and solution components, to emerge efficiently.

The motivation of this thesis was to create a new ecosystem model of evolutionary computation (ESEC) and to investigate the influence that topology and interaction can have on the outcome of evolutionary search. The thesis begins by considering the field of ecology and models of ecosystems, with a particular emphasis on evolutionary models, structures and processes. Next, existing models of evolutionary computation are considered with a strong emphasis on aspects of topology. Modern developments in the field of graph theory provide new insight into complex systems and the properties of efficient structures.

A range of investigation themes have been developed for the ESEC model, and a detailed survey of topology models and properties was undertaken to guide the selection of suitable structures. An empirical study considers in detail the specific influence of various population structures on evolutionary search outcomes, and shows that the specification of population topology can influence both the efficacy and efficiency of evolutionary search. The results are a motivation for future investigations to consider in more detail how and why such influence can be used to an advantage as a way of optimising evolutionary search applications.

Acknowledgements

Like complex networks and processes of adaptation, that have consumed my attention for so long, this thesis and body of work is the result of many unique influences. I sincerely appreciate the many people that have helped me to complete this thesis, but I will only acknowledge a few.

Prof. Tim Hendtlass has been a wonderful mentor and an invaluable supervisor. Without his enthusiasm, guidance and sense of humour, many things would be different in my life. I know this is true for all of us that have passed through his care. Tim inspired me as an undergraduate, believed in my potential, and assisted me in developing an academic career – even though he knew it would likely be to the detriment of my PhD completion! You were right in many ways.

The support of Dr. Howard Copland as my co-supervisor is greatly appreciated. I have always enjoyed our conversations during this process, although the number of new ideas such sessions would create was dangerous. Your understanding, rigour and attention to detail were exactly what I needed.

To all my colleagues over the years (old and new) within both the Centre for Complex Systems and later the Centre for Intelligent Systems and Complex Processes, thank you for the discussion (usually meaningful) and support (always appreciated).

In particular I wish to thank my good friends Matthew Dafilis (yes, $\delta T/\delta t > 0$) and Gerard Murray for their enduring support and understanding. Gerard – my family has adopted you.

To the many students I have had the honour of teaching and supervising, I thank you for your encouragement and feedback, and for allowing me the occasional diversionary "rant" about my research interests. A wise teacher once quoted to me, "I learn as I teach, and I teach as I learn". I have certainly experienced this! Thank you to the students who let me learn with them.

In complex networks, there are critical nodes; without them, a system fails. My family are the most important of all. To my heavenly Father, thank you for saving me. To my parents, Marilyn and Paul Woodward, thank you for always supporting and encouraging me, and setting me on this path (even though you probably didn't quite imagine *this* particular path). To my wife Zanyta, I know that words really can't explain how much your support has meant to me – I'll do my best to show you as time goes by. And lastly, to my children Kaelan and Talia, who have missed their Dad; it is definitely time for a scruff!

Declaration

I hereby declare that this submission is my own work and to the best of my knowledge it contains no material previously published or written by another person except where due acknowledgement is made in the thesis. Any contribution made to the research by colleagues, with whom I have worked with at Swinburne University of Technology or elsewhere, during my candidature, is fully acknowledged.

I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged.

Clinton Jon Woodward

Table of Contents

List of Figures

List of Tables

Part I

Ecosystems, Topology and Evolutionary Computation

Chapter 1

Introduction

1.1 Overview

When observing natural systems we are often presented with examples of elegant solutions to problems. It seems that nature regularly uses strategies that successfully exploit the characteristics of robustness, adaptability, self-organisation and efficiency. Attempts to understand and replicate such desirable qualities have been seen in many fields, from science and engineering to business and design.

Work discussed in this thesis has been inspired by the processes of evolution and adaptation, and the efficient properties of topologies as observed in nature.

The motivations were to create an ecosystem model of evolution to represent and capture the ideas of complex and efficient topological structures, to relate this to current evolutionary computation paradigms, and to use the proposed model to question and investigate both the properties and potential of the model.

1.2 Evolution and Genetics

Concepts of adaptation, genetics, inheritance and the term "survival of the fittest" are familiar to a large part of the general community. With modern computational techniques, the application of *artificial* evolution can take advantage of nature's strategies and apply them to new problems far removed from the context in which they have been observed.

In Charles Lyell's "Principles of Geology" [220], it was argued that our world is in a 'steady state' of change due to geological forces which constantly reshape our environment. This was one of the influences of Charles Darwin (1809-1882) who deduced, concurrently with Alfred Wallace (1823-1913), that species could adapt and that individuals that are better suited to their environments have a better chance of survival.

It is this principle of "survival of the fittest" that is the underlying mechanism of natural evolution. Publication of the book popularly known as "The Origin of Species" in 1859 by Darwin [62] cemented his reputation as the first author on the theory of evolution, even though he was not, as is popularly thought, the single contributor of the theory itself [63]. Jean-Baptiste Lamarck (1744-1829) is recognised as the first to formalise shared thoughts about the process of organic evolution in biology in the early 1800s.

During the mid-1800s the Austrian monk Gregor Johann Mendel (1822-1884) pioneered work in the science of genetics. His research, involving the cross-breeding of various plants, showed that biological inheritance was the result of the transfer of physical elements, and that these elements were directly related to the physical characteristics of the plants. Mendel believed his results were significant and was disappointed that no one seemed to realise their importance when he presented and published his work. However in 1900, 16 years after Mendel's death, three other scientists independently repeated Mendel's experiments.[1] It was only then that the larger scientific community began to realise that Mendel had already discovered the significant elemental nature of inheritance which we now know as *genetics*.

The field of modern genetics only truly formed in the 1920s once the tools of maths and statistics had developed to deal with the amounts of data often associated with genetic structures and its many variations. New insight into epigenetic factors[2], development influences and the dynamic networks of gene expression, all continue to expand the field of modern genetic knowledge.

It is important to note the distinction between the process of evolution, and the specific mechanisms of genetics. They are both important to the work presented in this thesis.

1.3 Ecology and Ecosystems

Ecology is a multi-disciplined and integrative area of scientific study. Its subject matter is the entire world, both living and non-living parts. As a field, ecology encourages a method of observation and assessment that integrates how all parts of a system fit together. The objective of ecology is to understand how each part influences, and is influenced by, other parts.

Ecosystems are defined as a unit that include a community of organisms, an environment, a network of interactions and processes, and energy. Understanding an ecosystem is an understanding of the processes that govern the transformation of material and energy contained in the system. An ecosystem model includes networks of interactions at many different levels. Scale is simply an arbitrary matter of relevance to the question being asked.

From ecology and ecosystems we can gain insight into natural systems that are not simply collections of isolated components, whose behaviour is not a simple summation of properties, but rather examples of emergent complexity.

It is this objective of integrated system level understanding that has motivated a large portion of the research in this thesis, which attempts to combine ecosystem ideas with the tools of evolutionary computation.

[1] See Roger Blumberg's resource website titled "Mendel's Web" at http://www.mendelweb.org for English translations and commentaries of Mendel's work.

[2] A factor that changes an organisms expressed phenotype (physical form), but not its genotype (genetic form). See the glossary in Appendix A.1 for related terminology.

1.4 Evolutionary Computation

It is the combination of concepts from Darwinian evolution and the Mendelian idea of elemental genetic transfer that allow artificial models of evolutionary computation to be applied and utilised. Evolutionary computation provides useful methods of search and optimisation for problem domains that are difficult for many other techniques. There are various implementations of evolutionary computation, many of which have been developed and adapted for specific needs.

The term "Evolutionary Computation" (EC) was used in 1993 for the first journal formed to encompass the separate computational research fields making use of biologically inspired evolutionary processes [113]. EC can be inclusive of concepts from both evolutionary biology and computer science, although differences in nomenclature can be deceptive. A common perspective, which is adopted in this work, is that the term Evolutionary Algorithm (EA) applies to instances of EC. The terms EC and EA have been used interchangeably within published work, and predominant researchers continue to promote the EC terminology.

The essential aspects of an EA can be described as a population of potential solutions (individuals) that are competitively modified and replaced by evolutionary operators. A fitness function determines each individual's performance and this is used to influence the reproduction and replacement of individuals in the population. Note that the directed aspects of EA search are the *selection* of individuals for reproduction and the *replacement* of individuals in the population.

There have been three major historical areas of early EA activity within the EC field: Genetic Algorithms (GA), Evolutionary Strategies (ES) and Evolutionary Programming (EP). Genetic Programming (GP) is a more recent specialisation of GAs, and there are now other new additions such as Differential Evolution (DE) and Cultural Algorithms (CA).

It is important to keep in mind that while the term EA and others like GA and ES are similar, an EA has quite a distinct meaning within the EC field. The term EA is an algorithm category or classification; it embodies a number of different EA instances including – but not limited to – those already listed. In this thesis algorithms using coevolution are also included within the EA classification.

With the sharing of ideas most, if not all, of the traditional distinctions are now historical rather than practical. The different EA areas are discussed in Chapter 3, along with a framework for considering EAs within an ecosystem model.

1.5 Performance and No Free Lunch

There is an old adage that "there's no such thing as a free lunch". When applied to problem solving algorithms, this implies that good algorithmic performance must come at a cost. This idea has been formalised by the work of Wolpert and Macready as the "No Free Lunch" (NFL) theorem [370].

A simple description of the NFL theorem is that *no single method is best for all*

problems. The "no free lunch" idea has popular appeal and many feel it is intuitive, yet the application of its principles is at times ignored by researchers in the elusive hope of creating high performance and generally applicable algorithms.

Search algorithms that display excellent performance for specific problem domains cannot be expected to perform equally well for all domains. Usually, there needs to be compromise between the qualities of algorithm *generality* and *specificity*. If these qualities are mutually exclusive for a problem domain, an algorithm cannot attain both.

A better description of the NFL theorem uses a comparison of two algorithms over an entire theoretical domain of problems: *For any pair of search algorithms there are as many problems that the first algorithm out performs the second algorithm as there are problems for which the reverse is also true*. An interesting consequence of this is that when we compare, for example, an evolutionary algorithm (EA) to a random search algorithm, there are as many problems for which our EA is likely to be better as it is likely to be worse. This direct statement – that an EA search will be worse than a random search for some problems – has caused debate, especially given that there has been a common view that EAs would always be better than a random search. This should not be expected, especially in light of the NFL theorem ideas.

For problem solving algorithms, there are ways to incorporate hard-earned knowledge about the nature of the search space, or even exact solutions for specific domains (and with good reason to do so). It is possible to give an algorithm a "free lunch" to good performance. Such "free lunch" techniques include data pre-processing, specialised search initialisation and performance normalisation techniques.

Unfortunately, without careful consideration, the exact contribution of "free lunch" knowledge to an algorithm can be somewhat obscured by researchers as part of "specialisations" or hidden under the banner "generic improvement" (which is rarely the case). For this reason, when comparing and developing algorithms, we should take care to distinguish between performance gained from specialising *search* ability and the incorporation of existing "*free lunch*" knowledge.

1.6 Algorithm Performance

The main measures of algorithm performance are *solution quality*, *algorithm efficiency* and *algorithm repeatability*. The importance of these measures varies for different problem domains and specific application objectives.

Interestingly, when the performance of one of these measures can be lowered, another can usually be improved. For example, consider a scheduling problem for a school where students, classes and teachers must be allocated. A solution of the highest possible quality can be found if we can take all the time needed to do an exhaustive search. Similarly, if we need a solution quickly and we are prepared to compromise a guarantee of highest quality solution, an incomplete heuristic search technique can be used to quickly find what is needed.

It is also apparent in this simple scheduling example that there may be multiple solution *qualities* to consider, such as the notion of "best" solution for students, teachers or

room utilisation. Multiple algorithm performance qualities may indeed be independent or dependent (competitive or cooperative), adding to the complexity of the domain and any performance measures.

1.7 Problems and Problem Solving

There are many kinds of problems and the appropriate methods for solving or searching are varied. However, it helps to try and define clearly what *type* of problem we are trying to solve, and hence to choose appropriate methods.

Optimisation problems are a type of search, in that the objective is to find "a solution" that fits a given set of constraints; one of these is that a desired quantity be optimised. This might be the minimisation or maximisation of one or several outputs of a given system.

Engelbrecht [102] has suggested some useful "basic ingredients" for all optimisation problems and they are summarised below:

- An **objective function**. This is the quantity (or quantities) for which we wish to find the best solution.
- A **set of variables**. These variables affect the value of the objective function.
- A **set of constraints**. Limits or bounds to the values that variables may have.

The goal, then, of problem solving techniques is to assign values to the variables such that we achieve an objective function output that is satisfactory and a solution that is within the set of constraints.

For some problems there is no objective function except that all constraints must be satisfied, and hence are known as constraint-satisfaction problems (CSP). It is possible that not all variables affect the objective function – indeed this may be part of the problem: to find out which variables are significant.

Understanding an entire problem is the result of considering all three "basic ingredients". Complex rules are possible within constraints, and this not only limits value ranges but also valid combination of values.

A problem solving technique may need to allow a search to proceed through regions of variable space that are not within the constraints. In this form, there are two sets of constraints: one for the *search space* and another for *"viable"* solutions. This kind of distinction is important for EC algorithms, where there may well be an advantage or a need for the evolutionary process to allow non-viable solution attempts as intermediate evolutionary steps to the final viable solution.

Engelbrecht continues in [102] by classifying problems using seven characteristics. The characteristics are reiterated below:

- **Number of variables**. The more variables there are, the greater the likelihood of complex or combinatorial difficulties, and the more resources needed to solve or search the problem space.

- **Type of variables**. Real continuous values (where say $x \in \mathbb{R}$), integer or discrete (as $x \in \mathbb{Z}$), or a mixture of both. Combinatorial problems are constrained to permutations of integer-valued solutions.
- **Degree of non-linearity of the objective function**. For example, the objective function could be linear, quadratic or any other type of non-linear form. This includes the possibility of discontinuous functions.
- **Constraints**. As already mentioned, variables may be constrained with simple boundary conditions, however there may also be other intermediate value constraints or complex regions of constraint.
- **Number of optima**. If there is only one "optimal" solution, the problem is unimodal. Multimodal problems contain more than one optima.
- **Number of optimisation criteria**. Multi-objective problems need to satisfy more than one internal sub-objective function. Uni-objective problems are simply a single function.

So, given these classifying characteristics of problems, how then can we go about the business of problem solving, be that simply search or specifically an optimisation goal? Consider the following approaches:

- **Know *the* answer** (or "simple and direct"). If you already know the answer to a problem, the search for the solution is over. If the problem has multiple answers, know all of them. If there is a range of solutions, know the range. The point being that there is no need to search, and a problem solving algorithm is simply the use of existing known solution(s). This is not a complex or a naïve search algorithm; it is a way to solve problems at lowest cost and best performance. This is the most direct way to find a solution and end the search, and defines a useful benchmark.
- **Guess** (or "random search"). If we are unable to use knowledge of the problem domain to help our method, a random approach may still be much better than an exhaustive one, but this depends on the exact nature of the performance measure. Of course it is possible to "target" a random search, or distribute our random guesses based on hints or clues, but then it would no longer be a random search but a guided approach. A truly random search is an unguided search. It has uses, but a guided or "directed" (informed) search is expected to be better *if it is possible.*
- **Know *every* answer** (or "exhaustive search"). Once an exhaustive search is finished, we can use the "know the answer" method and select the best solution available. For small problems an exhaustive search is a very effective and justifiable method. However, for large problems with multiple variables a "combinatorial explosion" can soon make exhaustive search impractical, even with modern advances in computing power. This is also commonly known as the "curse of dimensionality". So, it is better if we can search efficiently and not waste time searching for solutions

where they do not exist or are not up to the standard we need. An exhaustive search is not directed in any way, but simply a sequential process. Once knowledge is exhaustively discovered, stored and organised, the search for an answer is a single direct step.

- **Know *how to find* the answer** (or "guided search"). In this case we don't know the answer, but we know how to find it. This is a guided search approach. It is akin to searching for water in low areas of terrain because of the likelihood of creeks and rivers. This method also implies that we already know something about the problem domain, or at least its nature. Again, this help in our search is a "free lunch" before we start. After that, we are on our own. If our knowledge of the domain is poor, we may also miss good solutions entirely. A fresh spring of water may exist in locations far from low areas of terrain, an anomaly to "typical" features. So there are two components to using "how to" knowledge: the *nature of the problem* and the *method* to make use of what we know.

Optimisation methods can be grouped into classes that relate to their characteristics [102]. Problem solving methods are either *constrained* or *unconstrained* within the search space. *Multi-objective* problems can be searched differently to *single-objective* function problems to take advantage of linkage between objectives. *Niching* techniques, which are designed to exploit "niche" areas of problem space, can be effective in exploring multimodal spaces.[3] If an objective function depends on a variable of time, there are techniques specifically for *dynamic* problems.

The exploitation of "how to" knowledge can be a complex and difficult task, and with the likely real-world scenario of unknown or erroneous knowledge of a problem, practical algorithms need to be robust in their "how to" approaches in order to ensure they are not trapped by assumptions or deceptive domains.

1.8 Parameter Adaptation and Control

It appears intrinsic, and is perhaps fundamental, that for evolutionary techniques to efficiently evolve solutions of quality, an appropriate technique must be selected, and parameters must be uniquely adapted to specific problem domains.

Well known problems are best solved by a known solution or specialised, problem-specific algorithms. However, real problems are commonly not "well known". Realistically then, excellent algorithm performance for difficult, unknown or changing problem domains needs to be achieved through adaptive strategies.

There are a variety of terms that are used related to the concept of selecting algorithms for problems, or methods suitable for adapting algorithm parameters. These include *meta-heuristics*, *meta-algorithms*, *meta-optimisation*, *dynamic-algorithms*, *adaptive heuristics* and *adaptive algorithms*.[4] An attempt to arbitrarily define each term would be difficult and of little universal value.

[3] The topic of niches and niching methods is discussed in more detail in both Chapter 2 and Chapter 3.
[4] Hyphenation of the meta prefix is typically inconsistent.

The common "meta" prefix comes from the Greek meaning of "beyond" or "higher level". A "heuristic", meaning "to find", is often applied to methods in computer science that are used to find solutions to problems without the formal need to show proof of quality. To put it another way a *heuristic* is simply a "pretty good rule", while a *meta-heuristic* is "a pretty good rule for finding pretty good rules".

We can make a distinction between two main types of adaptive algorithms: those that simply adapt algorithm parameters and those that attempt to select from, or combine together, a number of internal sub-components (ie. algorithms). Within the evolutionary computation field there have been attempts at both types of algorithm adaptation.

There are many possible parameters and configuration strategies that can be altered to affect EA performance, and it is not surprising that there has been significant work into self-adapting EAs. An example is the work by Kizu et al. [203, 303, 202] labelled as a "parameter free genetic algorithm". "Adaptive parameters" is a better description of the processes employed, which in turn "free" practitioners of the need to specify some parameters. However, it does not remove the importance of parameters from an algorithms' performance, nor can it guarantee the adoption of suitable parameter values.

The survey paper of [97] is a well cited example in the area of parameter adaptation and control, which presented a taxonomy inclusive of *parameter tuning* (before search) and *parameter control* (during search) ideas, and clearly defined related terminology in this area of EAs. More recently there have been collections such as [219], and related work such as [252, 321] aimed at measuring the costs and benefits of parameter control approaches.

A "meta-heuristic" approach is not a magic cure for solving all problems. It is quite likely, as stressed by the "No Free Lunch" theorem, to find as many functions for which such a "meta"-based method performs poorly as for which it does well. Indeed, the additional complexity of a meta-heuristic approach will be an additional cost to any search. For this reason, although the techniques explored in this thesis are part of the field of meta-algorithms, it is acknowledged that such techniques need to be applied appropriately, and without the expectation of universal advantage.

1.9 Evolution and Topology

This thesis examines components that exist within an Ecosystem EC framework, and there are indeed several levels of structural complexity that can be identified, as presented in Chapter 3. The organisation and interaction of these structural properties has a strong influence on both *solution quality* results and the *efficiency* of the evolution process. This idea is strongly supported by the large body of work published in the EC and complex systems research fields that relates to the modification of structural aspects of evolutionary algorithms in order to improve their performance. Further discussion of existing work in these areas is covered in Chapters 2, 3 and 5.

In particular, *populations* of solutions and solutions composed of *cooperating components* are ideal topological elements (Figure 1.1). It becomes possible then, as part of the

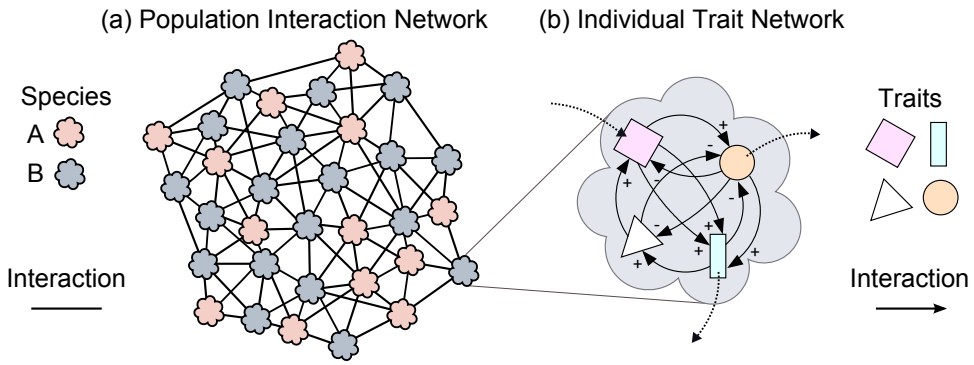


Figure 1.1: The ideas of (a) population topology where individuals of two different species are linked to neighbours, and (b) an expanded view of a single individual's topology. Examples of two possible networks within EC that can change, be changed and influence evolution.

adaptation of evolutionary computation to specific problem domains, to investigate and ideally control topological structures in ways that can improve algorithm performance.

1.10 Graphs, Complex Systems and Efficiency

1.10.1 Graph Concepts

The body of this thesis depends on the understanding of systems and connected structures in the mathematical terms of *graphs*. In this sense, a *graph* is a *set* of items often called *nodes* or *vertices*. Vertices are connected with *edges* (or links). Depending on the model, both nodes and edges can be simple or complex, such that edges can have direction, or values associated with them. Nodes can have different types or meaning within a graph model. There is a difference between the meaning of *network* and graph within some literature[5], however for the purpose of this work it is appropriate to use either term.

We can use graphs to represent and model many different systems, such as social networks, business associations, metabolic networks, the spread of disease, protein networks, neural networks, road traffic systems, the Internet, electricity grids, and hyperlinked web pages, to name just a few [26, 259]. We consider in some detail the possible distinctions of real-world networks in Chapter 4, along with models and techniques to help understand the properties real networks contain.

One model that deserves mention is the small-world model. The small-world phenomena has been noted and commented on since the 1960s [232], however it is the modelling of such small-world networks [360] that has prompted much of the recent work into complex systems and networks.

Interestingly, the research into networks and graphs is not limited to mathematicians. Because of the ubiquitous nature of networks and complexity in the world around us, complex systems and networks are everywhere and their study has applications to almost any discipline. It is hoped that the work in this thesis will also have broader applications.

[5]Technically, a network is a weighted directed graph, while a graph is the abstracted topology of nodes and connections only.

1.10.2 Complex Systems

What exactly is a complex system? This can be a somewhat vague or fuzzy quality to define. A complex system is not just a "collection" of many things, but rather the interactions between components that make the collection something more than just the sum of all the components [257]. As Craig Reynolds, well known for his "boids" modelling of animal group behaviours, has put it "A flock is not a big bird" [352].

We know from observations that real-world networks, both natural and man-made, tend to be *effective* (do the job required), and *efficient* (minimal cost and maximal functionality). Complex systems cannot be defined just by the nature of interactions, but also require the topology of connections.

Mark Newman has pointed out that although historically research and study of networks has focused on individual components (especially aspects such as non-linear interactions), research is now firmly focused on the study of connectivity properties [257]. In fact, the work of Barabásis, and other peers, has shown that behind every complex system there seems to be an underlying network with non-random topology [25]. So understanding complex systems is about understanding such non-random networks.

We can now share, measure, model, search, simulate and analyse networks in ways that were difficult or simply not possible before the advent of the computers and the internet. This also makes it possible to apply the lessons learnt from graph theory and complex systems research to new (and old) domains such as evolutionary computation.

1.10.3 Efficient Topology

The existence of *efficient topologies* has been observed in many natural and adapted systems. It is important to realise that the measure of "efficiency" is different in different domains. Efficient topologies are graphs of nodes and edges connected in a manner that minimises the cost of edges whilst maintaining short characteristic path lengths or other functional properties that are the measure of performance for the network. Examples of efficient topologies include social networks, telecommunication networks, citation networks, web pages with hypertext links, river systems, cardiovascular systems and the connections of biological neurones.

Chapter 4 takes a more detailed look at graph theory and current research into the measurement, properties and requirements of efficient topology structures, with a particular emphasis on aspects that can be utilised with evolutionary computation.

1.11 Research Objectives

The primary goal of this thesis is to present a new *ecosystem model of evolutionary computation* and use it to investigate the influences of topology and interactions on the process and outcome of evolutionary algorithms. The proposed model explicitly includes *topology* and *interactions* at the levels of *environment*, *community*, *species*, *population*, *individual* and *trait*.

In support of the primary goal are the following research objectives:

- develop an ecosystem model of evolutionary computation and validate its use in context by classifying related models in the field of EC;
- design and implement an ecosystem model of evolutionary computation that explicitly supports the structural levels and interactions required;
- to study the influence of topology and interaction at community and population levels using complex and dynamic topology, multi-layered communities, occupancy models and directional flow models.

1.12 Contributions

The main contributions of this thesis are:

- The development of an ecosystem model of evolutionary computation (ESEC) which is inclusive and supportive of prior work and flexible for future ideas.
- Software realisation of the proposed model and its ideas written in Python (the `esec` package) to provide a flexible platform for algorithmic experimentation.
- Using the proposed model it is possible to investigate complex organisational models at population, community and ecosystem levels.
- The creation of a set of investigation themes using the proposed model, and a body of empirical work to address the questions raised.
- An understanding that topology provides an alternative to traditional EA techniques of applying and adjusting selection pressure.
- Evidence that topology with specific properties provides a means of preserving population diversity, and insight regarding the influence that topology has to support niches and self-determined speciation.

Results show that topology properties of a population influence evolutionary process. Subsequently topology can influence the efficacy of success, the quality of solutions found and the efficiency (such as the search time required) of the search.

Understanding the relationships of topology and EC, and in particular within an ecosystem based framework, provides a useful means of configuring EAs in ways that may improve the likelihood of high quality solutions and/or the efficiency of search for particular domains. Using this knowledge researchers and practitioners would ideally be better equipped to select appropriate EC techniques and parameters when dealing with real-world problem domains.

1.13 Thesis Structure

This thesis is divided into two main parts. Overall, Part I is a review of the concepts and principles of ecology, topology and evolutionary computation, and proposes an novel ecosystem model for evolutionary computation which is used in Part II.

In Chapter 2 the principles of ecology, ecosystems, and ecosystem evolution (Section 2.4) are presented. A model of organisational scale is used to help relate both ecology and ecosystems to the structures of evolution. A community model defines and describe interactions within and between species and the environment, including co-adaptation and symbiosis. The mechanisms of evolution are described, including selection and mutation, and theories of speciation and evolutionary limits. Appendix A.1 is a glossary of relevant terms.

Chapter 3 presents the principles and applications of evolutionary computation as a biologically inspired search metaphor, with respect to the ecosystem (and community) model of evolution of Chapter 2, and with an emphasis on interactions and topology that are covered in more detail later in Chapter 4. Several canonical and relevant instances of EC algorithms are described for use in later chapters. A novel ecosystem model of evolutionary computation is proposed and described in detail in Section 3.2, which emphasises topology and interactions. A brief review and classification of classic EC work is included using the proposed model to demonstrate its value in context of the field.

Chapter 4 is a discussion of graph theory, topology concepts and network properties which supports and builds on the concepts presented earlier in relation component interactions and structures. This chapter includes a review of recent work into the measurement and modelling of efficient topological structures, including small-world properties and topology growth models.

Part II uses the proposed ecosystem EC model and considers both how the model relates to the current field of EC, and then, secondly, what questions might be asked of such a model. Some of the questions are investigated and demonstrate the value of the model.

Chapter 5 begins the investigative aspect of this thesis using the proposed ecosystem EC model, and considers "what can we ask of such a model?". Work directly related to the new model is considered and compared. Key questions are identified (Section 5.5), selected and an appropriate research method described in preparation of the investigations undertaken in Chapter 6 and supported by the models and discussion of open research opportunities presented in Chapter 7. Chapter 8 summarises the results obtained and presents avenues for further investigations and work.

Chapter 2

Ecology, Ecosystems and Evolution

2.1 Introduction

This chapter presents concepts and properties from the domains of ecology, ecosystems and evolution. As will be shown in later chapters, these fields are extremely relevant to our understanding of complex systems, complex topologies and artificial evolution. More important for this thesis, however, is that an ecosystem model provides an excellent framework for capturing extended models of evolutionary computation, which is exploited in later chapters.

This chapter begins by presenting a brief introduction to ecology and a very useful model of organisational levels that allows us to clarify both the relevance of ecology, ecosystems and later evolution within this organisation model. Section 2.3 on ecosystems builds on the basis of ecology, and includes a brief history before presenting the key structural and functional concepts of ecosystems and a number of relevant terms and descriptions that support later sections of this thesis. Appendix A.1 contains a detailed glossary to support this chapter. Lastly this chapter looks at evolution in Section 2.4, and the mechanisms at work, from an ecosystems perspective.

In the next chapter (Chapter 3) we take the ecosystem model of evolution and reform it as a framework for evolutionary computation. An ecosystem model also presents some open questions for evolutionary computation that are described and discussed (Chapter 5) and in several cases investigated empirically later in the thesis.

2.2 Ecology

Ecology is recognised as *the scientific study of environmental systems including the distribution and abundance of life and the interactions between organisms and their environment*. Put very simply, ecology is a study of the economy of nature.

The subject matter of ecology is the entire world, both living and non-living components. Figure 2.1 presents a relationship for the organisational levels. Table 2.1 presents descriptions for some of the more relevant terms.

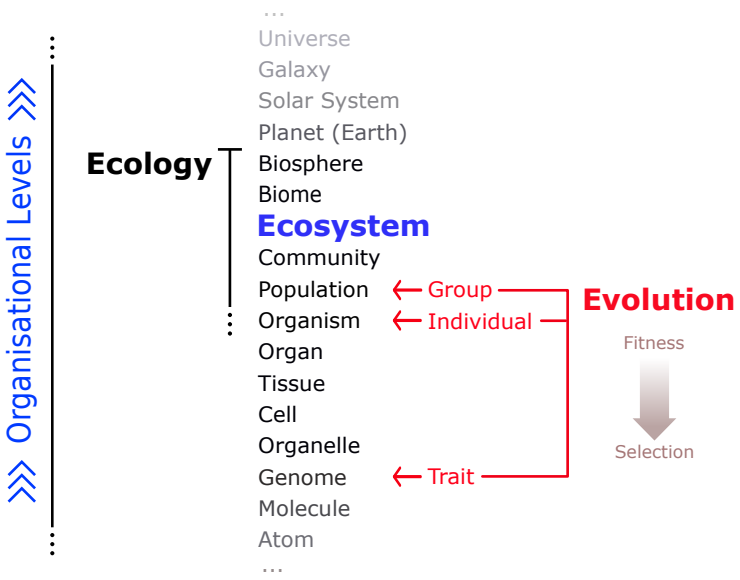


Figure 2.1: Organisational levels as they relate to ecology, ecosystems and evolution. See also Table 2.1 for a description of some of the more relevant terms.

When addressing a specific research question or study, however, the scope of matter is constrained to an appropriate size. In line with this there are four main areas of ecology, listed in order of increasing scale:

- **Physiological Ecology:** The response of a single species to its environment.
- **Population Ecology:** The abundance and distribution of individual species, and influencing factors.
- **Community Ecology:** The number and interaction of species in a given location.
- **Ecosystem Ecology:** The structure and function of an entire ensemble of organisms and species, their interaction with each other and their environment, and how this generates the whole.

Ecology questions relate to a specific organisational scale, such as the physiological ecology of a single plant species in response to salinity increases, or the population ecology of elephants in their native savanna habitat, or the global ecosystem ecology of food and fossil energy flow and its interaction with known species.

As an academic discipline ecology does not try to evaluate notions of good or bad any more than good or bad applies to mathematics or physics.

Industrial ecology is a more recent field that follows the flow of energy and matter (materials) throughout an industrial process, such as during the manufacture of cars. In this way the economy of the system can be measured with respect to a specified domain scope.

Ecology is, by its very nature, a broad and integrative discipline with links to many fields. This includes the physical sciences like physics, chemistry and the many areas of

biology, but also areas such as sociology, human geography, economics, agronomy, demography and so on.

Term	Description
Biosphere	Earth's largest ecosystem. The zone of air, land and water at the surface of the earth that is occupied by organisms.
Biome	Regional (large scale) ecosystem, composed of similar types of dynamic communities.
Ecosystem	The set of biotic factors, abiotic factors, interactions and processes between the organisms of multiple species and their environment.
Community	Different species interacting in a common space and time.
Population	Group of organisms (usually) of a single species occupying a given area at the same time. Typically, organisms with homologous alleles.
Organism	Individual. Any living thing; unicellular or multicellular.
Organ	A group of tissues that perform a specific function or set of functions.
Tissue	An ensemble of cells from the same origin that carry out the same function. Typically tissue cells are of the same type but not always.
Cell	The structural and functional unit of organisms, and the smallest unit classified as living.
Organelle	A confined and specialised sub-unit of a cell with a specific function. ("elle": small, a "little-organ" of the cell)
Genome	A full set of chromosomes and genes for an organism (a complete genetic sequence). The genome can be divided into chromosome and gene components (DNA and/or RNA).
Molecule	A sufficiently stable group of two or more atoms held together by a chemical bonds

Table 2.1: Selection of organisational levels and brief descriptions. See Figure 2.1 for an appropriate context for these terms in the organisational levels of scale.

There are both theoretical and empirical areas of ecology; applied ecology uses both, typically with the intention to regulate or bring about change to systems. Ecology encourages a method of observation and assessment that integrates how parts of a system fit together and interact; to understand how each part influences and is influenced by other parts, and how some whole systems are not predictable simply by their parts. When we are able to capture, model and replicate such complexity we gain new and potentially useful insights into natural and synthetic systems.

Two areas of ecology are particularly relevant to this thesis: *ecosystem ecology* and *ecosystem evolution*. They are presented in more detail.

2.3 Ecosystem

2.3.1 Definitions and Origins

As in the definition of ecology, ecosystems are composed of organisms interacting with each other and their environment. The specific concerns of ecosystems are the exchange of energy and the cycling of matter within the system, such that system level structure and functional processes emerge.

The term "ecosystem" was in fact coined in 1930 by Roy Clapham. However it was British ecologist Arthur Tansley who fully defined the term in his classic 1935 article on "The use and abuse of vegetational concepts and terms" [338, p299]. His description is often abbreviated, however the full passage is meaningful.

> But the more fundamental conception is, as it seems to me, the whole system (in the sense of physics), including not only the organism-complex, but also the whole complex of physical factors forming what we call the environment of the biome – the habitat factors in the widest sense. Though the organisms may claim our primary interest, when we are trying to think fundamentally we cannot separate them from their special environment, with which they form one physical system.

Tansley's article is significant to the field of ecology, not only because it clearly defined the ecosystem, but because it specifically argued against the then current idea of "superorganism" – a view that is now considered to have been a theoretical barrier to the field. Tansley also considered the "climax" concept, which is now viewed as a defunct step towards the more recent idea of "ecosystem dynamics" which encompasses the phenomena described by climax terminology.

Eugene Odum played a major role in the promotion of ecology including ecosystems. His seminal book "Fundamentals of Ecology" has been the predominant textbook of ecology since its first edition was published in 1953. His definition for an ecosystem ([264]) is as follows: [265]

> Living organisms and their nonliving (abiotic) environment are inseparably interrelated and interact upon each other. Any unit that includes all of the organisms (ie., the "community") in a given area interacting with the physical environment so that a flow of energy leads to clearly defined trophic structure, biotic diversity, and material cycles (ie., exchange of materials between living and nonliving parts) within the system is an ecological system or ecosystem.

2.3.2 Structure and Function

An ecosystem represents the notion that living organisms continually interact with each other and their environment, and that they are able to produce complex systems with emergent properties. Ecosystems embody such concepts as "the whole is greater than the sum of its parts" and that "everything is connected".

With respect to the levels of organisation scale presented earlier in Figure 2.1 ecosystems are placed above the community level where populations of species interact with each other, and below or equal to the biome level (regional ecosystems) and below the biosphere – the largest known organic ecosystem.

Although it is reasonable for the purpose of study to define fixed system boundaries (such as component organisms, population groups and specific interactions), ecosystems are by their definition inherently conceptual. There are, however, clear ecosystem requirements: biotic (living) and abiotic (non-living) factors, interactions and energy.

A description of interactions and process is essentially a description of a dynamic topology; understanding and representing which parts of the system interact, when they interact, and how they influence each other. It should also be recognised that there are dynamic and seasonal influences (abiotic factors) which consequently modify interaction dynamics and characteristics.

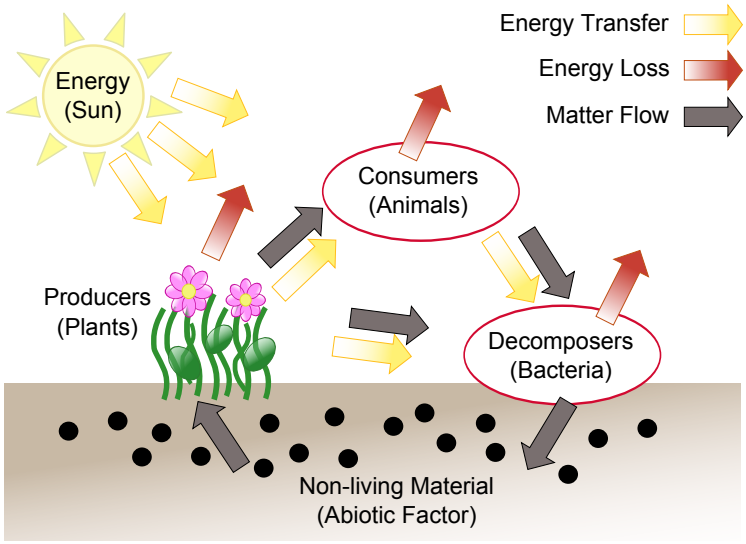


Figure 2.2: The flow of energy and matter within an ecosystem model containing biotic and abiotic factors, interactions and open energy driven processes.

As natural systems are open energy systems, energy is always lost during conversions and interactions except for those factors that are able to utilise energy, such as primary producers (Figure 2.2). Without energy to facilitate processes and interactions, an ecosystem will cease to function.

The cycling of matter through an ecosystem is of most value when it is understood to be closed; understanding how energy is used to move material is a defining process for an ecosystem. Well known matter cycles include the carbon, nitrogen and oxygen cycles of the biosphere.

Two common topology descriptions for ecosystem interactions are a “food chain” and a “food web”. Food chains are a listing of organisms in order of primary producers, secondary consumers and so on, while a food web indicates all feeding interactions between organisms (species) in an ecosystem.

To summarise, the *components* and organisational groups within an ecosystem (with respect to the levels of organisation presented earlier in Figure 2.1 and Table 2.1) are:

- an **environment** including a specific scale and the specific nature (the *topology* and *process*) of interactions between all components;
- **communities** composed of multiple species and located in overlapping locations of the environment;
- **populations** that represent all the individuals of a common species in the environment, including sub-populations for specific sub-groups such as breeding adults, juveniles or male/female gender; and,
- **individuals** that interact with the environment and other individuals of the community (using *energy* to enable processes).

2.3.3 Life Cycle Model

The life cycle of biological species is an iterative process that underlies and facilitates evolution, as illustrated in Figure 2.3 for sexually reproducing diploid organisms. Note the existence of the *population structure*, the components of *selection* and *reproduction*, and the *development* and *insertion* of offspring into the population. Note also that there is no implicit *initialisation* or *termination* – such events are not defined for this simple model.[1]

If the environment of a population has limited resources, or individuals are subjected to competitive pressure in order to survive, any features or adaptations that provide individuals with an advantage are more likely to be preserved, and hence passed on to new individuals. This is the classic "survival of the fittest" phenomena described by Charles Darwin.[2] The process of evolution on the life cycle model is discussed in more detail in Section 2.4.

2.3.4 Community Model

An ecosystem *community* is all of the organisms that occupy a specified location at the same time. Community is a useful level of organisation within an ecosystem scale, able to contain many different and intermixed organisms.

While there are no explicit subgroups within a community, it is possible to take different implicit *views* of the community as presented in the examples of Figure 2.4. Each view is an answer to a context specific question such as "where is species A?", "what are the interactions between species A and B?", "where are all the adults" or "what is the distribution of juveniles?". For a specific individual, answers to local neighbourhood questions such as "where can I move to?", "what is my competition?" or "who are my potential mates?" and similar are required for local activity (interaction).

[1] They might be defined within a larger view of successional community change which includes speciation and species extinction events.

[2] The term "survival of the fittest" was actually coined by philosopher Herbert Spencer when discussing Darwin's theories.

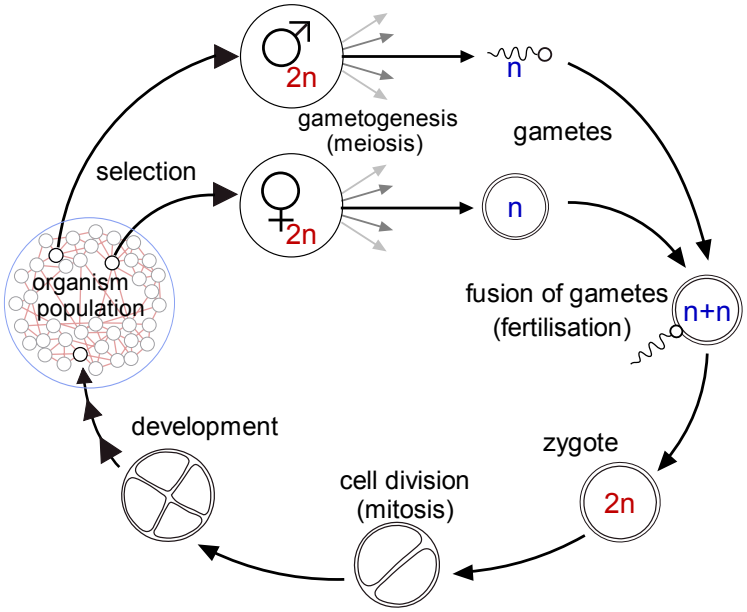


Figure 2.3: The life cycle of sexually reproducing biological organisms. Note the selection of parents from the population, the creation of gametes via meiosis and the fusion of gametes during fertilisation to produce a new organism that develops in the population. In this model n represents half the genetic material from a *diploid* parent individual that contains a double set of genetic information (as present in humans).

The ecosystem includes the community and the *environment*. Properties such as occupation density and interactions are limited by the environment. Individuals may interact and change the environment, which in turn alters the interactions between individuals of the community. The environment may include *niches* that limit occupation to particular species or individuals.

Communities can also be defined by what is exchanged in interaction, such as cultural ideas, beliefs or behaviours, where members of a community share such *memetic* information. The exchange of memetic information can occur at a much higher rate than other factors that influence evolution, and has been included as the basis for some artificial models of evolution (see Section 3.3.7). Interestingly, the complexity of memetic information does matter to the rate of transfer in the community: simple and influential ideas propagate more successfully than complex ones of similar influence.

The definitions, roles and validity of memetic ideas have stimulated much debate and controversy, likely due to its topical nature; the transfer of ideas, philosophy, religion and other beliefs. It is clear, though, that memetic transfer does occur within an ecosystem community and that it can have strong influence that may need to be taken into account when trying to understand or describe parts or views of an ecosystem.

2.3.5 Components, Properties and Processes

Terminology and Scope

There are many components, properties and processes that can be defined. This section is not an attempt to define all of the terms and models of the ecology and ecosystem

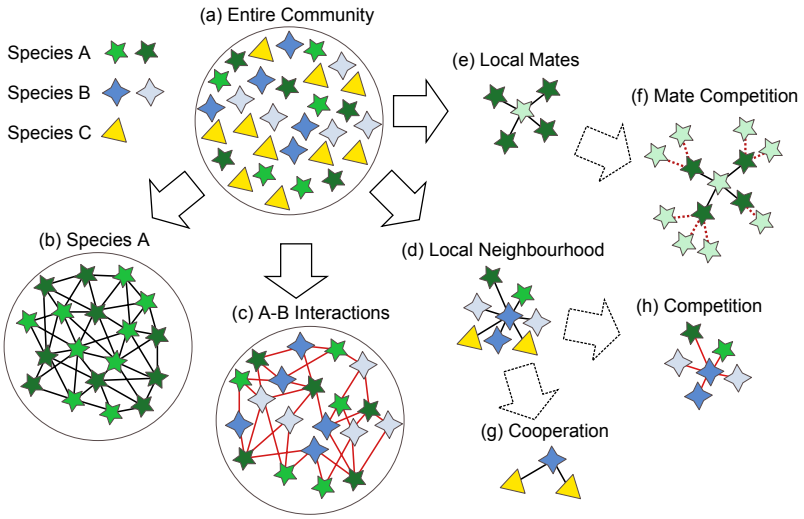


Figure 2.4: Representation of community views. The entire community (a) is composed of three species (A, B and C), two of which include sexual reproduction (A and B). (b) is a single species view, (c) a view of the interaction between A and B species (local interactions only), and (d) the local interaction neighbourhood for a single individual. Similarly, (e) is a local pool of mates for a single sexually reproducing individual and (f) shows the corresponding pool of mate competition as a result. (g) and (h) are simple local cooperation and competition interactions.

fields, however it should provide enough relevant details to support the models used in later chapters. Further support is provided by the ecology glossary in Appendix A.1.

Environment

The environment is a combination of all external (extrinsic) conditions and potential effects on the inner environment (Table 2.1). It could also be defined as all abiotic factors (such as the physical limitations) of nature that create a *heteromosaic* of conditions.

Natural environments are rarely static and so the survival of any species depends on their ability to cope with environmental changes (Table 2.2).

Term	Description
Cyclic	Periodic (rhythmic) changes, such as seasonal variations, day/night cycles and lunar (tidal) influences.
Directional	A sustained direction of change, typically over a long period of time, such as glaciers, erosion, siltation, salination.
Erratic	Change without rhythm or periodic nature, such as earth quakes, tsunami, volcanoes, fires, land-slides, cyclones or hurricanes.

Table 2.2: Three distinct temporal terms to describe environmental changes.

Community

Community has already been clearly identified as a crucial distinction of the ecosystem idea (Table 2.1), and there are many important aspects of structure and interaction on which

the definition of community is important. As there are many terms related to sub-groups within a population, they are left to later relevant sub-group topics for discussion.

Term	Description
Community Species	
Structure	A list of species order by abundance.
Interaction	The number of species that can sustainably flourish.
Density	The number of individuals (of all community species) per unit of space. Absolute or relative.
Community Dynamics	
Stability	The degree to which a dynamically stable community will return to its original state after a disturbance.
Resilience	The speed of return to a stable state.
Resistance	Ability to resist (avoid) change to the current state.
Fragile	A community that is dynamically stable only within a limited (narrow) range of environmental conditions.
Robust	A community that is dynamically stable across a wide range of environmental conditions.

Table 2.3: Community terminology related to both species and dynamic changes

Community is not only the structure and number of interactions between species, but also dynamic qualities such as *stability*. Table 2.3 includes descriptions of terms related to both species composition of a community and dynamic community changes.

One theory of community stability suggests that initially, communities are relatively simple in structure and interactions, and as evolution progresses develop to very stable and complex systems. This does not imply that all complex systems are stable or that very stable systems are complex. Perhaps incremental development of a large complex system requires intermediate stable steps, although periods of instability may be equally as important.

Population

A population has already been defined *as the group of individuals of a single species occupying a given area at the same time* (Table 2.1). It is at times useful to include multiple species depending on the question being asked of a system, but this discussion is limited to a single species idea.

Within a population the number of individuals, and the composition of age distribution, changes over time. A *population pyramid* is a way to show the age structure of a population by breaking ages into different groups (infant, youth, elder etc) diagrammatically placing the youngest age class at the base and stacking successive age classes above it.

Similarly, a *life table* gives a summary of the age or stage-related survivorship, based on natality and mortality rates. This table can be constructed from a single static sample

of the population, or developed over the cohort life time of a sample group from birth to death.

The *natality* (birth) and *mortality* (death) rates, and the *migration* rates of individuals in (*immigration*) and out (*emigration*), provides an overall picture of a populations *growth*. See Table 2.4.

Term	Description
Natality	Birth rate of new individuals per unit of time.
Mortality	Death rate of new individuals per unit of time.
Immigration	Outside individuals entering population per unit of time.
Emigration	Individuals of the population leaving per unit of time.

Table 2.4: Population growth rate terminology

The total size of a population at a new time step $N_{(t+1)}$ is the old population size N_t and the summation of the intrinsic growth rate factors r; births B_t, deaths D_t, immigration I_t and emigration E_t. See equation (2.1):

$$N_{(t+1)} = N_t + B_t - D_t + I_t - E_t \tag{2.1}$$

$$= N_t + r_t \tag{2.2}$$

Influences of the intrinsic growth rate r_t help to characterise population *dynamics* and *fluctuations*, including *cycles* and other changes in *density* as well as any *regulation* factors. Overall, population regulation is either extrinsic (outside of the population) or intrinsic and directly related to the populations properties and behaviours (such as density).

Two indicative population growth models are *exponential* (indicating little or no limits) and *logistic* (indicating environmental density limits). Limited growth is commonly *density dependent* such that death or birth rates change in direct response to the population density. It is possible that density dependent changes may *over-compensate* or *under-compensate* such that after a significant disturbance a new stability of the population emerges which is different from its initial balance.

The *carrying capacity* K is the maximum sustainable equilibrium limit of the intrinsic growth rate r for a given population density N. See equation (2.3):

$$\frac{dN}{dt} = \frac{rN(K-N)}{K} \tag{2.3}$$

The term *population ecology* describes a study which focuses on the changes in size and density of a population over time and space, and the contributing factors. It is a useful measure over generational change.

Species

Within a taxonomy, a *species* is a group of organisms whose members have the same structural traits and are able to interbreed with each other. *Species diversity* takes into

Term	Description
Density	The number of individuals per unit of space.
Fluctuations	Variations over time in the population size.
Cycles	Oscillations in population size (between high and low density).
Dynamics	The variations in population size and density over time and space.
Regulation	A population size or density regulated (limited) by some factor (such as density, competition or resource limitations).
Carrying Capacity	A measure of equilibrium between births and deaths at the maximum sustainable population size.

Table 2.5: Population dynamics terminology

account both the *richness* and relative *abundance* for the specified environment. Within an ecosystem community group of interacting species, the top predator affecting all organisms of lower trophic levels is known as the *keystone*.

Term	Description
Diversity	A community measure that takes into account both the relative species abundance and richness.
Abundance	The number of individuals belonging to a given species.
Richness	The total number of species in the community.
Keystone	The top predator within a community of species.

Table 2.6: Species related terminology

Distribution

Individuals will *disperse* within the environment, some according to generational process (such as offspring moving away from parents), or due to neighbourhood density preferences (from high to low). Alternatively individuals can choose to aggregate or *cluster* for mutual benefits such as "safety in numbers". Interestingly, some species[3] develop differently depending on density pressure, a process known as *polymorphism*. Dispersal is limited by the abilities of the individuals and the current environment.

Spatial distribution can be constrained by factors such as altitude, latitude and the clustering behaviours of individuals. Distributions may be *random* or *regular*, where regularity is typically due to behaviour such as territorial size, mating or parental care limits.

Habitat describes where an organism can be found, and a detailed description might include *cyclic* changes such as seasonal variation or weather patterns. A single ecosystem environment may support a rich diversity of habitats.

[3]The East African locusts are a widely cited example. Normally they develop into a sluggish and solitary bright green form. However, if maturation occurs in large constricted groups, they metamorphosis into a darkly coloured, highly mobile form enabling rapid dispersal.

Term	Description
Dispersal	The spread of individuals away from each other (from parent or birthplace to breeding locations).
Clustering	The grouping of individuals for mutual benefit.
Distribution	The spatial range of a species, or the spatial arrangement (pattern) of a species in a habitat.
Habitat	The place an organism lives, or is usually found.
Island	(Habitat) An isolated (geographic) habitat.

Table 2.7: Environment distribution terminology

An ecological *niche* is a region where species can exist indefinitely. A niche description includes the environmental conditions, resources levels, population density and growth rate factors necessary for a stable system. All of the factors can be viewed as an n-dimensional space where the niche range of conditions is a closed volume.

A species may be forced into a *realised* niche other than its preferred fundamental niche due to competition. (See Table 2.8 for niche terminology.)

Term	Description
Complimentary	The tendency for coexisting niche species to differ along another niche dimension (resource).
Fundamental	The potential (idealised) range of all environment conditions in which an organism can thrive.
Realised	The range of the fundamental niches that a species occupies due to competition limits.
Breadth	The potential (idealised) range of all environment conditions in which an organism can thrive.

Table 2.8: Niche terminology

Modelling the spread of disease with a population (or a community) requires a valid model of the disease organism life cycle and how it effects a host. It also requires an understanding of how an infected host behaves. (In particular how they interact and possibly infect others.) Some diseases are successful in propagating because of their rapid growth and highly infectious survival (*epidemic* or *pandemic* spread), while other diseases exist and survive within a population indefinitely by having a minimal (or dormant) effect on hosts (*endemic*).

Term	Description
Epidemic	Disease affects *large* proportion of population at the *same* time.
Endemic	Disease affects *small* proportion of population *all* the time. (The disease is retained in the community).
Pandemic	Disease affects *entire* population at the *same* time.

Table 2.9: Disease spread terminology

Interactions

There are three community models of interaction between individuals within an ecosystem community:

- **Symbiosis:** Two or more organisms living in a relationship that benefits at least one of them. If the characteristics of the species evolve together (in concert) it is considered an example of *coevolution*. Such symbiotic relationships are the norm, not the exception. The three kinds of symbiosis are *commensalism*, *mutualism* and *parasitism*.
- **Predation:** A *predator-prey* relationship between animals, animals and plants, or between plants. The predator consumes the prey. Predator-prey cycles are a useful way to represent the oscillations of the two populations over time. The age (and hence traits) of population members can strongly influence the predator-prey relationship.
- **Competition:** Two or more individuals in competition for resources, either *intraspecific* (within a species) or *interspecific* (between different species).

Symbiotic interactions are commonly described in terms of interactions between two individuals of different species, such as a *host* species and a typically smaller *symbiont* species. However symbiotic relationships are not limited to two organisms or species; what a single organism cannot achieve alone, an ensemble may. The three kinds of symbiotic interaction are listed and described below:

- **Mutualism:** Both species benefit (a host and a symbiont).
- **Parasitism:** One species benefits (symbiont), the other is harmed (host).
- **Commensalism:** One species benefits (symbiont), while the other neither benefits or is harmed (host).

Parasitism can be considered a kind of predation, though it is characterised by a prolonged and close association between symbiont and host organisms, often with particular specialisation of the parasite.

Competition describes a diverse range of interactions, not only between species but also within a species population. See Table 2.10 which describes terms related to competition balance (*symmetrical* or *asymmetrical*) and the nature of influence (*exploitation* or *interference*) competitors have on each other.

The competitive exclusion principle is that *two species with similar environmental requirements cannot coexist indefinitely in the same niche*; one species will eventually be excluded or shift to a different niche. Species can also adapt behaviour such that they may minimise conflict and coexist.

Time plays as very important role in the establishment of species; the order and interval between species introduction to a habitat can be critical. Successful spatial variation can facilitate rich interactions between species.

Term	Description
Interspecific	Competition between individuals of different species.
Intraspecific	Competition between individuals of the same species.
Symmetrical	Balanced (evenly matched) competition.
Asymmetrical	Unbalance (strong-weak) competition.
Exclusion	Interspecific competition removal.
Exploitation	Use of a resource removes it from competitors (first-come-first-served principle).
Interference	The physical exclusion of competitor by another (territorial behaviour).

Table 2.10: Competition related terminology.

Succession

Succession is an orderly progression of changes in community composition due to changes in environment condition. It is a continuous change rather than an episodic or interrupted set of states or phases of development, and the rate of change can vary depending on the composition.

There are several theories that consider how a disturbance can open a habitat area and how this might affect later successional species as they populate an area:

- **Facilitate:** Pioneer species that, when established, enable other changes and species into the habitat.
- **Tolerate:** Existing modifications or occupation by early inhabitants have little or no effect on the subsequent successional species.
- **Inhibit:** Existing modifications or occupation by early inhabitants prevents or makes the environment less suitable for later successional species.

Consider also how an established habitat will respond to a large scale erratic change (such as flood or bush fire). Such events can modify a habitat so that it be rapidly populated by different successional compositions of species that would otherwise be excluded from or inhibited from the habitat.

2.4 Evolution

2.4.1 Origins and Fitness

A common definition for evolution is *a change in allele frequencies in a population of individuals over time*. As has already been noted in Chapter 1 the work of Wallace, Darwin and Mendel are all significant contributions to the development of theories in evolution and population genetics.

The notion of *fitness* is used throughout any discussion of evolution. A simple and effective definition is that *fitness represents how good an individual is in its current environment* due to the qualities or capabilities it has. The limitation of this definition is the concept of "good".

A more subtle understanding of fitness is needed. It is possible that two individuals with the same genetic material (genotype) will present differently (phenotype) due to past environmental influences (developmental) or current conditions. The current conditions include competition and available resources. From this we can see that fitness is a relative and contextual measure which manifests in the phenotype form and not a simple or specific absolute value.

Finally, the biological distinction of fitness is defined as *the capability of a particular genotype to reproduce* (the fitness value being retrospectively proportional to an individual's genes present in the next generation). Note that this means that an individual's fitness is not necessarily their own capability to reproduce, but of the genotype they represent.

2.4.2 Mechanisms

There are several mechanisms that can lead to changes in the relative proportion of gene types present in a population. These include *selection*, *mutation*, *genetic drift* and *gene flow*. Perhaps the most well know mechanism is selection, although there is some misunderstanding that this is always the dominant, or only, mechanism.

- **Selection** (or "natural selection"): Changes in allele frequency due to the relationship between values of a heritable trait and measure(s) of an individual's fitness.
- **Genetic drift:** Generational variations in allele frequency due to stochastic (random) processes.
- **Mutation:** Heritable changes to genetic information including both small scale and large structural changes in genetic material.
- **Gene flow:** The migration of genetic material from one population to another.

Significant consequences of these mechanisms include *adaptation* and *speciation*.

2.4.3 Selection

Selection, sometimes called Darwinian selection or natural selection, refers to the mechanism whereby particular varieties of genes (alleles) that confer a fitness advantage will increase in frequency from one generation to the next. Put very simply, "good" genes (that contribute positively to fitness) are selected and increase in favour of "bad" genes because the individuals that contain the "good" genes are fitter by some measure of the group, and so more likely to survive or be selected for reproduction.

But what determines good or bad? Generally this is an environmental concern, given that selection acts through the environment on the physical form of an individual (the

phenotype) and then consequently to its heritable components (the genotype). We can consider then that the relationship between the environment, the phenotype and the genotype may be simple (see Figure 2.5), but is more likely a complex interacting topology. A real and complete pathway map from a genotype to the phenotype, including interactions between genes, traits, individuals and the environment, is a complex network. Understanding the process of selection on a complex network is a non-trivial matter. Considering this complexity is a key concept to understanding the influence of selection.

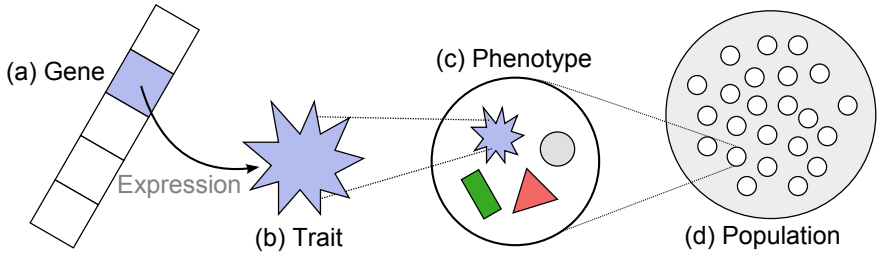


Figure 2.5: The pathway of (a) a single gene expressed from the chromosome to (b) a trait which, in combination with other expressed traits, forms (c) the phenotype instance of the individual. A collection of individuals forms a (d) population.

A principle concept is the relationship between an organism's trait (such as colour or size) and the organism's fitness. This idea is shown in Figure 2.6. A simple *linear selection* relationship is a suitable starting model. If the fitness value increases with respect to an *increase* in trait value, the trait will be under *positive directional selection* (pressure). Similarly, a fitness increase in proportion to a *reduction* in trait value is *negative directional selection*. A trait with no influence on fitness is under *neutral selection*.

Although useful as an initial model, linear directional selection is limited. Fitness in natural systems is really a utility measure of a trait with respect to the current environment properties. The fitness of any specific organism is localised experience over a period of time. A better model of trait fitness utility includes non-linear and dynamic relationships with the environment. (See Figure 2.6(b).) Consider also that the trait-fitness relationship may not be continuous; some trait values will result in non-viable organisms that never survive.

Under the pressure of selection, traits may change their distribution qualities within the population. Figure 2.7 shows an example of neutral and simple linear directional selection applied to a single normally distributed trait in a population. Unless genetic drift (see Section 2.4.4) is occurring, neutral selection should have no effect on the distribution of the trait values. Linear directional selection (Figure 2.7(b)) will skew the initial distribution, and eventually evolutionary constraints will limit any long term selection changes.

Stabilising selection (Figure 2.7(c)) demonstrates how the frequency distribution of a trait narrows around the mean value. The mean frequency does not change but the variance decreases. Recall that variance is a critical raw material for evolution, and so reduced variance is a detrimental result for continued evolution. It also demonstrates how a population can be under the influence of selection even though there is no change in the trait, just its distribution. This is a common issue in nature where stabilisation causes

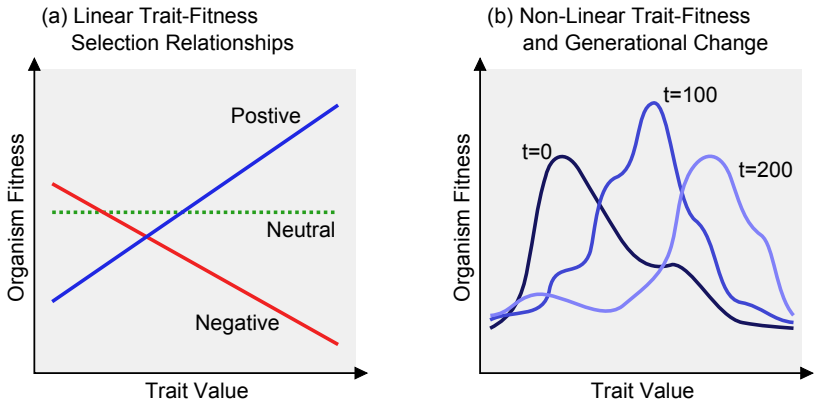


Figure 2.6: Trait-fitness relationship graphs. (a) Positive, neutral and negative linear fitness relationships for a single trait value. (b) A non-linear fitness relationship at different stages in a population's generational history (t) demonstrating the trait utility concept.

a trait convergence in the population. The result is a loss of genetic diversity and so a reduced ability for the species to adapt and select from appropriate trait variations.

Disruptive selection (Figure 2.7(d)) also retains the same mean of the trait frequency distribution in the population, while also specifically reducing the mean and increases the width of the distribution. Essentially, the two extremes of the trait value provide the best fitness, and this will cause the population to split over time into sub-groups. Potentially such a split will create enough trait distribution differences that reproductive incompatibility may result and so this is an important mechanism of speciation.

One additional selection type is that of *sexual selection*, which can apply to sexually reproducing organisms. Assortative mating can be positive (mating with similar) or negative (mating with dissimilar) individuals. Essentially, males or females select or compete for their mates. It is thought that assortative mating is very important to some processes of speciation. Traits that are affected by sex-related assortative mating are said to be "under sexual selection".

Sexual selection may only affect one sex (such as the suitor gender) and not both. Sexually selected traits may decrease overall individual fitness. For example, bright plumage for a bird may attract a mate, but may also make it more conspicuous and likely to be consumed by a predator. It reiterates that trait-fitness relationships for organisms can be a complex, multi-objective and dynamic function.

Coevolution and *coadaptation* include the interaction of multiple species, such that the fitness selection curve is then dependent on the current (dynamic) frequency of genes in another population or species. Again, this emphasises the non-linear and dynamic nature of fitness selection relationships.

2.4.4 Genetic Drift

Genetic drift is a very influential but often unappreciated mechanism. It is, essentially, changes in trait distribution due to stochastic processes that can lead to loss of genetic material. Genetic drift acts without the need for selection which may act more directly

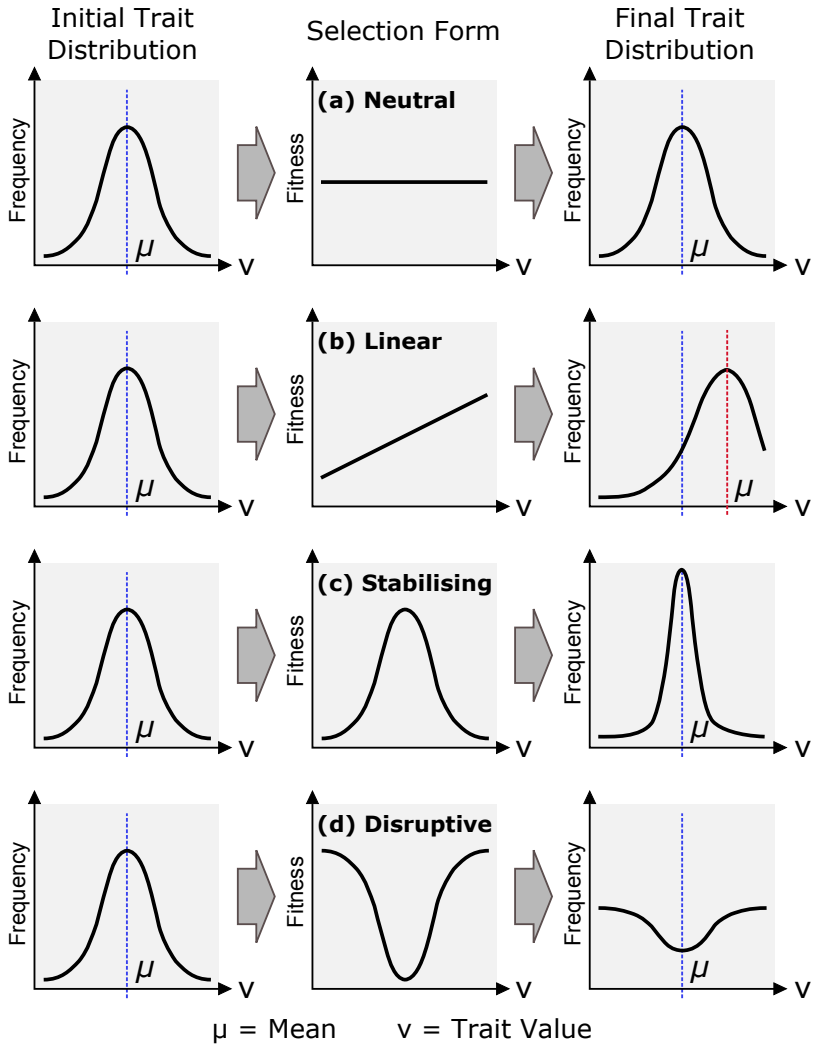


Figure 2.7: Graph illustrations of (a) neutral, (b) linear, (c) stabilising and (d) disruptive selection as it applies over time to a single trait, initially normally distributed in a population.

against specific trait frequency. Drift can occur whenever there is a random or stochastic process involved in the generation (reproduction) of new individuals.

Consider sexually reproducing organisms with diploid chromosomes; meiotic cell division – which is an effectively random halving of the diploid chromosome – creates gamete germ (sperm or egg) cells, which are then fused in reproduction to form a zygote. In this process of gametogenesis there is a random chance that some variations may not be copied as frequently as others, which in turn means that the population can be affected by genetic drift.

Likelihood of drift equating to either loss or fixation of a particular trait in the population is related to the population size. A small population is much more likely to be affected by drift than a large one. Also, as this is a stochastic process, the longer the time period the more likely that drift will affect the population.

Fragmented populations are more at risk than connected populations because they are effectively small isolated subgroup "pockets". It should be evident that the nature of

interactions and the topology of a population have a significant effect on drift as well as selection based convergence.

The mechanism of drift is not limited to genetic processes of recombination. It may also be due to random selections from the population, such as random survival selection as a result of a catastrophic event (fire, flood, volcano), or the random selection for reproduction. If the subset that survives and reproduces is not a representative sample (simply due to the reduced population size) then the diversity of the initial population will be lost and the new population will have drifted to a different distribution.

In a similar way the "founder effect" occurs when a small group colonises and develops a new habitat, such that the small sample of genetic material results in a different genetic diversity in the colony from the original population. By example, this has created serious health problems, such as prevalence for particular genetic disorders, in historically small and mostly "closed" colonial parts of the United States.

In natural systems there is rarely neutral selection with only the effects of drift. However, drift can always play a part as there is always interaction between organisms and species in an environment, and the movement of species and changes to their environment can induce significant drift related stochastic changes.

2.4.5 Gene Flow

Gene flow is essentially the movement of individuals between populations and, by so doing, a transfer or "flow" of genetic material. This can have neutral, positive and negative effect to the diversity of a population, partly regulated by the size of the population and the size of the flows, but also by the frequency of movement and the possible specific selection of migrants in or out of a population.

Models of gene flow may be very important to speciation. It is thought that periods of equilibrium, punctuated by either migratory or traumatic events, are important means of both refining adaptations (exploitations with respect to a specific environment) and also allowing for rapid changes (exploration with respect to a new environment).

2.4.6 Mutation

The term mutation is used for many different mechanisms, but all have a key similarity: they introduce heritable variations. These variations can be at the smallest of scales, where changes to the acid-base pairs of DNA result in different nucleotides, or at larger structural levels where sections of genetic (heritable) material are inserted, removed and reordered.

One of the most likely causes of mutation changes in organisms is a "copy" or transcription error, given that genetic material must be copied for all cell division and so this happens many times. Of the changes that might occur there are several possible consequences for organisms:

- **Severe Detriment:** Cell (or organism) will cease to function.
- **Minor Detriment:** Cell function is impeded, but still functions.

- **Neutral:** No change to the function (based on the currently expressed genes).
- **Minor Benefit:** A benefit but it is not significant or radical in function.
- **Significant Benefit:** Increases organisms fitness.

Significant beneficial changes are unlikely simply because of the complexity of the genetics and organisms involved. However, given the time period over which evolution can operate, evolutionary progress waits for the rare and beneficial changes that may present.

For changes to be inheritable they must be involved in reproduction or somehow incorporated or selected for the genetic material used in reproduction. Because of this the most likely location for mutational variation to have an inheritable influence is during reproduction interactions or processes. For haploid organisms (animals), this is in the production of gamete cells (gametogenesis) or in the fusion of gamete cells (fertilisation).

2.4.7 Speciation

Speciation is the processes whereby new biological species may arise. The mechanisms of speciation are rarely observed and strongly debated. There are four main theoretical modes proposed: allopatric, peripatric, parapatric and sympatric speciation. A visual representation is shown in Figure 2.8 and each is described briefly:

- **Allopatric:** The population is split into two isolated groups. (Possibly the result of habitat fragmentation.)
- **Peripatric:** A relative small proportion of the population is isolated in a peripheral population. (This is directly related to the founder effect.)
- **Parapatric:** Localised mating frequency change related to environment niches.
- **Sympatric:** Genetic divergence within a single population in a homogeneous environment. (Possibly the result of disruptive selection.)

In each case, the speciation process begins with a single population, and then some type of initial division or change occurs, which results eventually in two populations of genetically dissimilar species that can no longer interact reproductively (species).

Of all the modes presented, the strongest evidence supports allopatric speciation while the modes of parapatric and sympatric speciation are less supported. All populations can undergo genetic drift or dissimilar selection pressure which is likely to play a part in all modes to some extent.

The causes of large scale habitat change and fragmentation can include environmental alteration due to such events as fires, floods, earthquakes, landslides, river changes and volcanoes, or the migration of other species. Island geography is often used to illustrate the long term changes that can occur within an isolated genetic system in support of allopatric speciation.

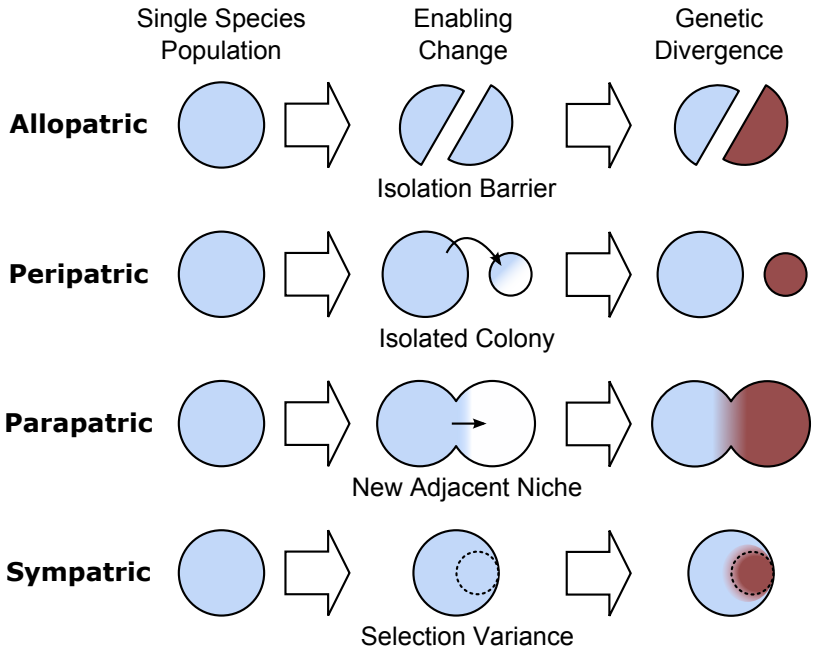


Figure 2.8: Speciation modes showing a uniform population, the enabling change and the resultant divergence and classification of separate (genetically isolated) species populations.

Environmental niches can occur along an otherwise equivalent environmental gradient due to factors such as contamination or species competition. Niches are required for parapatric speciation.

Sympatric speciation is still one of the most debated modes of speciation, given that it requires an almost spontaneous divergence within an environmental gradient. Disruptive selection is one of the most popular mechanisms proposed to explain how (if not why or if) sympatric selection might occur.

2.4.8 Limitations

There are inherent limits placed on any evolving system. All limits are intrinsically part of the environment; evolution is simply a interacting process that can occur within the material and entities contained in an ecosystem. Several of these limitations are worth considering in more detail: genetic variation, developmental stages, time, physical limits and the interaction of traits.

Genetic Variation: The available genetic material of the parent population limits the possible qualities offspring may have (plus some occasional mutation). This is the largest of constraints. Variation is the raw building material of evolution to work with. This constraint is also evident when populations are affected by small sizes and sampling issues (such as stochastic drift or founder effects).

Developmental Stages: Some organisms require distinct developmental stages in their phenotypic development. These stages could be changed by variation in the genes that control development regulation, though stages may be critically dependent on

the environment. Multiple significant changes may be required to "skip" a development stage or to alter environment requirements, and so such changes are less likely than other variations.

Time: Evolution needs significant periods of time, and the larger the organisation scale, the longer time period required for evolution to effect change. Time is typically measured in "generations" and this is significantly different for single-cell bacteria than it is for larger, longer living, multicellular organisms.

Physical Limits: The physical limitation and properties of an environment do not scale at the same rates. As such, an increase of a single trait value may not be supported by increases in other required traits, or limited by physical relationships. For example, an ant could not be scaled up to the size of an elephant because there are many physical (structural, material, chemical and so on) limits that prevent this.

Interaction of Traits: Traits are rarely independent, either because they are located near each other on a chromosome and so may "hitch-hike" under selection, or because of the complex interacting network of gene expression. (See Figure 2.9 and Figure 2.10.) The genetic interaction of pleiotropy and polygeny are particularly important. Similarly phenotype traits can interact with each other with respect to fitness.

Not only is genetic variation in a species required for selection to operate, a limited "gene-pool" is also a known high-risk. For example, if a pathogen were able to overcome a high-frequency defence-related genetic trait, the entire species would be susceptible.

Theories suggest that biological systems may be able to avoid high-frequency traits through several processes: favouring *neutral substitutions* (that is, allowing or encouraging genetic diversity as a "built-in" mechanisms of evolution); *intrinsically diversify* given any different environmental conditions (because this is a successful strategy); or that *fitness favours unique alleles* in such a way that high-frequency alleles are considered "less-fit".

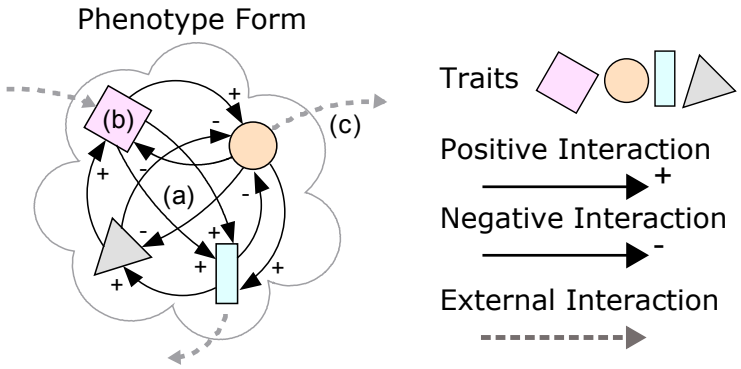


Figure 2.9: Trait interaction network. The properties of an individuals' phenotype form (a) are a result of the expressed traits (b) and their unique interactions with other traits and the environment. 'Fitness' can be illustrated as an output (c).

Pleiotropy is where a single gene can influence multiple traits. Therefore, anything that affects one gene may have multiple trait effects, beneficial and detrimental. Some

influences may even be antagonistic toward the same trait (Figure 2.10). Similarly, a *polygenic* trait describes a phenotype feature that can be attributed to two or more genes interacting (and the environment), and in order for the trait to change there may be several interdependently located genes that would need to be altered. Clearly these gene expression interactions can affect the way selection alters population allele frequency, and limit the likelihood that simple mutation will advantageously alter the trait.

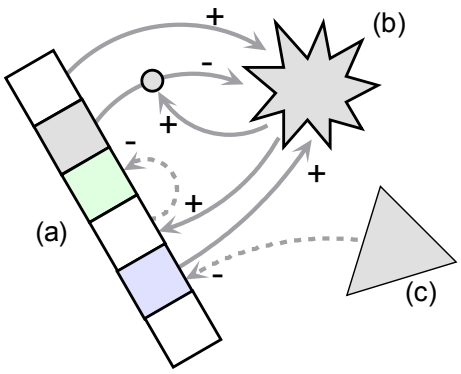


Figure 2.10: Illustrating gene expression network complexities: pleiotrophic and polygenic interactions. Note how (a) multiple genes can interact with positive and negative effect to the (b) expressed trait. Similarly, it is possible for (c) prior expressed traits to have influence.

Lastly, traits may be linked physically in their phenotype form such that multiple phenotypic traits cannot all be increased. For example, given a limited energy budget for an organism, the energy needed to allow an organism to grow large may increase its survival, but require a longer juvenile development stage. As a result this might delay fertility and so reduce the species survival probability, and hence reduce fitness.

2.4.9 Evolution and Organisational Scale

As we have already considered an organisational view of ecology it is worth pointing out the levels at which evolution can occur. As long as there is variety in the heritable traits that an individual might contain evolution may apply.

Richard Dawkins popularised [71, 72] a reductionist gene-level view of evolution by considering a chromosome as a population of genes. In this micro-evolutionary model a gene is the individual, and the chromosome is the population. Certainly for simpler organisms such as viruses and bacteria this view has several advantages, and is still popular in molecular evolutionary research.

Moving up from a micro-evolution scale and a classic organism evolution scale, we can consider macro-evolution. This view is particularly useful if there are traits that can be best described at a macro-organisational level and no lower. As long as a trait is heritable in some fashion and related to fitness, and there is variability required by evolution, a group can be considered a single unit.

For example, population density of bacteria growing on a particular media may be best described as a *group trait* (an emergent property). Note that although population density is a quality of individual bacteria, density contributes to the entire culture group.

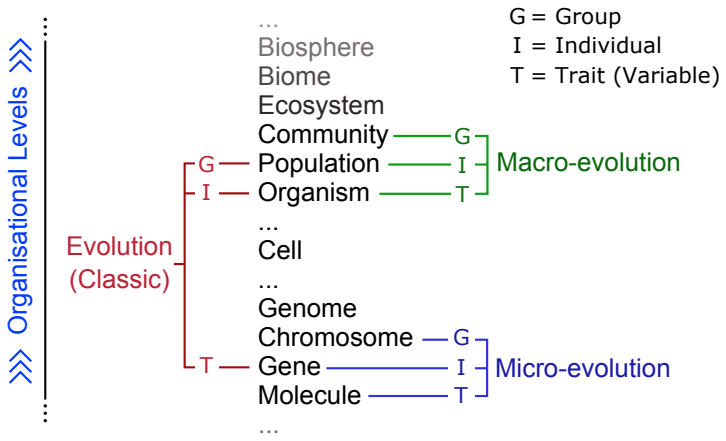


Figure 2.11: A representation of classic, micro- and macro-evolution models as they relate to different levels of the organisational scale presented. In each case there are three organisational levels used for *groups*, *individuals* and variable heritable *traits*.

Different groups or species of bacteria could then evolve as the individuals under selection, where fitness is related to groups' population density.

Given these scale-independent concepts of evolution, we can restate the earlier common definition for evolution: *a change in frequency of variable traits in a group of entities over time*. In this way the unit of a trait (unit of evolution) is not fixed to a specific level (allele) and the idea of a group of entities is applicable across organisation scales (not just populations of organisms).

As we climb the ladder of organisation scale, and due to the effect of aggregation, we typically lose entity variation and reduce the total number of entities involved. As a consequence, evolution – which requires variation in order to effect change – tends to be weaker and slower. It is to be expected then that a macro-evolution process would require a much larger period of generational time over which change might occur. It is also possible that while most of the principles that are valid at micro-evolutionary levels still apply, their influence may be different, and there may be other fast-acting mechanisms of greater influence.

For example, it is possible to include the cultural or memetic transfer model, within an ecosystem community, into the model of evolution and scale. In contrast to the inter-generational transfer of genetic traits, organised communities can change rapidly due to the intra-generational transfer of ideas between individuals. This places cultural evolution clearly at the macro-evolution level, though with a rapid rate of influential change.

Finally, the predominant majority of natural systems contain *co-adapted*, *competitive* and *cooperative* species. A model of ecosystem evolution must support *coevolution* through the interaction of species and mechanisms including selection.

Chapter 3

Evolutionary Computation

3.1 Introduction

3.1.1 Objectives

The aim of this chapter is to introduce the field of Evolutionary Computation (EC), and Evolutionary Algorithms (EA) in particular as a biologically inspired search metaphor. It makes specific reference to the ecosystem and evolution concepts presented in Chapter 2, and places emphasis on interactions and topology that are covered in greater detail in Chapter 4.

Biological concepts are first discussed and used as a problem solving model to create a simple evolutionary algorithm including basic operators and important concepts such as *representation*, *fitness* and *convergence*. Included are the ideas of the adaptive and robust qualities of EA search with comparison to conventional optimisation techniques.

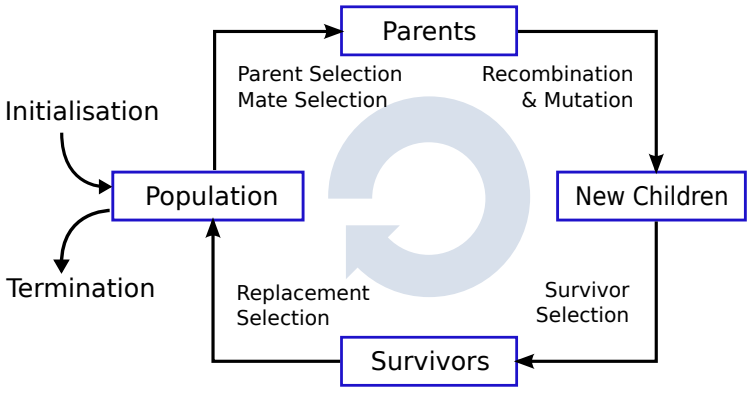


Figure 3.1: The simplified life cycle model used as the metaphor for artificial evolution. This metaphor can be applied to the domain of general problem solving.

A detailed model of general EA components is presented in Section 3.2 which includes simple as well as complex interactions, operations and topology. The components are related to an overall ecosystem model of evolutionary computation. This framework is used as a reference structure for the review of similar work in Chapter 5 and a number of properties and measurements that are appropriate to ecosystem EC.

Several reference EAs are described in Section 3.4 including the canonical Genetic

Algorithm (GA) and Evolutionary Strategies (ES), a more recent real-valued Generalised Generation Gap GA known as G3, and the structured population models of fine-grained Cellular EAs (cEA) and coarse-grained Distributed EA (dEA). The reference EAs are discussed and compared, and used in later investigations.

Appendix B is a collection of benchmark problems that can be used to test the performance of evolutionary algorithms, and are supported by a software framework (ESEC in Appendix E) created for the thesis. Not all of the problems presented are used in the investigations of the thesis, however they have been included to support future work.

Chapter 4 supports the proposed ecosystem EC model by reviewing in detail the area of graph theory and topology, especially topology models that can be used by the ecosystem EC model. Chapter 5 considers questions that the model can help to investigate.

3.1.2 The Simple Evolutionary Algorithm

EA Origins

The idea of using evolution for "automated" problem solving has existed since before the advent of computers [111].

The origins of conceptual models of evolution started in the 1930s with the highly influential work of Sewell Wright. He proposed the visualisation of evolutionary progress as a "fitness landscape" containing peaks (*niches*) of high fitness and clusters (*demes*) of good fitness [373, 374, 375]. Whether this idea is valid or not has attracted much debate, however the notion of a fitness landscape naturally leads to the idea of the search for optima within such landscapes, and hence evolution as an optimisation process. Others strongly contend that evolution is an *adaptation process* in response to *evolutionary pressure*, not an intrinsic optimisation process. This distinction is a significant conceptual difference when comparing EC to classical optimisation methods.

Several reviews of the EC field have noted that perhaps the first proposed application of EC as an algorithmic search tool was by Box [41], who suggested an "evolutionary operation" that could be used to increase productivity in an industrial application. Also in 1957 Alex Fraser presented a "simulation of genetic systems" [121]. Other cited examples of early simulated evolution include the work of Bremermann [42] and Reed et al. [291].

Since the 1950's the field of EC has developed, in many independent forms, and has become a significant and mature research field.

Applied Biological Metaphor

The imitation of biology[1] and the use of biological metaphors have proved to be excellent sources of ideas for many successful computational approaches to problems [274]. This includes not only the metaphor of evolution considered in this chapter (see [274, 136, 168] and many others), but also metaphors such as Artificial Immune systems [104, 35], Swarm Intelligence[102] and the broader perspective of Collective Intelligence paradigms [123], and other evolution variations including Cultural Algorithms [293].

[1] *Biomimetics* is also a popular term used to describe the imitation of biological systems.

Applied biological metaphor of evolution has inspired flexible, efficient and robust problem solving techniques for many domains [210, 275, 128, 98, 77, 123]. Similarly, research of artificial evolution has also been used to replicate properties, and gain understanding, of biological systems [73].

As discussed in Chapter 1 and Chapter 2, the early theories of Darwinian evolution and Mendelian genetics provide us with enough components for artificial models of evolution. Recent advances in both modern evolutionary theory and genetic theory can also be incorporated. The basic components utilised by an artificial Evolutionary Algorithm (EA) are the effect of environmental *pressure* to drive evolution, and genetic qualities or *traits* transferred and modified from parent to children individuals.

With the relatively simple ideas of an *environment*, populated by *individuals* that *compete for survival* and *reproduction*, we are able to develop systems – some with surprisingly complex properties – that can be used to solve problems. This begins by taking the biological life cycle model from Figure 2.3 of Chapter 2 and simplifying it to the essential aspects needed to support an abstracted evolution model that can then be applied to problem solving (Figure 3.1).

Table 3.1 presents the connection between evolutionary ideas (biological or artificial) and the key aspects of general problem solving. It is with these connections that we can use the metaphor of artificial evolution and apply it to problem solving.

Evolution		**Problem Solving**
Environment	$\longleftrightarrow$	Problem domain
Species **population**	$\longleftrightarrow$	Population of candidates
Single individual	$\longleftrightarrow$	Candidate solution
Individual fitness	$\longleftrightarrow$	Quality of candidate
Variable **trait**	$\longleftrightarrow$	Parameter value

Table 3.1: A representation of the connections from evolution concepts to aspects of general problem solving, with bold indicating the three key structural levels identified earlier. Adapted from [98].

Generally, it is only validated concepts of biology, evolution and genetics that are used as metaphor, although it is also quite possible to use ideas that are not valid in biology. For example, in biology incest is usually an undesirable occurrence for healthy genetic diversity, but for EC incest can be a useful feature to enhance problem exploitation. Similarly, in EC we can quite easily allow multi-parent reproduction of more than two parents, resulting in possible benefits for specific problem domains. Such multi-parent reproduction is rare[2] in real-world biology.

This simplified model of biological evolution does not include the ecosystem complexities of organisms and their environment as discussed earlier in Chapter 2. The opportunity

[2]There are many multi-parent "aggregate" or "explosive" breeding examples, where fertilisation occurs outside parent organisms such as the synchronous spawning of coral polyps or the clutch fertilisation of female eggs by multiple males for amphibians.

to include ecosystem concepts, and to what possible advantage, is looked at further in Section 3.2 and later in Chapter 5.

Basic Operation

Let us consider a simple structured artificial Evolutionary Algorithm (EA) using the basic biological life cycle model. As mentioned, EAs can be described as an iterative and stochastic process that operates on a *population* of *individuals* that each represent a candidate *solution* to a *problem* environment. A *fitness function* is used to evaluate the performance of individuals that, in turn, is used to competitively influence the *reproduction* and *replacement* of individuals in the population.

```
t = 0
INITIALISE population with random candidates
EVALUATE the entire population
UNTIL ('Solution Found' or 'Give Up') DO BEGIN
  SELECT fit candidates from population as parents
  CREATE children by VARIATION of parents
  EVALUATE new children
  SELECT fit 'survivor' children
  SELECT candidates from the population to REMOVE
  INSERT children into population
  t = t + 1
END
```

Figure 3.2: Pseudo-code for a basic evolutionary algorithm

Regardless of the detail or complexity of a particular artificial implementation of evolution, it could not be a complete and true description of natural evolutionary systems [113]. Fortunately, the essential elements of population based variation and selection are a useful paradigm in themselves, and the future opportunity for distributed and parallel implementation benefits have yet to be fully realised.

3.1.3 Search and Fitness Landscape

The fitness landscape model contributed by Sewell Wright has already been mentioned, both for its simplicity and the resulting contention of suitability for evolutionary models. For the purpose of initial discussion, we can begin with a simple artificial 1D fitness model that represents such an evolutionary fitness landscape.

Consider the first generation of a population of 10 random individuals in this model. See Figure 3.3 as an example of a fitness landscape (for a single genetic trait) and the population occupancy at three different generational stages of evolutionary progress. Each individual consists of a single gene trait, which is shown as a location (gene value) in the search space axis. Gene values map to a single corresponding fitness value. Although in this case we see the search space values and fitness values belonging to a continuous function, it is significant to realise that the population is only a sample of the domain and is not informed of the continuity. This is in contrast to gradient based search heuristics that rely on and exploit continuity.

The process of evolution, then, is to pressure the development of new individuals (future generations) so that they have gene values that are more likely to occupy fitter locations

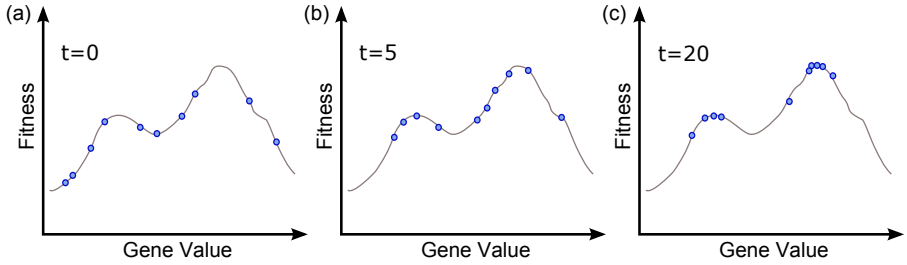


Figure 3.3: A simple 1D fitness landscape with a population of 10 individuals. The higher the fitness value the better. Stages (a) to (c) represent the evolutionary progress of the population at 0, 5 and 20 generations respectively. For this simple model we can consider the gene value the genotype and the fitness value the phenotype.

of the fitness landscape. Ideally individuals will occupy the optimal *niche* location peak of the landscape.

There should be a clear distinction between the search space (gene value) and the fitness landscape (fitness values). Evolutionary pressure influences the gene search space, in the expectation (hope) of finding a good fitness landscape value.

3.1.4 Convergence

There are many factors that can influence the progress of evolution. For example, it can be affected by the population size, the nature of the fitness "landscape", the method of selection of parent individuals, the fertility of parent individuals, the survival rate of individuals, and other similar features.

Fundamentally though, evolution is driven by the pressure of selection. Most commonly this is taken to be the selection of parent individuals used to create new children, though it applies equally to the selection of individuals to be removed also (thus reducing their influence), or other forms of "pressure" that are selectively applied to the algorithmic process.

Fitness pressure is an episodic influence of the selection of individuals for reproduction. The pressure is not deterministic, and so the outcome of adaptation can not be known. Solutions – even if they exist – may not be found.

As elegant as the fitness landscape model is for representing a population of individuals converging to a niche location of optimal high fitness, it is equally valid to conceive a population converging on less-than-ideal locations. The potential convergent outcomes:

- **Optimal convergence:** An individual (or the population) converges to a single region of search space – the best possible.
- **Sub-optimal convergence:** The population converges to a single suboptimal region of search space.
- **Concurrent optima convergence:** The population is divided and concurrently converges to two or more separate search space locations. Essentially this creates different subpopulation "species" within each niche area of the search space.

It is possible that the population will not converge to a suitable search space region. The following potential influences should be considered:

- **Genetic drift** describes the behaviour of a finite population undergoing stochastic selection. Quite simply, genetic diversity is lost due to insufficient sampling (sampling error).
- **Lack of time** in a practical sense to allow evolution to occur. This is a very real concern for high dimensional and interdependent problem domains.
- **Disruption**, such as external influences that result in a radical change of the fitness landscape, or a significant extermination of individuals, and thus removal of genetic material from the population. A simple random influence or noisy problem domain can be too disruptive for some search techniques.
- **Resolution**, in that the search space does not contain the details needed to represent appropriate solutions of the problem domain. A high resolution solution might be desirable, although high computational cost can be avoided if a simple low-resolution model is adequate.

As already described, the driving influence of an evolutionary process is the pressure that can be applied. If pressure influences are contradictory they may limit or cancel each other, or cause a disruptive effect such that the net effective pressure is neutral and the population genetically drifts.

It is possible that two or more similarly fit, but genetically different, search space locations exist. New individuals are then clustered to one of the alternative locations. However, it is unlikely for such a "dynamic equilibrium" to be sustained over generational time due to the nature of stochastic sampling allowing a single solution type to dominate. It is even less likely to occur at an overall population level if there is a high degree of connectivity.

3.1.5 EAs as Robust, Adaptive Search

Adaptation, Optimisation and Robustness

The result of evolution (natural or artificial) is *adaptation*. Search using an evolutionary algorithm is intrinsically a heuristic that *iteratively*, *non-deterministically* and *stochastically* tries to adapt solutions. Optimisation problems have been a classic and also somewhat troublesome application of EAs. But as Eiben and Smith point out, linking evolution to optimisation problems "is as straightforward as misleading"[98, p5].

The misunderstanding of evolution and optimisation prompted many researchers to directly stress that EAs are not, in themselves, "function optimisers" [364, 98, 77]. Rather, it is better to describe EAs as *robust adaptive systems* which can be applied to optimisation as an "adaptation process" (that is *search*) [76, 157].

Gradient unidirectional hill climbing heuristics are commonly and appropriately used for continuous function optimisation. However they may also be easily trapped (or deceived) by local optima in the search space [2]. The performance of hill climbing techniques can be used as a baseline performance measure against specific EA implementations [178].

Evolution is a stochastic process that has some potential advantages over gradient based techniques. EAs have been successfully applied to many non-linear and stochastic optimisation problems. Although EAs can rarely match the convergence speed of gradient techniques on smooth problem spaces, EAs are less likely to be "caught" or "deceived" by a local optimum on rugged problem spaces. For this reason EAs are generally considered "robust".

It is interesting that the original aim of John Holland, one of the pioneers of the EA field, was to develop an adaptive and robust search system rather than to search for singular optimal solutions [166]. Robustness is a desirable quality, though it may be at the expense of search convergence speed. A cautious or less greedy search may take longer to reach a desired search outcome.

There is another practical quality of EAs: they can be readily applied to problem domains where there is no single optimum – a situation that can be quite difficult for other strongly directed or greedy techniques.

Search, Scores, Fitness, Quality and Pressure

As evolutionary *adaptation* can be considered a *search process*, we consider each possible solution to be a point in a *search space*. Some points may not be viable, depending on the problem domain and representation schemes used [184]. The successful *trajectory* or *projection* of the evolutionary progress from the initial population points to good solutions is dependent on the *suitability* and *correctness* of fitness values assigned to individuals. Further, progress is effected by the influence (utility) fitness has on evolutionary operators (such as selection and replacement). There is a distinction between an individual's *raw score* or value in the search space, and how this score value relates to a *fitness* value. In some domains, the score can be used directly as the fitness value, but in others a conversion is required. Conversion allows for normalisation or scaling of fitness values, and inversion so that minimisation or maximisation search domains can both be mapped to a single maximisation fitness domain.

Raw score conversion to fitness is usually the role of the "selection" operator, and there are many variations. However most selection operators belong to either proportional or rank based techniques.

The heuristic nature of an EA means that finding optimal solutions cannot be guaranteed, and so solutions may be of less than optimal quality. The definition of *solution quality* is an important problem specific (and commonly user specific) constraint that should be considered when EA techniques are applied to new problem domains.

Evolutionary pressure is a useful term for identifying and describing how the process of evolution is directed or "pushed" over each generation. Primarily this pressure is attributed to the role of fitness, and because both the determination of fitness, and its influence on

evolutionary operators, can vary in application, determining and comparing evolutionary pressure is a useful, but difficult to define, quality.

A search with strong evolutionary pressure will converge quickly. Evolutionary pressure must accommodate all desirable solution qualities, otherwise those qualities are not preserved in the population. Weak evolutionary pressure can allow the persistence of undesirable qualities, but this may be useful (even essential) to the diversification of the population. We can describe evolutionary pressure from both a macro- and micro-level, where the pressure is specific only to individuals, while a micro-level indicates pressure that is sensitive to the component parts or traits within individuals. Pressure can be adjusted at both a specificity level, and at a magnitudinal level. Determining the right nature and scale of evolutionary pressure to apply can seem as much an art as a science, and this is a central question in applying EC to search in specific domains.

Application Examples

The applications of EC are many and diverse, covering fields of business, engineering and science. Examples (some of which overlap) include control systems [173], data mining [285], fault-tolerant systems [340, 341], game playing [15], machine learning [136], ordering problems [305], scheduling [377], set covering and partitioning [218], physical designs [33] and strategy acquisition [150].

Almost all reviews of the EC fields present successful applications of EA [17, 237, 98, 77]. Several classic application examples are included later in Section 3.3. Based on the results of the EC field, there is little doubt that EAs – applied to many different fields with a variety of implementations – are a useful paradigm for solving some problems classes.

3.1.6 EAs and Conventional Optimisation

It has been shown that EAs are good for specific problem domains, and this result agrees with general algorithmic information theory, and specifically the "No Free Lunch" theorem [370]. For example, an established view supported by work such as presented by Miettinen et al. [231], is that EAs can provide good performance when:

- there are *no known solutions*;
- the *number of search parameters is very high*; or,
- the *number of potential solutions is high.*

The points presented in Table 3.2 are based on the views expressed by Engelbrecht [102] and makes a useful presentation of the perceived differences between more conventional "classical" optimisation techniques, and the role that EAs can play as a search (adaptation) technique.

3.1.7 Further Resources

There are many publications and resources for the continually growing field, both in its breadth and depth, of evolutionary computation (including ALife). See Bäck, Fogel and

	Classic Optimisation	Evolutionary Search
Domain	Best suited to linear, quadratic, strongly convex, unimodal and other similar specialised domains.	Best suited to discontinuous, non-differentiable, multimodal and noisy domains.
Process	Deterministic and sequential search steps, usually starting from a single location in search space.	Stochastic (probabilistic) adaptation steps, implicitly parallelised by the population. A set diverse initial starting locations samples the domain.
Guidance	Derivative based (usually first or second order) to guide search steps.	Individual fitness is used as a search pressure. This pressure is typically applied via selection, but reproduction or replacement is also possible.

Table 3.2: A comparison of Classic Optimisation methods and EA Search (adaptation), based on the qualities of best suited problem *domain*s, the central *process* mechanisms of each method, and nature of search *guidance*.

Michalewicz [17, 18, 19], Davis [70], Mitchell [237], Man [223], Eiben and Smith [98], Fogel [113] and De Jong [77] for excellent detailed introductions, history and coverage.

The International Society for Genetic and Evolutionary Computation (ISGEC)[3], supports communication and academic forums for researchers (including the annual GECCO and biannual FOGA conferences).

Specific journals of note include *IEEE Transactions on Evolutionary Computation*[4], *Evolutionary Computation*[5], and *Genetic Programming and Evolvable Machines*[6] which is available as part of the membership of ISGEC.

There are four main EC conferences. The first two have resulted from the merging of prior conference series, and they are the *Genetic and Evolutionary Computation COnference* (GECCO) and the *Congress on Evolutionary Computation* (CEC), both of which are held on an annual basis. The European based biannual *Parallel Problem Solving from Nature* (PSNN) was specifically established in 1990 to encompass many streams, including EC. The biannual *Foundations of Genetic Algorithms* conferences (FOGA) support a specific focus on theoretical aspects of EC.

Heitkoetter and Beasley compiled a popular Frequently Asked Questions (and answers) list known as the "Hitch Hiker's Guide to Evolutionary Computation" and made it available online in several locations[7].

[3] See the ISGEC website at http://www.isgec.org/
[4] Published by IEEE Computational Intelligence Society. See http://ieee-cis.org/pubs/tec
[5] Published by MIT Press. See http://mitpress.mit.edu/
[6] Published by Kluwer Academic Publishers. See http://www.kluweronline.com/issn/1389-2576
[7] A commonly indexed location is at http://www.faqs.org/faqs/ai-faq/genetic/

3.2 An Ecosystem Model for EA

3.2.1 Introduction

This section presents an ecosystem model for evolutionary algorithms. The model includes ecosystem concepts of topology – inherent in natural environments, population structures, communities and species – to form a coherent model of evolutionary algorithm components.

There are many generalised EC frameworks and comparison models presented in literature. The initial influences for this framework include the classic "Handbook of Evolutionary Computation" [17], Bäck's reviews (such as a chapter of [274]), Bentley's introduction chapter of [33], and more recently books by Eiben and Smith [98], a "unified" approach by De Jong [77] and a philosophical view by Fogel [113].

Notable differences between the model presented here and other general models typically relate to the number or type of operator components, the order of operations, and understandable biases from GA or ES perspectives.

Components are introduced in Section 3.2.2 and described from Section 3.2.3 to Section 3.2.9. The influence of components is discussed.

3.2.2 Components

The key processes of the neo-Darwinian paradigm have been identified as **reproduction**, **variation**, **competition** and **selection** [228]. These can be related to the structural components of *ecosystems* identified earlier: **environment**, **communities** of *species*, **populations**, individual **organisms**, and heritable **traits**.

All of these ideas and components can be incorporated into an evolutionary search algorithm model, along with other algorithm requirements (that are often implicit in other frameworks).

- **Search domain** clearly defined (environment) including the number of variables and their constraints.
- **Candidate representation** (encoding) of traits appropriate for the domain variables.
- **Population structure** (representation) to define the interaction of individuals.
- **Evaluation** function(s) (fitness) for assessing candidate value with respect to a particular context.
- **Selection** processes for several different contexts including the selection of parent, parent mates, child survival, replacement and migration.
- **Variation** operators (for reproduction), such as crossover and mutation, representation specific.
- **Migration** rules and policies (between subpopulations, also using selection).
- **Initialisation** process.

- **Termination** conditions.

This collection of ecosystem EA components is now organised into groups and considered in more detail.

3.2.3 Representation

Search Domain

A search domain can be defined by an objective function f that is affected by a set of variables (unknowns), say a vector $\vec{x}$ of x_1 to x_n values. For optimisation search, the objective is the maximise or minimise $f(\vec{x})$ such that $\vec{x}^*$ represents an optimal location of the search problem domain.

For constraint satisfaction problems (CSP) the objective is to find an optimal set of values $\vec{x}$ that also satisfy a list of defined constraints. Constraints can apply to any domain, and nearly all problem domains have boundary conditions for variables. The set of constrained feasible solutions can be described as $\mathcal{F}$, and denoted as a subset of the entire search space $\mathcal{F} \subseteq \mathcal{S}$.

As introduced in Section 1.7, Engelbrecht [102] has suggested a useful list of search domain characteristics including the:

- *number of variables* in $\vec{x}$ for f;
- *type of variables* within $\vec{x}$;
- *degree of nonlinearity* of the objective function f;
- *set of constraints* (either on each variable in $\vec{x}$ or the output f or both);
- *number of optima* (the modality) within the space of f; and,
- *number of optimisation criteria* such that multiple objectives are subobjectives of the overall search objective.

When considering how a search method will perform, it is important to realise the distinction between the actual search domain used, based on internal representation, and the real-world or abstracted problem domain. This is particularly the case for EC based search, where the representation of the problem may be an encoded "genetic" representation that may potentially add or detract to search performance. EA search is always a search of the encoded space, not the problem space, even if the two spaces are equivalent.

Individuals and Traits

Choosing an appropriate representation for individuals is a crucial part of applying an EA, and the choice impacts both the efficiency and the complexity of the algorithm.

Trait representation can be as simple as a collection of binary values, integer or real number values, character strings, sets (sequences) and sub-unit variance structures [17].

While most EAs use a vector representation of bits or floating point values, genetic programming is a notable exception in that it explicitly uses a tree structure to represent programs.

Genotype to phenotype mapping allows complex relationships. Specifically, *pleiotropy* is the influence of genes on multiple phenotype traits, and *polygeny* where several genes combine or interact to form the final phenotype trait.

An interesting and well known variation for representing individuals is the messy GA (mGA) structure in which genes represent both position and value of a solution [140]. This allows both the gene value and its 'location of affect' to be explored in the search, but adds additional computational cost.

Similarly, Copland presented a "genetic encoding scheme" based on a model inspired by the storage of genetic information in DNA [57]. In this model traits are not explicitly represented in a single location, but instead encoded across multiple locations of a common genetic string, and a single location can contribute to multiple traits.

An individual does not need to be homogeneous and can include a number of different trait types and sizes. For example, a neural network configuration could be represented by a binary string, where 1's represent connections or nodes, and the weight values as real values. Of course, mixing trait types in recombination (crossover) should probably be avoided.[8]

As another example of non-trivial genome trait mapping, in [371] a hybrid approach was presented using multiploid individuals (genomes containing two or more chromosomes) using neural networks as a trait expression mechanism, and applied to classification tasks. The hybrid genome included both values for network node activation, network weight values, and bias weight values. The approach was applied to the development of data classification systems, where an individuals fitness is a function of its ability to correctly classify a number of data points.

The use of possibly redundant genotype information can help to avoid premature population convergence, and the complex abilities of individuals may enable them to be applied to complex domains. There are evolutionary costs in that the additional trait complexity requires a longer amount of time to evolve successful solutions. Local optimisation of traits (in this case, by training the neural network using a local back-propagation algorithm) can help reduce overall search time.[9]

In the general case any increase in individual representation complexity – even if highly beneficial – will require increased computational cost in other components, such as specialised recombination, mutation and fitness allocation. Complex representation may be worthwhile if selection is able to act effectively on the components, but it is also desirable to avoid obfuscation and needless complexity.

[8] Mixing between trait types could be used as a means of introducing new variation, although without clear justification.

[9] Local search creates a Lamarckian or Baldwinian scheme, depending on whether local changes are written back into the genome or not (respectively).

Populations

A population is a multiset[10] collection of individuals. For most EAs it is the contents of the population that change in response to the evolutionary process, while specific individuals are static.

The structure of the population defines the interaction of individuals, and hence influences other components, namely evaluation and selection. Where a simple *panmictic* topology[11] is used each individual interacts with, and is influenced by, all other individuals of the population. In a model composed of subpopulations, evaluation and selection are localised within each sub-group, and there is also a need for a migration policy that defines movement of individuals between subpopulations. Fine-grained cellular topology models not only localise operators, but also overlap neighbourhoods of interaction [224, 65, 199, 98].

Given that evolution, and particularly recombination, acts upon the available variation that can be selected, the size of the initial population has a direct relationship to the amount of genetic material initially available [51]. Because of this, many techniques have been proposed by researchers to estimate appropriate initial population size based on the structure of individuals (i.e. number of genes or schema 'building blocks') and the problem domain requirements [74, 148, 137].

Strategies to adapt population sizing during evolution have also been used [14]. There are interesting results in allowing population sizes to adapt and change based on ecological principles, limited resources and competition.

If mutation is the primary variation operator, then the dependency on the population to supply variation is reduced, and the minimum population size required for an EA to be effective is decreased [141, 339].

Using observation from natural populations, we know that a specific environment can support a finite number of individuals, and that concepts such as the 'occupancy rate' in natural systems will vary due to various environmental influences. Such influences might include food, water, climate, predators, geographic constraints and other more severe environment altering events such as volcanoes, bush fires and floods. There is also strong evidence that such influences alter evolutionary progress. Some of these effects are investigated in more detail in later chapters.

Distributed and parallel EAs are examples of the use of structured subpopulations (demes) (Figure 3.4). There are several useful aspects of structured populations, such as the potential for reduced EA "wall-time"[12] and elegant application to practical hardware constraints. Multiple population implementations also require the development of methods and rules to govern the migration and transfer of individuals between subpopulations [52]. The population size estimation techniques mentioned for single populations have also been extended to the number and size of subpopulations (demes) required in distributed and parallel EAs [51].

[10] Defined as a generalised mathematical set that, unlike a set, can contain multiple instances of the same element (individual).

[11] The term *panmictic population* comes from the field of a population genetics, and describes an environment where all individuals are potential mating partners.

[12] This expression simply refers to the amount of time required – as measured by a clock on a "wall".

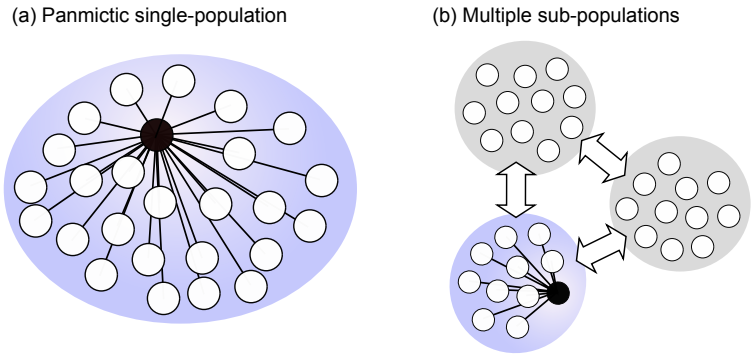


Figure 3.4: Two different population models. In (a) a simple panmictic population model where a single individual can interact (shown as a star topology) with all other individuals of the population. In (b) a more complex sub-population model where interaction is limited to interactions with other individuals in the same subpopulation (or "deme") with defined rules for migration between subpopulations.

Community

A community can be used to describe several views of a population and the interaction between individuals within an ecosystem evolution model:

- Population organisation for a single species including, for example, sub-groups for juveniles, adults or gender.
- Interacting subpopulations in a multi-deme structured, cellular or distributed EA model.
- Multiple species interacting through competition or cooperation, and their possible co-adaptation.
- Environment-based or interaction-based (fitness) niches.
- Exchange of ideas or beliefs (as in cultural algorithms).

As discussed earlier in Section 2.3.4 and Section 2.3.5 community includes structural ideas such as the number of species that a community includes, how species interact and the density of interactions. A community can also be a frame of reference for dynamic properties such as the stability, resilience and resistance of a community to changes. Importantly though, community level interaction has proved to be a useful macro-evolutionary model for adaptive search [278, 281].

3.2.4 Evaluation

The evaluation of an individual is performed within a specific context: the current environment, defined by the population structure and the interactions with other individuals of same or other species. The population context may be an entire panmictic pool, or localised sub-population demes, or fine-grained neighbourhoods. Evaluation may involve a community of species competing or cooperating, possibly in an increasing system of

complexity and competitive development.[13] To get an idea of the possible complexity consider Figure 3.5 which illustrates epistatic linkages between three species.

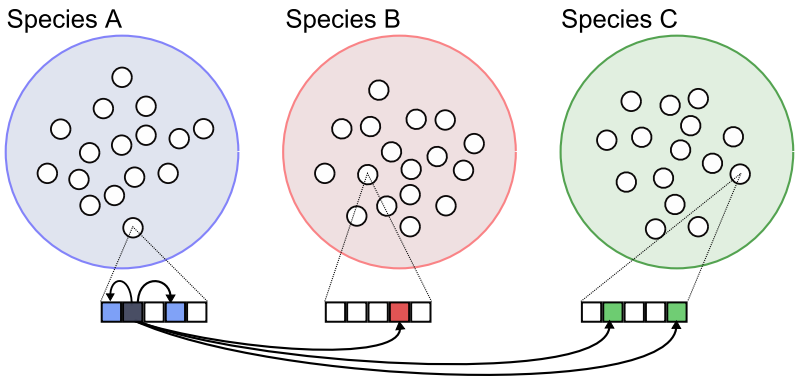


Figure 3.5: A representation of the epistatic linkages for a single gene in the NKC model [187, 322]. There are $s = 3$ species, each individual has a genome length $n = 5$, with $k = 2$ intra-genome linkages and $c = 3$ interspecies gene links. The linkages of the second gene in the first individual are shown.

The process of evaluation begins as traits are converted from an internal genotype representation to the external environment (via expression or mapping) as the phenotype.

For an EA, the phenotype is evaluated by a 'cost' function, also known as the objective function, which may increase or decrease depending on how good the individual is in the current domain. This relates to maximising or minimising problem domains respectively. The assigned cost value is then used as part of an indication of 'fitness', however the exact contribution depends on the selection operator; fitness is a context specific utility concept. Fitness, from the ecosystem perspective, is an organism's ability to survive and its ability to reproduce – or at least have representatives of the same genetic traits reproduce.

A fitness function f can be expressed as a mapping of an individual's n genes x (of data type X) to a real value R:

$$f : X^{n_x} \rightarrow R \tag{3.1}$$

As a cost value may not always be easily or directly used as a fitness value, there are many different techniques that can be used to convert or map the cost of an individual to a 'fitness' value. This process is often integrated as part of the selection component, where cost is immediately used or converted into the appropriate fitness value. For this reason, an evaluation function is often just referred to as the fitness function.

Specialised methods are required to deal with problem domains with multiple objectives (criteria), in particularly those with conflicting or *non-commensurate* objectives [118, 119]. Examples of such methods include *weighted-sum*, *minimax*, *target vector*, *median-rank* and *Pareto ranking* approaches [120].

For the weighted-sum approach, individual objective results are weighted and summed to provide a single cost value for assignment. The difficulty lies in the setting or adapting of

[13] See the Red Queen Hypothesis discussed later in Section 5.4.4.

suitable weight values, especially without *a prior* knowledge of either the problem domain or the genotype search space [120].

Evaluation of individuals is often the most computationally expensive process of an EA. The computation cost of evaluation can be separated from the computation cost of fitness value calculation if treated as separate components. This is particularly relevant when dealing with evolution occurring upon network structures, as the raw evaluation cost is one distinct operation, while the relative fitness of each individual may be specific to a localised neighbourhood, such as a sub-graph of an entire population.

Because evaluation can be expensive (and hence a need to avoid an exhaustive search), it is not surprising that researchers have investigated ways to reduce or approximate evaluation (and fitness) calculations. Such modified EAs are frequently known as "fast" EAs [108]. Similarly, the distribution of evaluation processing is an effective method for improving the 'wall time' performance of EAs, particularly when the evaluation time is long compared with the trade-off of communication overheads that distributed processing requires.

3.2.5 Selection

Relative Context

Selection applies to many different stages of an organism's life-cycle, not a single event, and is based on the current environment and context rather than a static reference. Within many EA examples selection is often restricted to parent selection. However, consider that a process of selection can occur when determining which candidates to:

- **reproduce** (to be a parent of offspring),
- **mate** with for reproduction (as part of the previous concept),
- **survive** childhood (from the offspring pool),
- **remove** from the adult population (life stage or death), or
- **migrate** to another population (sub-group or environment).

Evaluation in each case may be with respect to different criteria, in which case a different "fitness function" can be used (based on the relevant traits or environment conditions). For a simple EA, an absolute fitness value may suffice. For more complex models of interaction, as in structured population modes or coadaptation, fitness is clearly a relative (context based) measure and hence so is any selection based on it.

A short summary of the most common selection operators is presented in the following sections. Selection operators can be characterised by the degree of selection pressure they create. One common measure is the takeover-time for a single trait to propagate to the entire population under selection alone [138]. De Jong orders selection from weakest to strongest [77]:

- **Uniform Random** (stochastic).
- **Fitness Proportional**.
- **Linear Ranking** and **Binary Tournament** ($k = 2$).

- **Nonlinear Ranking** and $k > 2$ **Tournaments**.
- **Truncation** (deterministic).

Uniform Random Selection

The simplest and weakest of selection techniques, individuals are selected randomly and uniformly without regard to traits or fitness. Random selection is often used as a neutral stage in conjunction with a stronger selection method in another stage. Genetic drift may occur under uniform stochastic selection. Topology might still limit the available pool of individuals to select from, as in distributed or cellular EA population structures.

Fitness Proportional Selection

The probability p_i that individual i is selected from the population (of size n) is proportional to the individual's absolute fitness score f_i with respect to the rest of the population's absolute fitness scores [166].

$$p_i = \frac{f}{\sum_{j=1}^{n} f_j} \tag{3.2}$$

Being one of the first proportional selection schemes it has been extensively studied [98]. One issue is that an outstanding individual, whose raw fitness score is much higher than other individuals, will dominate selection and saturate the population. Similarly, when all individuals' fitness values are relatively similar there is almost no selection pressure; essentially uniform selection. Also, transposition (offset) of relatively similar fitness windows results in different relative sampling proportions.

The last two points can be reduced by the use of a sampling window, such as subtracting all fitness values by the least fit member (or its running average).

Roulette Wheel selection is another response to these problems. Essentially, fitness values are normalised (by dividing all fitness values by the maximum), and individuals are then uniformly sampled using a cumulative fitness proportional fitness allocation. This can be described as follows:

1. Determine the fitness f_i for each individual $i = 1 \ldots n$.
2. Calculate the probability "slot" size p_i (See (3.2)) for each individual.
3. Create a list $[q_1, q_2, \ldots, q_n]$ of the cumulative selection probability where $q_i = \sum_{j=1}^{i} p_j$. Note that $q_{(n)} = 1.0$.
4. Generate a uniform random number $r \in (0, 1]$.
5. Starting at $i = 1$ iterate until $q_i > r$.
6. Return individual x_i.

Repeat steps 4-6 as needed to create a sample pool.

Although elegantly simple, Roulette Wheel selection still suffers from poor sampling behaviour. If multiple samples are needed, then a "multi-armed" approach of evenly spaced samples, known as Stochastic Universal Sampling (SUS), is preferred [22].

Tournament Selection

Tournament selection is perhaps one of the most commonly used schemes. It samples a uniform random group of k individuals from the entire population n (where $k < n$). The best of the k tournament group "wins" and is used. Binary tournament selection is simply the case where $k = 2$.

Tournament selection does not require global knowledge of the best individual (and so avoids sorting). As tournament comparisons are relative, selection is similar to rank-based schemes rather than those based on absolute fitness. Tournament sample groups can be with or without replacement.

As the size of the tournament group increases, the greater the chance it will contain good individuals, and so selection pressure increases. If tournament size is too small, there is a greater chance that poor individuals will be selected. Varying k is thus an easy way to adjust selection pressure.

Selection pressure can also be weakened by making tournament outcomes probable rather than deterministic. In this way, there is a random possibility that the weakest, or random, individual of the tournament will be selected.

Rank-Based Selection

Rank-based selection orders the group of individuals by fitness. It is then the rank-order r_i that is used to allocate selection probabilities p_i to each individual i. The mapping of an individual's rank to selection probabilities is arbitrary and can be done in many ways. The advantage over fitness proportional selection is that the best individual will not dominate the sample [22, 324].

A simple linear rank-probability mapping results in a non-deterministic linear sample of a similar strength to binary tournament selection. However, a non-linear exponential ranking scheme can increase selection pressure and is commonly used [364] such that:

$$p_i = \frac{1 - e^{r_i}}{c} \tag{3.3}$$

where c is a normalisation factor selected so that the sum of the probabilities is 1 (unity). In this case, ranking is sorted by decreasing order of fitness. The rank-probability mapping scheme can be changed to vary selection pressure.

Truncation Selection

Truncation selection is a form of deterministic rank-based selection, and one of the strongest schemes. It is inherently elitist.

Essentially, only a set number or percentage of the most fit individuals from a group are selected. Of the selected individuals all are equally likely to be used. The greater the ratio of group size to the selected sub-group, the stronger the selection pressure.

In Evolutionary Strategies (ES) (see Section 3.3.2) truncation selection is used to reduce the child pool of μ offspring, or the combined parent λ and child pool $(\lambda + \mu)$,

down to the λ parent pool size. These approaches are known as (λ, μ)-ES and $(\lambda + \mu)$-ES respectively.

Note that such a strong selection scheme works with ES because of the large amount of mutation-based (Gaussian) variation introduced during reproduction.

In a similar way, replacement selection of individuals is often done based on fitness or age, so that the worst or oldest of the population are replaced by new individuals. This is still truncation selection but with a different method of rank order determination.

Elitism

Closely related to selection schemes is the policy of elitism, which ensures that the best individuals are always retained or copied into the population. The advantage is that the best candidate(s) are not lost, however it can also increase the rate of convergence and lower the diversity of the population.

Hall of Fame

A hall of fame is an external record of the best individuals in the entire generational history of the population. The list can be used as the parent pool for reproduction operations, as part of a migration/injection model, or simply as a record keeping exercise. If the hall of fame list does not influence reproduction, it simply provide the lossless record of elite individuals without the skewed selection bias elitism would normally have.

Niches and Crowding

There have been many attempts to utilise selection to create population niches and crowds in order to encourage both exploration and exploitation in search landscapes [354, 233, 155, 54, 229].

For example, biasing selection probability of parents – such that individuals from similar niche areas are selected – can encourage exploitation. Biasing selection so that individuals from different niches are selected may encourage exploration. Similarly, replacement selection can be biased toward children replacing similar adult individuals via fitness proportional, tournament or probabilistic processes.

3.2.6 Variation

Recombination and Mutation

The job of variation is to create new individuals that are variations of past individuals. The two dominant variation operators are recombination and mutation, although some view recombination as simply a macro-level mutation operator.[14]

The specific process of variation depends strongly on the representation and expression of individuals. For example, sequence sets or tree representations benefit from specialised recombination and mutation operators that preserve the feasibility (validity) of candidates.

[14] A reason why this ecosystem model favours the component name of "variation" rather than specifically recombination, crossover or mutation.

Binary Crossover

Within the Genetic Algorithm (GA) field (see Section 3.3.4 for more detail) the crossover operator has been the most common form of recombination, especially for simple binary string representations. The three most popular crossover types are one-point, two-point and uniform. In each type, two parents are selected and recombined to create two children. See Figure 3.6 for examples of each crossover type.

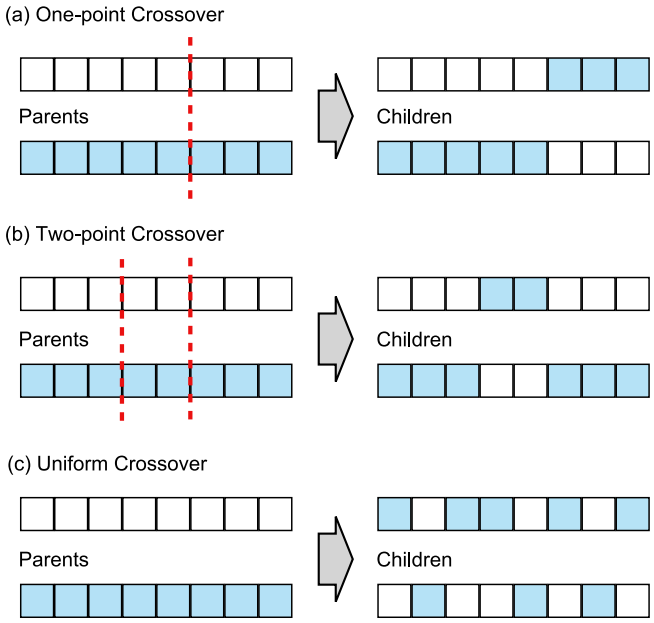


Figure 3.6: Examples of standard (a) one-point, (b) two-point and (c) uniform crossover between two parent individuals used to created two child individuals. The dotted line represents randomly selected crossover points in (a) and (b). Shaded cells are used to indicated the transfer of genetic traits to children. The actual type of genes and values are not shown, as such details are independent of these crossover methods.

Arithmetic Crossover

For real-valued vector representations, arithmetic cross-over can be used in a similar concept to binary crossover. Quite simply, a uniform random mid-point between each parent's values is selected (for each position of the parent vectors). This means that children's values are within the value range of the parent's respective values, effectively sub-sampling the values. There are many other forms of arithmetic crossover.

Later in Section 3.4.4 another real-valued crossover is presented, called Parent Centric Crossover (PCX), which uses a group of contributing parents. The overall effect is that PCX samples a distribution of values around each parent in direct proportion to the separation distance of the parents. This has been a highly effective variation operator for real-valued function optimisation problems [81].

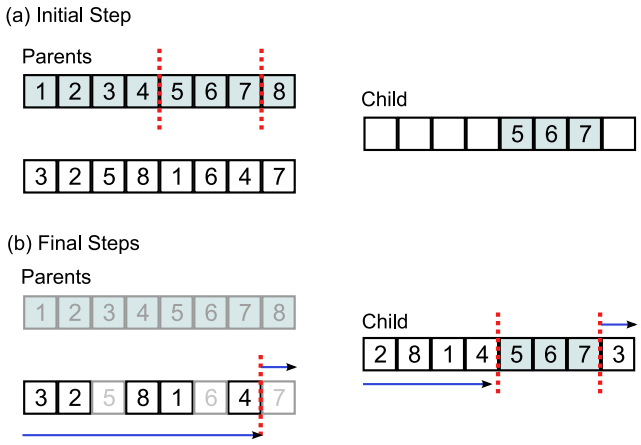


Figure 3.7: Order-based crossover for permutation individuals

Order-Based Crossover

In Figure 3.7 an example of *order-based crossover* [147, 70, 98] is shown. In this case, a section of the first parent is randomly selected and copied to the single child. Then the remaining genetic material from the second parent, that has not been copied, is transferred to the child in an order preserving manner, cycling from the end to the start until all components of the permutation have been represented in the child. The underlying idea is to preserve as much of the relative order information as possible during crossover.

Bitstring Mutation

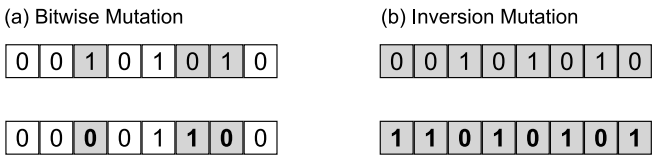


Figure 3.8: Simple mutation examples. In (a) three bits are randomly selected and mutated. In (b) the example shows the entire individual inverted.

In Figure 3.8 are two forms of binary string mutation. Bit-toggling or "bitwise" mutation is simply the toggling of bit values. Similarly, inversion mutation is a macro-level change by toggling the state of every bit, so ensuring that any missing bit variations have a chance of being reintroduced.

Sequence Permutation

When more domain specific representations are used, such as sequence permutation, there is an opportunity to introduce mutational variation that preserves the permutation set requirements. For example, in Figure 3.9 four different valid permutation mutations are shown. Each operation alters the order and ensure the result is a valid permutation.

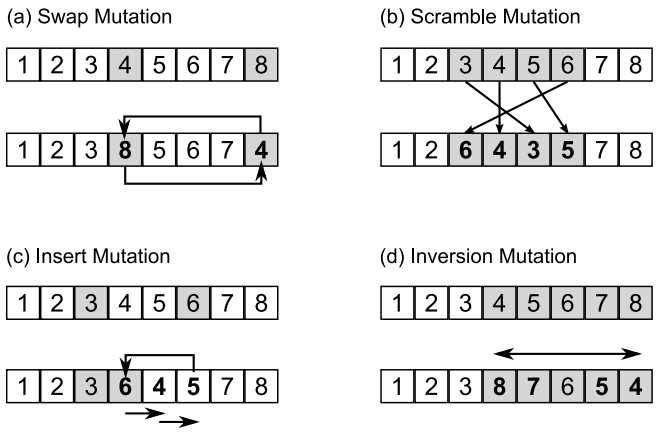


Figure 3.9: Sequence permutation mutation examples. Grey cells indicate selected genes. Numbers in bold indicate altered values after mutation has occurred. In (a) a simple swap method is used, (b) uses two selected points and scrambles genes between the two points (inclusive), while (c) is a displacement or insert based operation that effectively moves and shuffles genes. (d) is an inversion style mutation that reverses all the gene positions between two selected points. Note that in all cases the sequence result is valid and there is no need for a repair operation.

Specialised Variation

As in the permutation examples, other domain specific representations can benefit from specialised variation operators, either recombination or mutation based. For example, program trees used in Genetic Programming (see Section 3.3.5) benefit greatly from structure aware recombination and mutation operators.

It is also possible to repair invalid candidates or to simply penalise fitness based on viability, which can help in that the search space is more relaxed than the constraints of the problem domain.

3.2.7 Migration

Migration is the transfer of individuals (or 'clones' in some cases) between populations, typically in parallel or distributed EA implementations. Sometimes these subpopulations are called *islands* or *demes*, while the process of transferring individuals is known as *displacement* or *injection*.

In a general sense displacement could also relate to movement of any new or old individual – including replacement – however in this work the use of this term is limited to subpopulation based concepts.

There is evidence to suggest that migration models, in particular distributed and parallel population models, have an advantage other than the practical distribution of computation resources. For example, isolated subpopulations provide a supportive niche where specialisation and exploitation of the search space is viable, while occasional integration (migration) still enables overall search space exploration. Such advantages appear to be representation and problem specific [51, 52].

3.2.8 Initialisation

Individuals and Traits

Initialisation involves the creation of a new population of individuals and the initial setting of trait values. For some parameters there are known *a priori* values, while for others complex interdependencies mean the best result is often far from a simple random choice. If regions of the search space are not represented in the initial distribution of trait values, recombination without mutation will struggle to explore the missing sample regions.

Trait values are typically set according to a random distribution of the allowed domain of values. Specific random distributions, such as Gaussian, Cauchy or Linear may be used, although this implies a knowledge of the problem domain and is a form of directing the initial search. Although useful, it should be considered carefully in comparing techniques.

Some researches question whether the computation to effectively incorporate knowledge into the initialisation is worth-while compared with the progress of normal exploration [98]. Practically, seeding a population with useful information or components is usually not hard. If an algorithm is robust enough to also ignore or overcome poor starting conditions, there are few detractors and so seeding may be justifiable.

Individual representations that also encode variation parameters, as used in evolutionary strategies, must also be initialised suitably for the application.

The process of initialisation can be analysed from the perspective of supplying suitable 'building block' (BB) material (see [136]) for the operators of selection and reproduction (crossover) to work with. In this case, initialisation provides the means of 'sampling' the search space, and so the magnitude and distribution of trait values can be crucial to EA performance. The BB perspective does not apply well to EAs that do not rely on the crossover operator, such as Evolutionary Strategies (ES) and Evolutionary Programming (EP) techniques (discussed later in Section 3.3.2 and Section 3.3.3) respectively, but the nature of initialisation is still very important and influential to performance.

Populations and Communities

An EA implementation will typically perform initialisation by allocating memory space for an entire population, and then create the initial individuals with their specific distribution of values. Given that some population structures use a dynamic population size, determining the initial size of the population might also be a concern of the initialisation component.

Similarly, if an EA includes a community model of subpopulations or interacting species, these also need to be initialised, organised and connected.

Not all population models use maximal occupation; it is possible to use models of available space within an environment habitat, and as evolution progresses locations become occupied, and relative fitness competition and interesting extinction events can be used to stimulate adaptation.

3.2.9 Termination

There is no intrinsic way for EAs to know when to stop. There is certainly no biological equivalent except extinction. Practically though, EA implementations utilise several termination criteria. The main examples of EA termination conditions are based on:

- a maximum number of generations;
- a maximum number of children produced;
- a maximum number of full individual evaluations;
- a limited amount of time;
- a stagnation of the current solution quality (relative or absolute if available); or,
- a loss of the measured population diversity.

The issue of when to stop an EA is usually not specific to a particular 'dialect' but rather a mixed selection of the conditions listed that are practical for the current application or situation.

3.2.10 Components and Influence

Order and Dependency

Adaptation within an evolutionary algorithm is driven by the preferential treatment of "good" individuals and/or the negative attention given to "poor" individuals. This positive or negative *selection* pressure is applied through *fitness* functions. Some EA implementations avoid the use of both types of selection as their combined influence can be excessive. However, without sufficient trait variation within a population, or injected into it, selection has no gradient to act upon. Variation is critical to selection.

One way of expressing the combined influence of selection $s(.)$ and variation $v(.)$ operators on a population of candidate individuals $\mathbf{x}$ from one generation t to the next $(t+1)$ is with the following notation:

$$\mathbf{x}[t+1] = s(v(\mathbf{x}[t])) \tag{3.4}$$

In this way, we can see the direct influence of variation to selection, and variation itself is based on the current candidate population. This type of representation leads to a Markov chain view of evolutionary algorithms, and the probabilistic description of operator influence on fitness distributions [113].

In the context of an ecosystem model, both variation and selection are based on a topology $\mathbf{g}$. Therefore, in order to characterise the influence of operators, topology must be included. Similarly, selection is not limited to one stage and can be used for parent selection pressure s_{parent} before variation $v(.)$, as well as survival selection s_{survive} pressure. This can be expressed as:

$$\mathbf{x}[t+1] = s_{\text{survive}}(\mathbf{g}, v(s_{\text{parent}}(\mathbf{g}, \mathbf{x}[t]))) \tag{3.5}$$

Generational Gap Models

During the process of variation new individuals are created, and some form of *survivorship selection* and *replacement selection* is used to determine *if* and *where* (respectively) in the population the new individuals will be allocated. However, the *when* of this generational replacement has not been clearly defined [75].

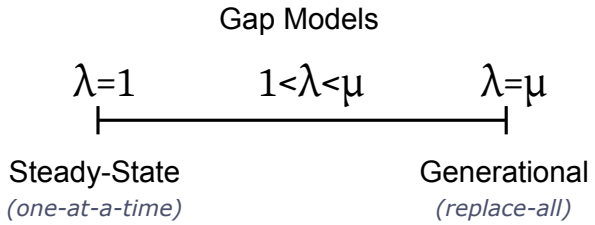


Figure 3.10: Unified gap model from steady-state to generational

There are two extremes: the entire pool of λ offspring individuals replacing the entire μ parent population in a single generational change $\mu = \lambda$, and the minimal case of a single new offspring $\mu = 1$ added to a population. In between this "gap" are a range of replacement techniques known collectively as "generation gap" models (see Figure 3.10), where *steady-state* $\mu = 1$ and *generational* $\mu = \lambda$ models are two special subclasses [334, 6].

Parallel Implementation

To consider the interaction and influence of components we should also consider the practical advantage of parallel, and hence distributed, processing models. Cantú-Paz [49] (and related work by [263]) has classified parallel EAs into four major categories, shown in Figure 3.11 and described below:

- **Global:** A single panmictic population model with parallel execution of operators such as evaluation and variation. Essentially a master-slave model with central control that can be much faster than sequential processing.
- **Coarse-grained:** Sparsely connected subpopulation "islands" ("demes") working independently except for infrequent migration of individuals (or clones) [56, 337, 32, 51].
- **Fine-grained:** Regular spatial distribution ("cellular") of the population with localised overlapping neighbourhoods defining the scope of selection operators (for competition, relative fitness, replacement) [224, 245, 145, 65, 66].
- **Hybrid:** A combination of parallel models at two-or-more levels.

An interesting result of coarse-grained models, observed by Cohoon and others [56], is that novel solutions were derived shortly after migration events. This result supports the theory of *Punctuated Equilibria* [101] which suggests that communities of species tend to have long periods of stability, punctuated with rapid change events. The rate of migration,

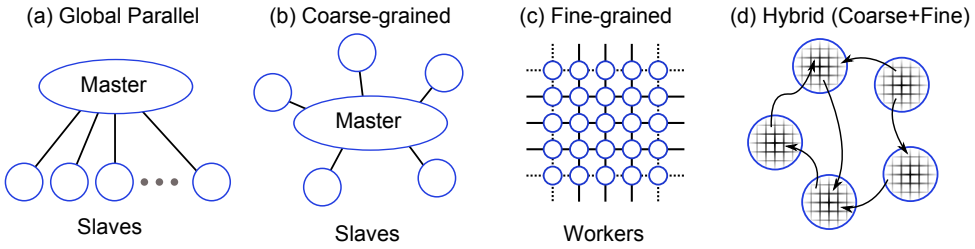


Figure 3.11: Parallel EA Models as defined by [49]. (a) Global (master-slave) (b) coarse grained, (c) fine grained and (d) hybrids.

and migrant selection type, has a strong influence on both convergence speed and the success qualities of a coarse-grained search.

Many hybrid models have been defined in the hopes of improving performance or search outcome. For example, a combination of fine-grained topology encapsulated by island models with sparse connections and infrequent migration, is an interesting model to encourage both diversity and exploitation. Of course, increasing the number of parameters and configuration variations creates complexity without necessarily any clear or measurable justification. Self-organisation is preferable in this case. See Cantú-Paz [49, 51] and Alba and Troya [7, 6] for more detailed reviews and discussions.

Summary

Table 3.3 present a grouped summary of the key components identified for inclusion in an ecosystem model of evolutionary algorithms.

Component	Function Description
Representation	The *search domain*, the structure of *individuals* and *traits*, *populations* and *topology*, and *community* interactions between *species*.
Initialisation	Creation and seeding of *individuals*, *population* structures and *communities*.
Evaluation	*Expression* or mapping of genotype to phenotype, and the use of *context* specific (environment) *fitness* functions for particular *selection* tasks (to determine fitness values).
Selection	*Fitness function* based selection of *individuals* for different *stages*; *parents*, *mates*, *offspring* survival, *expiration* and *migration*.
Variation	Methods and parameters for *recombination* of traits or introducing new trait *variation*. Includes *fertility*, *crossover* and *mutation*.
Migration	*Immigration* and *emigration* of individuals between *subpopulations* or *communities*. Based on *selection* policies.
Termination	Test algorithm *performance* and specific problem *objectives*.

Table 3.3: Ecosystem EA components and brief descriptions

3.3 Common Dialect Classification

3.3.1 The EA Union

Within review literature, there are four types of evolutionary algorithm (EA) that have emerged as the main "dialects" [98] of contemporary Evolutionary Computation (EC). They are Genetic Algorithms (GA), Genetic Programming (GP), Evolutionary Strategies (ES) and Evolutionary Programming (EP). All four are intrinsically based on evolutionary selection principles, and all have been applied to domains of search and problem solving.

All of these dialects are instances of evolutionary computation (EC), or simply evolutionary algorithms (EA). The classification of the main types of EC is strongly identified by many reviews and introductions to the field of EC [17, 34, 274, 111, 98, 128, 77]. Some reviews may refer to only ES, EP and GAs, and leave out GP due to its later development history and similarity to GAs. The recent development and popularity of Differential Evolution (DE) [330, 331] is also considered by many as a significant EA paradigm.

It is interesting to note that the three initial types of EA (EP, GA and ES) were developed independently for many years [111, 77]. Each include biologically inspired similarities, and as communication between practitioners of the EC fields increased, a healthy sharing of ideas followed. This has diluted the differences between the dialects which are now historical rather than distinctive. As David Fogel has put it, "the practical utility of each of these terms [dialects] has evolved to be essentially useless" [113, p86] since little or no information is conveyed by a dialect distinction, and theoretical understandings apply across the range of EAs.

One value of a brief look at historical difference of dialects is that it provides a context for discussing the many possible variations of EA components including some justifications and practical expectations.

3.3.2 Evolutionary Strategies (ES)

The development of Evolutionary Strategies (ES) was first carried out by Bienert, Rechenberg and Schwefel in the 1960's at the Technical University of Berlin, and have been studied extensively in Europe [289, 308, 309, 310, 312, 290].

ES were first applied to find solutions to engineering design problems, such as aerodynamic and hydrodynamics design optimisation problems where it was impossible to effectively model and optimise the problem conventionally. Other early examples include optimisation of the shape of reinforced concrete shells [156], prosthetic design and thermal water jet design [20].

As a characterisation, the features of ES were the encoding of design parameter values as a fixed-length real-valued vector (the modelling of a phenotype), mutation used to create new variations (no recombination), deterministic selection of the best solution(s), and strategic parameters such as the "on-line" self-adaptation of mutation parameters.

This type of solution encoding is considered to be *behavioural* as opposed to *structural*. "Consequently, arbitrary non-linear interactions between features during evaluation are expected which forces a more holistic approach to evolving solutions" [11].

In a typical ES, N parents are selected uniformly (not fitness based) and used to create new offspring solutions. Then N survivors are selected deterministically (truncation) either from the offspring or the combined pool of parents and offspring [326].

The main reproduction operator used has been Gaussian mutation. Distinctively, recombination was not initially used (no crossover). Later an *intermediate recombination* operator was tried in which two parents are selected and their values averaged to create new offspring solutions.

A canonical reference ES is described in Section 3.4 showing the population structure, transitions and operators.

3.3.3 Evolutionary Programming (EP)

Evolutionary Programming (EP) was created by Lawrence Fogel in the 1960's with the hope of "artificial intelligence through simulated evolution" [116, 117]. EP was largely overlooked in the 1970's, however it was later redeveloped and promoted in the late 1980's by Lawrence's son David Fogel [109] to solve more general tasks including prediction, optimisation and machine learning work [110].

Although developed earlier and independently from ES, EP uses a similar strategy of *mutation driven variation* and no crossover operator [21]. The genotype encoding used by Lawrence Fogel represented transition tables for 'Finite State Machines' (FSM), while later applications adopted specific representations appropriate for the search domain. Fixed-length real-valued vectors are common.

The selection strategy is to allow the entire population of N individuals to be parents (ie. one offspring each), and then probabilistically select survivors based on fitness from the combined parent and offspring population ($2N$). Reproduction is typically via Gaussian mutation. There are three EP forms of note; *standard* EP, *meta*-EP and *Rmeta*-EP, with meta-EP being the most common. Differences relate mainly to the degree of self-adaptation used [17, 21].

EP has been successfully applied to real-valued function optimisation [114, 115], and abstract problem domains such as searching for effective artificial neural network architectures [313]. In this case, weights were represented as real values while nodes and connections were represented as symbols of a finite set.

3.3.4 Genetic Algorithm (GA)

Genetic Algorithms (GAs) are probably the most well known of evolutionary algorithms. Some of the earliest work on simulated evolution models, mentioned in Section 3.1.2, are perhaps best described as genetic algorithms. With respect to the field of EC, John Holland developed and popularised a GA model while at the University of Michigan during the 1960s and 1970s [164, 165, 166]. GAs were developed and analysed further by David Goldberg [136] and many others.

A GA emphasises a genotype level of representation and manipulation, and was specifically proposed to investigate and utilise robust adaptation. Three historic characteristic features of GAs are fixed-length bit-string (binary array) representations, crossover as the

primary reproduction operator with minor mutation, and proportional parent selection based on fitness.

The use of a fixed-length binary string as the genotype is quite appropriate for some set-based or binary problem domains. However, to apply GAs to function optimisation requires the mapping or expression of the genotype to a phenotype space. For a real-value phenotype, the binary genotype is mapped, via a look-up table or similar method, to a quantised set of output values.

Classic reproduction is via *one-point*, *two-point* or *uniform crossover*, inspired by gametogenesis (meiosis) in sexually reproducing multiploid organisms. The most common GA reproduction process uses two parents and creates two offspring at each step. Infrequent bit-flipping mutation is easy to conceptualise and apply to the simple binary gene values.

A great number of crossover and mutation operators have been proposed. Primarily though, reproduction operators function by either *recombining* existing information or by introducing new *variations*.

Traditionally GAs operate in a *generational* and non-overlapping manner, whereby a new population of offspring individuals will replace the older parent generation. Individuals are selected to be parents based probabilistically on their fitness values.

GAs have been applied to many discrete design and real-valued optimisation problems. Goldberg optimised gas pipeline control [132] and structural designs [136], while other examples include keyboard layout optimisation [131], strategy acquisition [150] and many others. (See [16, 17, 237, 98, 77].)

A canonical reference GA is described in Section 3.4 showing the population structure, transitions and operators. A recent Generalised Generation Gap (G3) is also described which is a real-valued steady-state algorithm using specialised Parent Centric Crossover (PCX) that is particularly well suited to real-valued function optimisation [82].

3.3.5 Genetic Programming (GP)

Genetic Programming (GP) was developed by John Koza [210, 211, 212] around the early 1990's and later researched and promoted by others [214, 24]. As a comparison, GP can be considered a GA with specialised genotypes (data structures) and recombination operators. Specifically, genotypes are expressed as phenotype program trees.

There is some earlier work by Cramer to evolve programs of a simple, fixed length, sequential form [59]. Similarly, Lawrence Fogel evolved states and transitions for FSM (automata) for his earlier work developing Evolutionary Programming (EP) [117]. In contrast GP uses a flexible *variable length genotype* data structure and *sophisticated recombination* operators that work on the form of program tree "branches".

The genotype data structures in GP are computer programs, or more descriptively "symbolic expressions" (s-expressions) that can be represented as *program trees*. Leaf nodes are value labels, while each internal node is one of the available function labels. A program tree is evaluated in left-most depth-first manner, where each leaf is a correspond-

ing value, and each parent node a function whose children are used arguments. Because of this tree structure, solution *programs* – not solution *parameters* – are evolved.

Inherently, there exists the possibility for equivalent phenotype forms to be produced by different genotypes. There are often many unique program trees that can provide the same functional outcome. GP represents an important change in concept from objective parameters to programs.

GP is not limited to searching for simple mathematical relationships. The homepage of John Koza[15] presents a list of 36 instances where, specifically, genetic programming (GP) has discovered "human-competitive" results, including some that infringe or duplicate the functionality of 20th and 21st century inventions, and two patentable new inventions. The results cover areas such as sorting networks, cellular automata, electrical circuit design, controllers, antenna and computational molecular biology.

3.3.6 Structured EAs

In contrast to the common EAs described so far with single population models of *panmictic*[16] structure, there is also a long standing tradition of *structured* population models. Two of the most popular groups of structured EAs are the *distributed* EA (dEA) and the *cellular* EA (cEA)[17] [7, 6].

Figure 3.12 shows a basic comparison of structured population topology. In the traditional panmictic EA model each individual can (potentially) interact with all other individuals, while distributed and cellular models support subgroups and different levels of interaction.

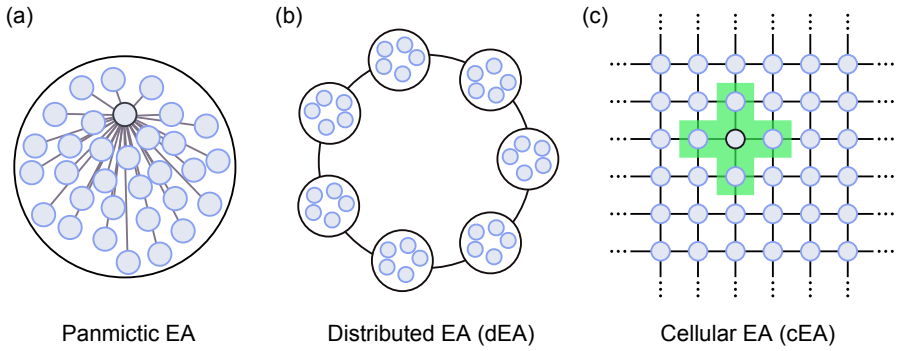


Figure 3.12: Basic population models for (a) panmictic (showing a single individuals' star topology interaction to all other individuals), (b) coarse-grained distributed EAs and (c) fine-grained cellular. Adapted from Fig 3. of [6].

Distributed EAs are a "*coarse-grained*" decentralisation of a single population by partitioning it into several subpopulation islands and managing sparse exchanges of migrating individuals. A dEA needs to include controlling parameters and regulate when migration occurs, and how individuals are selected and inserted from and into subpopulations [32, 337]. The migration policy determines the topology.

[15] http://www.genetic-programming.com/ accessed January 2009 and last updated July 2007.
[16] Meaning a population where every individual can interact with every other individual.
[17] The use of lowercase 'd' and 'c', for dEA and cEA respectively, is consistent with reviews such as [6]

Cellular EAs create a decentralised "*fine-grained*" localised neighbourhood for each individual, and as a result localised selection and variation operators. The overlap of each small neighbourhood can help the exploration process [23]. The shape of neighbourhoods can also be varied with a direct impact on search progress as it alters the effect of selection and the spread of traits [300, 299].

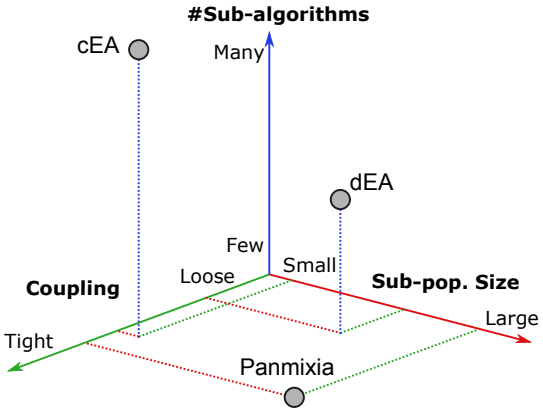


Figure 3.13: A representation of the relationship between sub-population size, coupling and sub-algorithms needed for cellular EA (cEA), distributed EAs (dEA) and standard panmictic population models. Adapted from Fig 4. of [6]

One practical incentive for structured population models is simply to take advantage of available parallel hardware. Overall however, both dEAs and cEAs seem to provide a sampling benefit of the search space over a basic EA implementation [143, 4, 51].

While dEAs typically have a few large sub-populations with loose migration coupling between them, cEAs are characterised by many small neighbourhoods with very tight (overlapping) coupling between them. This idea is represented in Figure 3.13. Note that a panmictic population is considered to have tight coupling simply because interactions are centrally controlled.

There are other less-conventionally structured EA models, and the concepts of Figure 3.13 are representative rather than definitive. For example, a hierarchical model can include both fine and coarse grained topology, with supportive migration policy to help support incremental or developmental evolution models [161, 160].

3.3.7 Other Approaches

Introduction

There are many variations of the evolutionary computation paradigm; some of unique and novel quality, others simple variations of a theme. There are also new paradigms that have some qualities or features of EC but are not be directly classified as EC by researchers. Some of the more interesting and recent proposals include Differential Evolution, Extremal Optimisation, Artificial Immune Systems and Cultural Evolution. In each case similarities and differences to the main stream EC paradigm are considered.

Differential Evolution (DE)

Differential Evolution is an evolutionary algorithm introduced by Storn and Price [330, 331]. The canonical approach uses real-valued vectors for individuals, similar to Evolutionary Strategies, and has been applied to function optimisation.[18] DE follows a basic EA generational flow, however reproduction is performed using a "weighted differential" vector to create a "mutant population" rather than a traditional crossover and mutation scheme.

In DE, a *target* vector (individual) and a *base* vector are selected from the current population. The *base* vector and a weighted *difference* vector, created from two additional sample vectors, are added to create a mutant vector. The *target* vector and the mutant vector are then combined, using crossover, to create a *trial* vector. If the *trial* vector (offspring) is better than the *target* vector the *target* vector is replaced.

It is the use of the differential vector, sampled from the population, that enables DE to scale variation changes in relationship to the distribution of the population in the search space. In doing so DE avoids the need for a separate mutation distribution parameter as used in ES.

Extremal Optimisation (EO)

Extremal Optimisation (EO) does not use a population of individuals and so is not considered an example of an EA. EO works with a single individual and attempts to identify, and make changes to, the worst component [38, 37]. Components need to be represented in a way that enables the contribution of each component to be "credited" with a portion of the individuals overall fitness. This is also in contrast to EAs that avoid the difficulty of the credit assignment problem.

In relation to ecosystems, system organisation and evolution models presented earlier in Section 2.4.9 of Chapter 2, the EO micro-level of credit assignment and component adaptation can place EO as a micro-evolutionary model. Although it is unlikely that the field of EC would want to place EO as an EC paradigm (because of EOs differences to existing and well accepted EAs), it is possible for the proposed ESEC model proposed later in Chapter 5 to include EO as a micro-evolutionary example.

Artificial Immune Systems (AIS)

Artificial Immune Systems (AIS) were first developed in the 1980s as a problem solving metaphor based on immune network models [104, 35]. Well known AIS implementations utilise four main theories [64, 127] to enable adaptive change: *negative selection*, *clonal selection*, *immune networks* and *dendritic cell* (*danger theory*) .

AIS are clearly a useful biologically inspired computation model, however AIS do not closely resemble the evolutionary models of EAs [127] and so are not considered by many researchers as part of EC despite the "selection" concepts.

[18] The DE homepage of Storn is at http://www.icsi.berkeley.edu/~storn/code.html

Cultural Algorithms (CA)

As mentioned in Section 2.3.4 as part of the discussion of communities, it is possible to consider the evolution of culture as the transfer and alteration of ideas. The Cultural Evolution (CE) model proposed by Reynolds [293] uses principles of human social evolution as an extension to conventional EAs. Cultural Algorithms (CAs) make use of a "*belief space*" (or "meme pool") which contains useful knowledge of the search space that can be shared across the population and across generations to assist and bias a search. The beliefs effectively allow the search space to be reduced. Both the population space and the belief space are updated and co-adapt with each other.

Cultural Algorithms are similar in some respects to the evolutionary idea of the Baldwin effect, in which partial fitness of an individual is credited to its ability to acquire and learn, but not directly part of the genotype.

Artificial Life (ALife)

It is worth noting the models from the ALife ("artificial life") field, especially given that ALife considers the notions of topology and ecology to a greater extent [287, 169, 31] than EA models of the field. ALife is usually not primarily concerned with computational performance, but rather in emulating interesting and emergent qualities exhibited in natural systems.

Summary

The number of alternative approaches mentioned here is limited and concerns those based on evolutionary principles. There are many other adaptive strategies inspired from biology that can also be compared to EAs in several regards. For example, within the field of swarm intelligence (see [102]) the biological examples flocking birds and ant colonies have both been used to create optimisation algorithms – particle swarm optimisation (PSO) [189] and ant colony optimisation (ACO) [90]. Both examples use populations of individuals which can decentralise and parallelise a search process, and both methods have been applied to many domains, including multiple objective optimisation (MOO). Clarifying and classify the breadth of research in related areas is a challenging and ongoing objective. See, for example, [13] for a taxonomy of multiple objective ACO algorithms.

An extensive consideration of the many biologically inspired search metaphors and their applications is far outside the scope of this thesis.

3.4 Reference Algorithms

3.4.1 Introduction

Several standard and structured evolutionary algorithms are now described in more algorithmic and configuration detail as a reference for investigations in later chapters.

3.4.2 GA: Genetic Algorithm

Table 3.4 presents a representative summary of configuration components and parameter values used in a genetic algorithm instance. Note that GA performance is very sensitive to parameter values and typical values can not be given. Although very popular during the early development of GAs, bit-string representation is no longer the common scheme used for problem domains. A small code[19] example of a GA search routine is shown in Listing 3.1.

Representation	
Population	Panmictic (size m)
Individuals	Bit-string (length l)
Traits	Mapped* (geno-to-phenotype)
Selection Pressure	
Generations	Non-overlapping
Parent Selection	Fitness proportional
Mate Selection	Fitness proportional
Survivor Selection	Truncation (age)
Elitism	*Optional*
Variation	
Parents : Children	$2 \rightarrow 2$
Recombination	One-point crossover
Mutation	Bit-flip
Settings	
Parent pop. size m	> 20
Offspring pop. size n	$n = m$
Crossover prob.	$[0.5, 1.0]$
Mutation freq.	$1/l$

Table 3.4: Canonical GA configuration

The required form of trait mapping depends on the phenotype required. For example, binary SAT problems map directly to a bit-string genotype, while integer or real values (for continuous-value function optimisation) need to be mapped, and this requires the resolution (number of bits per value) to be specified also.

Survivor selection is listed as "Truncation (age)", indicating that the parent population and the new offspring population are combined and then reduced (truncated) to the original parent population size m, ordered by age. In this way, the older parent individuals are removed, and only the new offspring remain. This behaviour is also indicated by the "non-overlapping" description of the "Generations".

One-point crossover uses two parents and creates two children. n-point and uniform crossover are also common. If a crossover probability is used, then there is a chance of pure parent cloning. Classic GAs use strong crossover and low mutation ($1/l$) but this also can be changed to any probability to increase injected variation.

As the entire parent population is replaced by the new offspring population ($n = m$) there is no survivor selection pressure (neutral). Tournament selection is a popular

[19] The code syntax is that of the Python programming language.

alternative parent/mate selection scheme to fitness-proportional selection. This sequence of selection pressure is also known as "up-front" as it happens during parent selection and before variation.

Listing 3.1: Standard GA Search in Python

```
def GeneticAlgorithm(m, n, fn):
    # fn is a fitness function to evaluate individuals
    parent_pop = Population(size=m)
    t = 0
    while search_not_finished():
        offspring_pop = select_and_clone(parent_pop, size=n)
        do_crossover(offspring_pop)
        do_mutation(offspring_pop)
        evaluate_all(offspring_pop, fn)
        parent_pop = offspring_pop
        t += 1
```

The sample code makes some simple assumptions that a termination method, the parent selection method and a fitness function have been defined. Note that the implementation shown uses the unified offspring population model, where all selection and parent cloning is done as a group before variations. This is equivalent to an inner select-crossover-mutation iteration pattern for this type of population topology.

3.4.3 ES: Evolutionary Strategy

A simple non-overlapping Evolutionary Strategy configuration is shown in Table 3.5 using only Gaussian mutation for variation. An overlapping version would have stronger selection pressure and be implicitly elitist. In either case, selection occurs after variation. Adaptive mutation size is an effective strategy for domain specific scaling of variation.

Listing 3.2: Standard (m, n)-ES Search in Python

```
def EvolutionaryStrategy(m, n, fn):
    # fn is a fitness function to evaluate individuals
    parent_pop = Population(size=m)
    t = 0
    while search_not_finished():
        offspring_pop = select_and_clone(parent_pop, size=n)
        do_mutation(offspring_pop) # no crossover
        evaluate_all(offspring_pop, fn)
        parent_pop = select_best(offspring_pop, size=m)
        t += 1
        update_mutation_step_size()
```

The code sample Listing 3.2 assumes a method is available to update mutation step size. For example, according to the classic 1/5th adaptive update rule, when the reproductive success (ratio of children that have improved fitness) is above a threshold value the mutation step size is increased, and below another threshold value the step size is reduced.

Note that in this example mutation is the only variation operator (no crossover) and that the offspring population size is usually much larger than the parent population size.

Representation	
Population	Panmictic (size m)
Individuals	Real, vector (length l)
Traits	Phenotype (direct)
Selection Pressure	
Generations	Non-overlapping (m, n)
Parent Selection	Uniform random
Mate Selection	None
Survivor Selection	Truncation (best)
Elitism	No (Yes if overlapping)
Variation	
Parents : Children	$1 \rightarrow 1$ (clone)
Recombination	No
Mutation	Gaussian (adaptive)
Settings	
Parent pop. size m	< 10 (small)
Offspring pop. size n	$n >= m$
Crossover prob.	Not used
Mutation prob.	1.0

Table 3.5: Canonical (m, n)-ES configuration. Parent m and offspring n population sizes are traditionally represented by λ and μ respectively. The canonical overlapping ES model is described with a $(\lambda + \mu)$ notation.

Other ES variants make use of multi-parent crossover, where for each position of the offspring vector, either arithmetic, intermediate (average) or simple random discrete selection is used.

3.4.4 G3: Generalised Generation Gap Model

The Generalised Generation Gap (G3) model, proposed by Deb, Anand and Joshi [82, 81], is a modification of a commonly used real-valued Genetic Algorithm known as the Minimal Generation Gap model (MGG) [302, 201, 200, 345]. G3 is described by the authors as a "steady-state, elite-preserving, scalable and computationally fast model". The essential steps can be listed quite simply as follows:

1. Select the best and $\mu - 1$ parents uniform randomly from the population
2. Using the pool of μ parents, create λ offspring using recombination (ie. PCX)
3. Select another uniform random individual p_r from the population
4. Select the best individual from a combined $\mu + p_r$ group and place into the population at p_r position

As a point of difference to the MGG model, the G3 model removes a relatively expensive roulette-wheel selection operation and replaces it with a simple block (elitist) survival and replacement selection. A typical G3 configuration is shown in Table 3.6. Note that there is no mutation, however the recommended recombination operator (PCX) is stochastic and introduces random variation.

Representation	
Population	Panmictic (size m)
Individuals	Real, vector (length l)
Traits	Phenotype (direct)
Selection Pressure	
Generations	Steady-state (overlapping)
Parent Selection	Deterministic (the best)
Mate Selection	Uniform random ($\mu - 1$)
Survivor Selection	Truncation (best)
Replacement Selection	Uniform random
Elitism	Yes (implicit)
Variation	
Parents : Children	$3 \rightarrow 1$
Recombination	PCX
Mutation	No (see PCX)
Settings	
Population size m	$\sim [100, 200]$ problem sensitive
Parent group size μ	~ 3
Offspring group size λ	$\sim [2, 4]$ (small)
Crossover prob.	1.0 (always)
Mutation freq.	n/a
Variance σ_ζ and σ_η	0.1

Table 3.6: Standard Generalised Generation Gap (G3) Configuration, based on results presented in [81].

Parent Centric Crossover (PCX) is based on the same idea used for the real-valued Simulated Binary Crossover (SBX) [80] in that more probability is assigned for an offspring to remain closer to parents, while also using the separation or "differential" characteristics between parents to moderate variation.

PCX begins with a pool of μ parent individuals. The mean vector $\vec{g}$ of the parents is first calculated, and then for each offspring created, a random parent[20] $\vec{x}^{(p)}$ is selected and its direction line $\vec{d}^{(p)}$ from the mean calculated. Then, for each of the remaining $\mu - 1$ parents, perpendicular distances D_i to the line $\vec{d}^{(p)}$ are calculated and averaged $\bar{D}$. An offspring $\vec{y}$ can then be created using:

$$\vec{y} = \vec{x}^{(p)} + w_\zeta |\vec{d}^{(p)}| + \sum_{i=1, i \neq p}^{\mu} w_\eta \bar{D} \vec{e}^{(i)} \tag{3.6}$$

where $\vec{e}^{(i)}$ are the $(\mu - 1)$ orthonormal bases that span the subspace perpendicular to $\vec{x}^{(p)}$, and w_ζ and w_η are zero-mean normally distributed variables of variance σ_ζ^2 and σ_η^2 respectively.

The "parent centric" nature of this operator is easy to see if we take multiple sample offspring from a fixed group of parents as shown in Figure 3.14. Another extreme PCX parameter example would be of $\sigma_\zeta = 0.9$ and $\sigma_\eta = 0.9$ (not shown), in which case the

[20] Published descriptions of PCX have suggested "random" as well as "best" parent selection. Using the best provides a very large performance increase on non-deceptive problems.

distribution of new offspring would spread and overlapping, but still centred around each base parent.

Although PCX does increase the computation complexity of reproduction, it has been shown to be an extremely efficient approach for real-value optimisation problems, reducing overall computational search time [81].

This idea is also very similar the central concept of Differential Evolution (DE) (see Section 3.3.7). In DE a sample of two individuals from the current population are used two create a difference vector, which is then added to a base individual to form a "trial" vector. The "trial" individual is used in reproduction (crossover) with a "target" individual to form a new individual. As the sample individuals are selected from the population, the difference vector scales with the diversity of the population. Unlike DE, PCX creates additional parameters that need to be set appropriately for particular problem domains.

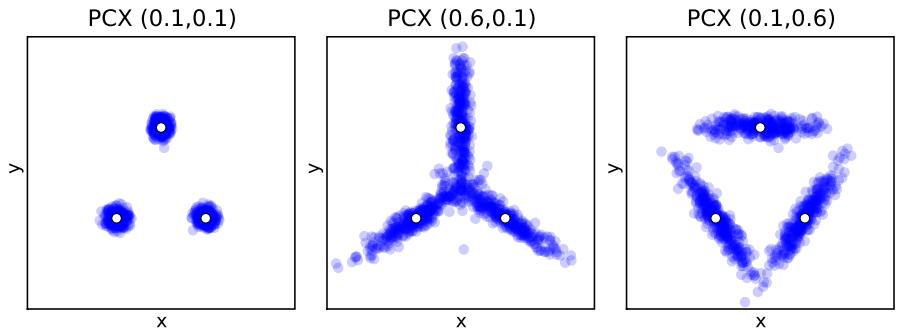


Figure 3.14: Three sample examples of $\lambda = 1000$ offspring each, in a 2D space, around a parent pool of $\mu = 3$ using Parent Centric Crossover (PCX). Note the orthonormal distribution around each parent, for different values of σ_ζ and σ_η respectively.

A sample implementation of the G3 algorithm is shown in Listing 3.3.

Listing 3.3: Sample G3 search in Python

```
def G3Search(m, mu, lam, fn):
    # fn is a fitness function to evaluate individuals
    pop = Population(size=m)
    t = 0
    while search_not_finished():
        # 1. Select random and best parent (join)
        parents = select_random(pop, mu-1) + [best(pop)]
        # 2. Create offspring using PCX
        offspring = [None]*lam # pre-size offspring list
        for i in range(lam):
           offspring[i] = recombine(parents)
           evaluate(offspring[i], fn)
        # 3. Select random population member to challenge
        c_pos = randrange(0, m)
        offspring = offspring + [pop[c_pos]] # join
        # 4. Replace with best of pool
        pop[c_pos] = best(offspring)
        # ...
        t += 1
```

3.4.5 cEA: Cellular Evolutionary Algorithm

The idea of the cellular approach is to perform decentralised selection and variation for each cell, with operations restricted to local "pools" of individuals. Neighbourhoods overlap, creating a smooth "diffusion" of trait transfer, due to selection pressure, across the grid as evolution progresses.

Table 3.7 lists a configuration for a classic cEA model of a 2-dimensional 32×32 toroidal grid using a $k = 4$ Von Neumann Neighbourhood of five individuals and four connections. Other well-used classic neighbourhoods include the $k = 8$ Moore and $k = 12$ Extended Moore neighbourhoods. See Figure 3.15 for examples.

Representation	
Population	2D Grid (size $m = w \times h$)
Individuals	Bit-string (length l)
Traits	Mapped (geno-to-pheno)
Selection Pressure	
Generations	Steady-state (distributed)
Parent Selection	Deterministic (in order)
Mate Selection	Linear rank proportional
Survivor Selection	Truncation (best)
Elitism	*Implicit*
Variation	
Parents : Children	$2 \rightarrow 1$
Recombination	One-point crossover
Mutation	Bit-flip
Settings	
Grid pop. size m	$m = 32 \times 32 = 1024$
Parent pool size $k + 1$	$k = 4$ (ie. Von Neumann)
Offspring pop. size n	$n = 1$
Crossover prob.	$[0.5, 1.0]$
Mutation freq.	$1/l$

Table 3.7: Cellular EA configuration based on a simple binary GA

Listing 3.4 show a based fine-grained cellular EA search using localised fitness (relative to the neighbourhood) and mate selection (from the local neighbourhood). In this example two-parent recombination is used (one-point crossover) with a $1/l$ chance of mutation. It would be possible to distribute (allocate) each cell's operations and marshal the entire new generation at once, or to simply allow concurrent access by different worker processes.

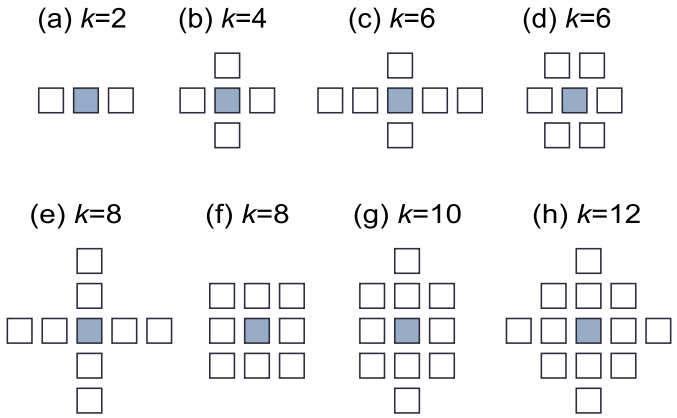


Figure 3.15: Planar regular neighbourhood models of increasing degree k including alternative configurations. Classic forms are the (b) Von Neumann, (f) Moore and (h) Extended More neighbourhoods.

Listing 3.4: Fine-grained cellular EA search in Python

```
def FineGrainedGASearch(w, h, fn):
    # Create and evaluate initial population
    grid = GridPopulation(width=w, height=h)
    t = 0
    while search_not_finished():
        for cell in grid: # in parallel (order)
            evaluate(cell) # current relative fitness
            neighbour = select_neighbour(cell)
            offspring = recombine(cell, neighbour)
            do_mutation(offspring)
            cell = offspring
        t += 1
```

3.4.6 dEA: Distributed Evolutionary Algorithm

In Table 3.8 one possible configuration for an island population model is shown. Key additions are the parameters for number of islands, island population sizes, migration interval and size of migration groups – this all adds complexity. (See earlier Figure 3.13 for a visual comparison of coupling, pop-sizes and sub-algorithms between panmictic EAs, cEAs and dEAs.) Migrant selection pressure can be applied both when individuals are selected for emigration, and when individuals are integrated during immigration.

It has been reported by Cantú-Paz [52] that selection types have a big impact on the success of the search. The migration selection policy of least computation effort is simple fitness based replacement where the best are selected and survive. However, this is not always the best integration (mixing) policy.

Listing 3.5 shows a simple synchronous island population example where migration is controlled by a single method, however given the infrequent migration policy, this is an especially good opportunity for decentralised (subprocess and distributed) island algorithm control.

Representation	
Population	Islands (sub-populations d)
Sub-population	Panmictic (size = m)
Individuals	Bit-string (length l)
...	...
Selection Pressure	
Generations	Non-overlapping (within islands)
Parent Selection	Fitness proportional
Mate Selection	Fitness proportional
Survivor Selection	Truncation (age)
Emigrant Selection	Truncation (best)
Immigrant Survival	Truncation (best)
Settings	
No. Islands	$d = 5$
Island pop. size	$m = 30$
Migration Interval	$dt = 20$ (generations)
Migrate group size k	k (individuals)
...	...

Table 3.8: Distributed island EA configuration simplified to selection properties. The ... are used to indicate that additional settings and values would be required but have omitted.

Listing 3.5: Coarse-grained island distributed EA search in Python

```
def IslandGASearch(demes, m, k, dt, fn):
    # Create initial island populations
    islands = [None]*demes # pre-size list
    for i in range(demes):
        islands[i] = Population(size=m)
    # Search
    t = 0
    while search_not_finished():
        for island in islands: # in parallel
            offspring_pop = select_and_clone(parent_pop, size=m)
            do_variations(offspring_pop)
            evaluate_all(offspring_pop, fn)
            parent_pop = offspring_pop
            if t % dt = 0:
                # send our best to random selected island
                nei = select_neighbour_island()
                best = select_best(parent_pop, k)
                send(best, dest=nei)
                # receive and keep best immigrants
                best = receive(source=nei)
                parent_pop = select_best(parent_pop + best, m)
        t += 1
```

3.4.7 Closing

The algorithms presented in this section act as a reference for later investigations. A particular emphasis was placed on selection and its various contexts, as these are a critical feature of the ecosystem model for evolutionary algorithms.

3.5 Summary

This chapter has attempted to bring together many of the components and processes of ecological ecosystem identified in Chapter 2, and to review the application of evolution as a stochastic and iterative search algorithm through adaptation.

Components of an ecosystem EA model have been presented with an emphasis on aspects that specifically relate to ideas of community, interaction and topology. Of these, selection clearly plays a very important role, and it is the different selection types, and their application in specific contexts, that create open topology related questions.

The simple topologies considered so far in standard models of evolutionary computation are a faint reflection of the complexity in natural systems. An appropriate implementation framework for ecosystem EAs must include the capacity for specific and flexible topology as part of operator components.

Chapter 4 continues the development of the ecosystem EA model by looking in some detail at complex systems and abstracted graph topology. In particular, it considers recent observations about the statistical properties of real-world complex systems and artificial models developed to emulate desirable qualities. Chapter 5 is a return to the ecosystem EA model and how it can be extended with complex topology ideas, and to consider the questions it opens for investigation.

Chapter 4

Graph Theory, Topology and Efficiency

4.1 Introduction

4.1.1 Networks and Graphs

Graphs and networks are ubiquitous in natural and artificial systems, and exhibit interesting and important qualities. Recent research, presented in work such as [26, 92, 257, 359, 259, 36, 361], has focused on the comparative study of networks, with particular emphasis on graph properties that are common to many systems. Several disciplines – notably mathematics, physics and sociology – have independently observed universal paradigm, concepts and properties within network systems. The interdisciplinary significance of complex networks and graphs is an important, practical and exciting aspect of graph theory research and application.

In simplest terms, a graph is a set of nodes connected by edges. The "nodes" can be used to represent many different concepts, and edges the relationships between nodes. However, the properties of real-world complex networks can not be understood by simply describing individual nodes and links.

This chapter examines current research into graphs and complex systems, including graph measurements and properties, and further investigates the techniques required to observe, measure, grow and utilise efficient topologies. This knowledge is used to support the models of ecosystems and evolutionary computation presented in earlier chapters, and the investigations of later chapters. Although this chapter includes topics not directly used in the later part of the thesis, such topics are included for completeness and because they relate to future potential developments.

4.1.2 Graphs Everywhere

It is generally agreed that "ideal" complex systems are exemplified in naturally occurring social and biological systems. However, information and technological networks built by humans also display remarkable structure and function [257, 26, 359].

Mark Newman has suggested, specifically for the purpose of discussion, a division

of real-world networks into categories of *social*, *information*, *technological* and *biological* systems [257]. Below is a modified list of the real-world network examples included in each category from Table II of [257].

Social Networks: Film actors, company directors, academic co-authorship, telephone calls, email messages, email address books, student relationships and sexual contacts.

Information Networks: WWW pages (hypertext links), search engine categories (ie Altavista directories), citation networks (academic publications), Roget's thesaurus and word co-occurrence in literature.

Technological Networks: Internet infrastructure, power grid structure, water and gas utility distribution, train routes, roads, software packages, software classes, electronic circuits and peer-to-peer network formation.

Biological Networks: Metabolic networks, protein expression and interactions, marine and freshwater food webs, neural networks and circulatory systems.

There are common and remarkable qualities to real-world networks, both of natural and synthetic origins. Firstly, real-world networks are *effective* in performing their required functions, and secondly they tend to be *efficient* in terms of construction cost or operation. Thirdly, real-world networks tend to be highly resistant or *resilient* to the random removal of nodes (or edges). However, targeted removal of critical nodes will have a significant impact.

Networks and graphs are all around us, and it can be shown that their complexities are not explained by simple analysis of single nodes or edges. In order to understand how it is that real-world networks are organised and function, it is also necessary to understand better the nature of complex systems.

4.1.3 Complex Systems

Defining Qualities

Chapter 1 already introduced the idea of a complex system and the difficulty in defining qualities. It is possible to list some defining characteristics and consider the overall behaviour of a complex system.

System Complexity: A complex system may have complex non-linear interactions between components, however it is not a single interaction that is significant – it is the overall behaviour at the system level.

Emergent Properties: The idea of a complex system is strongly associated with emergent properties that are greater than the sum of individual parts.

Adaptability: Complex systems are also known for adaptability in dynamic environments.

Unique Components: Rather than a system of homogeneous components, complex systems are usually formed by unique individual units. It is not possible to simply interchange units and create the same properties.

A *complex system* is not simply a matter of many components integrated to form a *complicated system*, such as a piece of mechanical machinery with many parts. Such a machine is still relatively easy to explain by a reductionist approach, where we are able to define the components and the laws that govern interactions in a causal finite manner. Abstracting a complicated system, when individual units are interchangeable, works well for describing atoms, chemistry and other systems with mainly homogeneous components. (It is important to keep in mind that there are still many complicated models developed for specific purposes.)

Understanding complex systems is not a question of abstraction and generalisation, but rather the impact of complexity and specific individuality that govern the emergent and, in many cases, highly dynamic qualities.

Random Models

In the 1950's Paul Erdös and Alfred Rényi began to develop statistical theories to describe real-world networks using random graph models [103]. Their work was based on earlier research of random networks by Solomonoff and Rapoport [325].

The early questions of network research focused on identifying individual components, or understanding specific non-linear interactions. Later, when data became available, real systems were compared to the random graph models, and the differences observed were very illuminating.

For example, Price [282] pioneered work with a study of academic citation networks. He discovered that the distribution of citations did not peak as had been predicted by the random models of Erdös and Rényi. Instead, Price found that most articles had a low number of later citations, but a small number of articles had many citations, and hence citations followed a power-law distribution. Later in 1976, based on an earlier idea by Simon [316], Price published a model called *cumulative advantage* that could explain his observations of citation networks.

Resurgence

It was not until more recent times – almost 50 years after the random models of Erdös and Rényi – that research work to describe networks underwent a resurgence. The main trigger for this is undoubtedly the facilitation provided by the Internet [9], itself an intriguing complex network. Together with the availability of large databases, computational power and a willingness for traditionally isolated disciplines to share ideas, the late 1990s saw a massive growth in network system research [257].

The focus of this new research was strongly oriented towards connectivity properties [257]. It has been observed by Barabási and others that there are underlying networks with non-random topologies behind complex systems [26, 217]. So, it makes sense then

that in order to understand complex systems we need to understand these non-random networks and their internal interactions [255, 60].

4.1.4 Small-World Phenomena

Origins

Hungarian author Frigyes Karinthy (1887-1938) was the first to propose the notion that people are connected to each other by only a small number of intermediate acquaintances. In fact, his 1929 short story "Chains" suggest that there would be no more than five intermediates between any two people in the world.

One of the first numerical studies of social networks was carried out by Stanley Milgram (1933-1984), and his graduate student Jeffrey Travers, while at Harvard University in 1967 [232]. Several experiments were conducted using folders (or letters) distributed to a selection of people who were then asked to pass the folders on (in person) to people whom they thought would be nearer to the folder addressee. The number of folders to successfully reach the target was low[1], however results indicated that the mean number of intermediaries was six.

Milgram conjectured that any two people in the world would have a similar small *separation characteristic*, and that this gives people the impression that we live in a *small-world*. The small-world phenomena has subsequently been verified [277, 209], and the terms "six degrees of separation" and "small-world phenomena" have now passed into popular use within plays [153], games (based on movie actors, sport stars and other celebrities), media stories, movies, books [208] and urban myth. More seriously, small-world networks of individuals involved in terrorist organisations have been used to understand the roles and interactions of such people.

Other interesting results from Milgrams' work were that certain "key" people (with high levels of connectivity) were common intermediates within certain groups of people, and that participants in the chain need to be competent enough to select appropriate people for the task.

More importantly, the *small-world phenomena* has been found to exist in not just social or trivial networks, and understanding the nature of the underlying topology has significant implications for many other research fields and complex systems [359].

Models

The specific characteristics of small-world networks have only recently been modelled and understood, encouraged in particular by the work of Duncan Watts and Steven Strogatz published in Nature in 1998 [360]. They investigated and showed, using various ring lattice network models and analysis of natural and synthetic real-world networks, measurable characteristics of networks that exhibit the small-world phenomena.

[1] In the celebrated 1967 paper only 5% of letters reached the target. Later experiments achieved much higher completion rates and also suggested some of the influential factors of completion such as social and demographic groups and the perceived value of the parcel.

Research showed that small-world networks contain groups or "clusters" of nodes, with many connections between common neighbours, and that such networks do not have connections between every node which would be expensive. Instead, small-world networks contain a sufficient number of suitably placed long distance connections so that a path between any two nodes in the network only requires, on average, a small number of connections [358]. It can be seen that a feature of small-world networks is that they are *efficient* with respect to network cost and performance.

In summary, the small-world characteristics observed and modelled include:

- On average, that the path length between any two nodes in the network is low.
- There is a higher level of clustering than would be expected from a simple random model with the same number of nodes and edges.
- The overall cost of the network is low with respect to the tasks or processes that occur on the network (efficient).
- Robust (resilient) against the random removal of nodes.
- Fragile to the removal of specific (critical) nodes.
- Scale-free distribution of degree is very common.

The small-world model has been generalised by others, in particular the work by Jon Kleinberg [204, 206] who has analysed the small-world model on topologies such as grids rather than ring lattices.

4.1.5 Graph Theory

A detailed presentation of graphs and efficient topology ideas requires a description of graph components, features and properties. Section 4.2 lists many concepts and properties of graphs, with a focus toward aspects that are important to this work. We begin with some basic terms and their meanings as shown in Table 4.1. It is also interesting to take a brief look at the directions and motivations of graph theory research.

The statistical properties of networks are directly related to network topology. Topology describes the elements and connections of a network excluding physical positions or geometry, and so networks that are topologically equivalent are not altered by changes in the position of elements (see Figure 4.1). Although statistical properties and network topology are strongly related, it is essential to realise that they are both independent of any representational (drawn) geometry.

It has been noted by Mark Newman [257] that there has been a recent interesting shift in network analysis research from small networks with specific properties to large networks and statistical properties. This can be noted in the form of research questions being asked by researchers. Specific questions for small networks, such as "What is the most important node?", have been replaced by statistical questions such as "What percentage of nodes need to be removed to significantly affect network performance?"

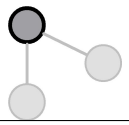
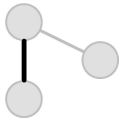
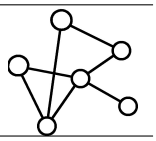
Term	Meaning	Example
Vertex	A single point or 'dot' in a graph. (Plural: *vertices*). Also called a *node* in computer science, a *site* in physics or an *actor* in sociology.	
Edge	An edge connects vertices. The two vertices are called the endpoints of the edge. Also called a *link* (computer science), *bond* (physics), *tie* (sociology), *line* or an *arc*. **Directed** edges restrict connection function to a single direction. The term *arc* is often reserved for directed edges. **Weighted** edges have an associated weight value that influences their connection function.	
Graph	A finite set of *vertices* connected by *edges*, also called a *network*. A description of vertices, edges and connections (excluding geometry or weight) is known simply as the *topology*.	

Table 4.1: Basic graph terminology and examples

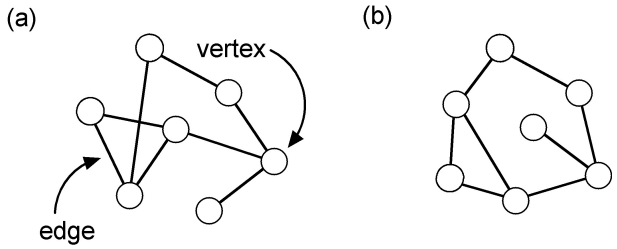


Figure 4.1: An example of simple graphs with seven nodes and eight links. Note that (a) and (b) have the same topology but are drawn differently. Effective graph drawing techniques can illuminate network properties.

This change in research to statistical analysis of larger networks has been made possible by the prevalence of communication networks and the computational power of modern computers. The desire to understand large communication networks (in particular the Internet) is one of the strongest influences driving modern graph theory research, as demonstrated by the prevalence of publications related to displaying and analysing Internet and WWW[2] topology.

Another important influence driving the shift in research is the difficulty or inability to effectively visualise extremely large networks – regardless of good visualisation techniques. Instead, an understanding of network characteristics provides an opportunity to relate the statistics of unknown large networks to networks that have been studied and characterised [259, 36]. This does not remove the importance of work being done in the field of graph drawing. (See Section 4.3 which considers visual representation in more detail.) Rather it illustrates the difficulties and expectations now placed on visualisation research. As before, effective graph drawing techniques allow us to utilise the remarkable

[2]The "World Wide Web"(WWW) is the term used by Sir Tim-Burners Lee in 1989 to describe his system of interconnected hypertext documents. The WWW is also known colloquially as simply "the web".

ability of the human eye and brain to analyse and understand the networks and structures in our environments.

The twentieth century has witnessed a substantial growth in the knowledge of graphs and complex systems. The study of graph theory is a fundamental aspect of discrete mathematics, and graph theory research continues to be an active and diverse field.

4.1.6 Additional Resources

The frequently cited works of Harary [154] or Bollobás [39] provide a good foundation into the mathematics of graph theory.

There are several recent books, directed towards the popular audience, that may interest the casual reader. These include Duncan Watts' book *Six Degrees* [359], Albert-László Barabási's *Linked* [26], and Mark Buchanan's *Nexus* [43].

For large reviews of work on the structure of complex networks, dynamics and function, see Newman [257, 258] and the more recent work by Boccaletti et al. [36]. The topic of growing graphs has been covered by both Albert and Barabási [9], and the work of Dorogovtsev and Mendes [91] (which has also been expanded into a book [92]).

A collection of essays covering many topics has been put together by Bornholdt and Schuster [40]. Similarly a collection of previously published work, with additional review material, has been collected by Newman et al. [259].

For shorter reviews on specific topics see Newman [253] (small-world models), Hayes [158, 159] (graph theory) and Strogatz [332] (network dynamics and behaviour). Section 4.3 touches briefly on the topic of graph drawing, and so the work of Di Battista, Tamassia and colleagues makes excellent review material [83]. Additional resources on the topic of graph representation, visualisation and drawing are listed in Section 4.3.4.

4.2 Graph Concepts

4.2.1 Overview

The basic components of vertices and edges have been mentioned in the introduction of this chapter, and basic graph components were illustrated in Table 4.1. The discussion in Section 4.1.3 and Section 4.1.4 has already suggested several important graph concepts and properties related to this work, specifically the small-world phenomena and the notion of efficient topology.

Further, Section 4.2.2 and Section 4.2.5 demonstrate concepts associated with sets of *vertices* and *edges*, then *paths* and *cycles*, and finally *graph types* and *properties*.

4.2.2 Vertices, Properties and Sets

Properties of *degree* (or more specifically *in-degree* and *out-degree* for directed edges) are good indicators of how information will be exchanged between nodes, and hence the terms *degree* or *valence* (as the term relates to other science fields) are good indicators of *connection capacity*.

Table 4.2 contains descriptions of vertex related terminology.

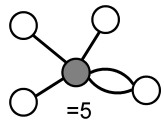
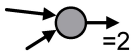
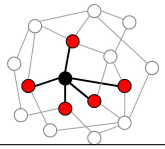
Vertex	Vertex Meaning
Adjacent	Two vertices are adjacent if they are connected by one or more edges.
Degree 	The number of edges connected to a vertex. Note that, because there may be more than one edge between two vertices, the degree may be greater than the number of adjacent vertices. Also known as the *local degree* or *valence* of a vertex. A vertex of zero degree is called *isolated*.
In-degree 	For a vertex v, the number of 'in-coming' directed edges with v as their terminal vertex.
Out-degree 	For a vertex v, the number of 'out-going' directed edges with v as their initial vertex.
Component 	A component is the set of vertices that a vertex is connected (adjacent) to by edges. This set of vertices is also known as a *neighbourhood*. For a vertex in a directed graph, there is an in-component (vertices that can reach it) and an out-component (vertices it can reach).

Table 4.2: Terminology for vertex properties and sets

A *component*[3] includes not only a set of vertices with degree properties, but also edge properties. Given that nodes can be of various types, potentially manipulating information in many different ways, and that edges may also be weighted and adjust information that travels through them, even a simple component (in this formal sense) can be a complex system in itself.

4.2.3 Paths and Cycles

A *path*, the sequence of edges between *initial* and *terminal* vertices, can be viewed as a static set of properties. A path may also describe the properties that effect an instance of communication or transfer between any two vertices. For example, knowing the number of edges, how they are weighted and the properties of each vertex passed through, provides a sequential description of the influences on any information travelling a specific path. See Table 4.3 for *path* related terminology.

There can be many, possibly equivalent, paths between two vertices in a graph. A *geodesic path* is defined as the shortest, and it is possible that several equivalent geodesic paths exist between two vertices. The properties of geodesic paths within a network are often central to statistical calculations of graph properties. When a path is directed and returns back to its initial node the path forms a *cycle*. Directed graphs with cyclic paths

[3]Unfortunately, the word *component* is also commonly used to refer to a single vertex or edge. This thesis uses, depending on the context, the word "component" both in its formal graph theoretic meaning, but also in a general sense.

Term	Meaning
Path	A sequence of consecutive edges in a graph, also known as a *chain*. The *length* of the path (also called *distance*) is the number of edges traversed (or the number of vertices minus one).
Initial vertex	First vertex of a path.
Terminal vertex	Last vertex of a path.
Cycle	Also known as a *loop*. A path in a directed graph with the same initial and terminal vertex.
Acyclic graph	A graph that does not contain any cycles or loops.
Bridge	An edge in a graph whose removal results in a disconnected graph.
Geodesic path	The shortest path in a graph between two vertices. There may be more that one equivalent geodesic path between vertices.
Euler path	A path that passes through all graph edges.
Euler cycle	A cycle that passes through all graph edges.
Hamiltonian cycle	A cycle that passes through each graph vertex once.

Table 4.3: Terminology related to paths, cycles and graphs

have strong applications to the study of control theory, information feedback and recurrent systems.

Formal names have been given to a cycle or path that passes through all graph edges, called an *Euler cycle* and *Euler path* respectively. If a cycle passes through every vertex of the graph once, it is known as a *Hamiltonian cycle* and finding such paths is the focus of a range of optimisation techniques.[4]

4.2.4 Critical Components

Within many real-world networks there exist important edges or vertices whose function is critical. This may be because they provide a unique quality to a network, or because their removal dramatically alters topology. If an important *bridge* link is removed from a network (for example a transatlantic telecommunications cable), it can result in two *disconnected* (or isolated) networks.

It is important to be able to identify critical links and nodes in a graph, as this is a direct indication of network robustness and susceptibility to attack. The issue of network vulnerability and robustness is very important to real-world networks, in particular communication and data networks.

As an example, the Google[5] search engine has become a standard node that many Internet users rely upon to access information on the Internet. If the Google web site ceased to work, it would have a significant impact on many people, and so the Google web site node can be considered a critical component for people searching for information.

Another interesting point to note from this example is that the Google node is only *critical* for those users that place a high *utility* value on it. This illustrates that a node

[4]The Travelling Salesman Problem (TSP) is a prime example of the search for a Hamiltonian cycle.
[5]See http://www.google.com

has different utility value with respect to the specific processes (such as navigation) that can occur on the network.

4.2.5 Graphs Terms and Properties

Graph properties of *size*, *order* and *degree* are basic measures of the number of vertices, edges and maximum vertex degree within a graph (see Table 4.4). The longest geodesic path between any two vertices in a graph is used as the graph *diameter*, and together with *size*, *degree* and *order*, provide an elementary measure of graph size and complexity. Similarly, *girth* is an indication of the smallest non-trivial cycle in a graph, and may relate to function and robustness qualities.

Term	Meaning
Order	The number of vertices in a graph.
Size	The number of edges in a graph.
Degree	Maximum degree of any vertex present within the graph. See also the degree of a vertex.
Diameter	The number of edges in the longest geodesic path between two vertices *or* the average geodesic path length.
Girth	The length of the shortest cycle within the graph. If there are no cycles, the girth is considered infinite.

Table 4.4: Basic graph properties

Most graphs of interest are *connected* in that every vertex has at least one edge connection, and so the entire set of vertices in the graph are connected. A *disconnected* graph, containing a set of isolated graphs, is typically of interest because of the transition to or from a connected to a disconnected state – processes that occur when networks break down or suffer from attacks. The transition event from a connected to a disconnected state is used as an indication of network *robustness*.

If each vertex in a graph has a direct connection to every other vertex, the graph is *complete*. Such topologies are rare in the real-world because the cost of making all possible connections is, by definition, *maximal*. Although maximally connected topologies provide the highest degree of connectivity, they are likely to be far from "ideal" with respect to the cost of graph construction.

Multigraphs, with the potential for multiple connections between vertices, can be considered "super-maximally" connected graphs. The addition of multiple links in real-world networks is common in order to add *redundancy*, robustness and additional capacity to strategic network sections. A prime example is the use of multiple trans-Atlantic telecommunication cables, as well as satellite links, to provide multiple connections between continents, increasing robustness and capacity.

A graph with weighted edges is called a *weighted graph* and similarly the use of directed edges results in a *directed graph*. Most real-world networks contain properties that relate well to the edge qualities of direction and weight. See Table 4.5 for a list of terminology for different graph types.

Graph Type	Meaning
Complete	A complete graph with n vertices (denoted K_n) is a graph for which each vertex is connected to all other vertices directly (with one edge between every pair of vertices).
Connected	A graph is connected if there is a path (chain) connecting every pair of vertices, otherwise it is called *disconnected* and the graph is composed of discrete subgraph components.
Disconnected	There is some isolated (discrete) subcomponent not connected to the entire graph. See *connected graph*.
Directed	Also known as a *digraph* or an *oriented graph*, is a graph in which all edges are directed. (Directed edges only allow function in one direction.)
Undirected	Also known as an *unoriented* graph, an undirected graph is a graph in which all edges are without direction.
Weighted	Edges of the graph are weighted. (Weighted edges have an associated 'weight' value that influences their connection function.)
Topological	An unweighted or unity weighted graph that is described only by the topology. Weighted graphs can be treated this way to isolate the topology for investigation.
Multigraph	A graph with multiple edges between vertices.
Planar	A graph that can be drawn on a two dimensional plane with no edges 'crossing' ('overlapping') each other. Planar graphs can be drawn in non-planar form, but the "planar" quality is topological and so independent of a specific presentation.
Regular	A graph in which every vertex has the same degree.
Random	A graph for which the properties such as vertices, edges and connections are determined in a random way. There are many random graph models.
Isomorphic	Two or more graphs are isomorphic if they have equivalent topologies (i.e. the same set of edges and vertices), but are drawn differently. (See also Figure 4.1)

Table 4.5: Terminology used to describe graph types

4.3 Visual Representation

4.3.1 Introduction

There are several forms of visual representation that can be used for network topology, the most common and traditional being a two dimensional plane. When a graph can be represented in a two dimensional (2D) plain without the need for edges to cross-over, the topology is known as *planar*. Note that a graph may be planar due to its topology (regardless of how it is represented), but there is no easy proof of the planar quality for large (complex) networks. Rather it is usually easier to identify nonplanar subgraphs that exclude the possibility of an entire graph having a planar form.

Graphs that are isomorphic (see Figure 4.1) have equivalent topologies but may be represented in different visual forms. Importantly, the statistical properties of isomorphic

graphs will be equivalent as the properties are a direct result of topology, not visual representation.

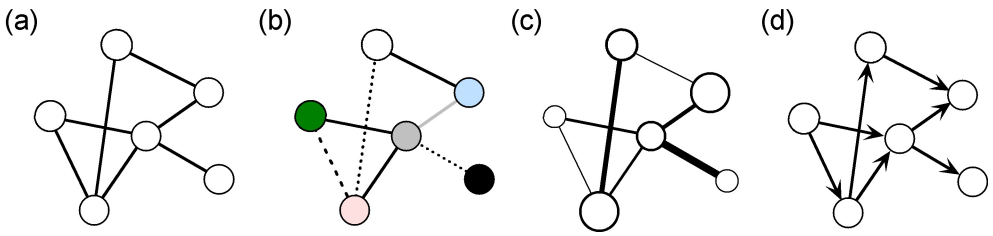


Figure 4.2: Examples of various graph types and drawings. The first three examples are undirected graphs. Graph (a) has uniform edge and vertex types, (b) has different edge and vertex types, (c) varied edge and vertex weights. Graph (d) is directed (digraph) with directed edges and uniform nodes.

Real-world networks can be composed of homogeneous or heterogeneous components, and so visual representation of networks may require different visual forms for edges and vertices. The use of *weighted edges* between nodes is a common way to represent properties of link strength or capacity. See Figure 4.2 for examples of graph type representations. Note that variations in visual representation can apply to both vertex and edge qualities. The example also demonstrates a graph topology that is planar but not represented in a planar form.

Effective graph drawing techniques that can organise the location of vertices and edges, particularly in a manner that illuminates topological relationships, are an extremely valuable tool. Such techniques can allow the human faculties of visual and cognitive processing to be utilised effectively. For example, the presentation of graph information can be used to identify critical components, important nodes, neighbourhoods, clustering and the utilisation of resources by processes. The additional use of colour, line weight, labels and other iconic notation further extend the accessibility of network property information to the human eye.

Three dimensional (3D) graph representations are an interesting and useful extension to traditional 2D representations, especially if such models allow users to interact and rotate, scale and reposition a graph and its elements.

4.3.2 Regular and Random Structures

There are several formal graph topology descriptions that are related directly to methods of graph drawing – the process of representation. Regular structures, such as *linear*, *lattice* (*grid*), *tree* (*hierarchical*) and *ring* structures, provide uniform topological features that can be very useful in constructing graph topologies with particular connectivity qualities. Similarly, a *random graph* can be constructed by using random methods to determine the placement of edges to connect nodes. (See Section 4.5.4 for more detail on random models.)

An interesting and important question, especially for the development of large networks, is "what happens to statistical properties when a topology is scaled up?". If properties remain statistically consistent regardless of the scaled size, a network is considered

"scale free" (the properties are free from size related factors). However most regular, hierarchical and random topologies are not scale free, and yet many natural networks do exhibit interesting scale free properties, and so this has been a strong incentive for many researchers to try and discover mechanisms capable of producing scale-free topology. Some of these graph modelling methods are discussed further in Section 4.5.

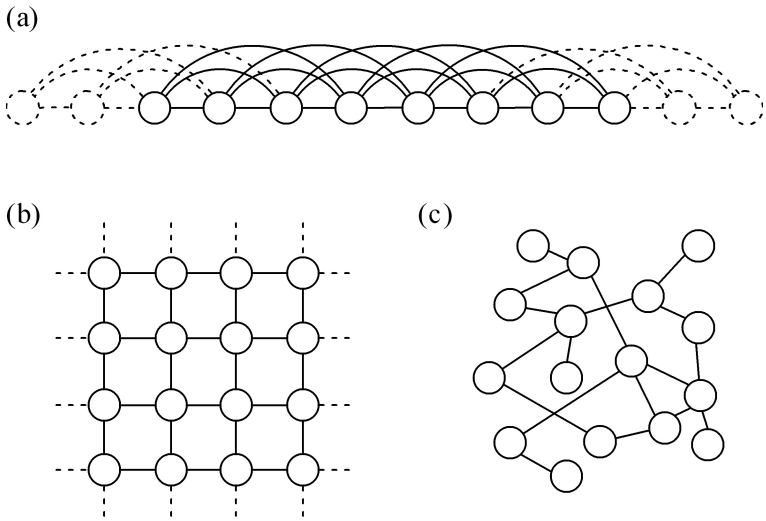


Figure 4.3: Examples of regular and random graph types. (a) shows a linear regular lattice, and if the ends were connected would represent a circular regular lattice. (b) is a standard regular grid lattice, and (c) is a simple random topology and layout.

4.3.3 Graph Drawing

The field of graph drawing has a long history and continues to be actively developed. Perhaps one of the reasons for this is that the work has obvious practical and visually *aesthetic* benefits, where applications are possible over a broad range of disciplines. Di Battista et al. have contributed annotated bibliographies [83] and a book [84] that are excellent references to significant publications in this field.

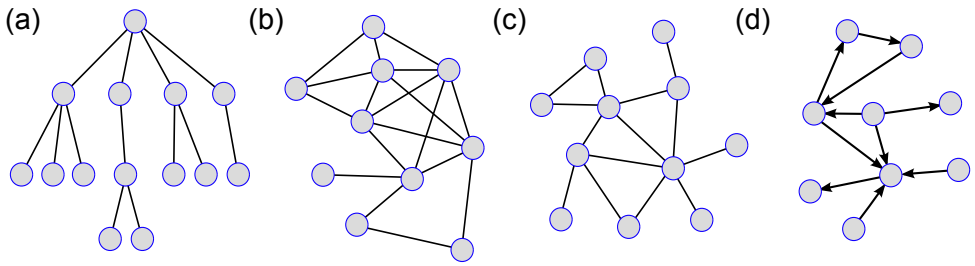


Figure 4.4: Examples of graphs organised by drawing groups. (a) Tree graphs have a definite root node and layered hierarchy, (b) general graphs are not restrained or directed, (c) planar graphs always have a possible planar representation and (d) directed graphs include the notion of direction (flow) through the topology.

Graph drawing algorithms can be divided into several groups based on the type of graph structure each represents: *trees*, *general graphs*, *planar graphs* and *directed graphs* [83].

Figure 4.4 presents examples of each group. The goal of many drawing algorithms is based on a set of aesthetic qualities, such as symmetry, avoiding edge crossing or bending, using minimal and uniform edges, and uniform distribution of nodes. Interestingly, many drawing or layout techniques are based on simulations such as the *spring embedder* system or *simulated annealing*; both are well known to the field of optimisation and search. The nature of each graph drawing group can be summarised as follows:

Trees are well suited for organisational charts and hierarchies such as family genealogies. Levels indicate equivalence and notions of *root*, *parent*, *children* and *sibling* nodes work well. However, trees do not need to be *rooted* and there exist algorithms for drawing *free trees*, including radial forms.

General Graphs are the most relevant to this thesis, as are they are used to represent general undirected networks. The drawing of "general" graphs include straight, curved or poly-line edges. General graphs can also be useful to search for and use a planarised form (if it exists) for drawing.

Planar Graphs are by definition graphs that can be drawn using a planar representation. Planar representations are aesthetically desirable, but can also be of significant theoretical and practical value (such as electronic circuit layout). Of course, planar graphs may be represented in nonplanar forms, and there are algorithms for both specifically testing for planar graphs and simulation based searches for planar forms. Interestingly, it has been shown that every planar graph can be drawn in a straight line form, although curved edges do allow for a more compact form if required.

Directed Graphs include and specifically consider the direction of edges in representations. Typically an effort is made to have all edges "flow" in the same direction, particularly for acyclic[6] directed graphs which are used to represent hierarchies (down for trees, right for processes etc).

The *spring embedder* layout algorithm [93] and other similar force-based layout algorithms deserve special mention as they are one of the most influential simulation based methods. They work well for n dimensional systems [122], not just those presented visually in two or three dimensions. Spring embedder methods are also known as "force-directed placement" algorithms.

Spring embedder systems are essentially a particle system model where graph nodes are considered particles that repel each other, and vertices are treated as physical springs causing attraction between connected nodes proportional to distance. The algorithm usually starts with a random placement (or addition) of nodes and then uses an iterative process that minimises the total energy of the system.

Iterative minimisation is stopped when either the system becomes stable, or a practical iteration limit has been reached. Additional nodes and edges create an exponential increase in interactions, and thus increases the computational cost for large systems. As there are

[6]The *acyclic* quality means that there are no cycles (or loops) in the graph. See Table 4.3 for path related terminology.

both stochastic and random elements to the algorithms, successive simulations may be used to find a desirable configuration.

A similar energy minimisation approach has also been suggested by Davidson and Harel [68, 69] where an aesthetic cost measure for "beauty" is defined to minimise crossings and take into account the closeness of vertices. The overall energy is then minimised (and "beauty" maximised) using a simulated annealing algorithm.

Fruchterman and Reingold developed a force directed placement algorithm [122] for layouts (suitable for simple 2D or n-dimensional spaces, and other methods for large grid-based networks). Another popular method is that by Kamada and Kawai for "drawing general undirected graphs", specifically connected graphs such as those grown by the Barabaási and Albert (BA) model (presented in Section 4.5.7). Reingold and Tilford published a method for "tidier" drawing of trees [292] which is useful for hierarchical systems.

The drawing of graphs is not the focus of the work in this thesis, however it is a very useful tool for presenting and understanding the structures that exist or evolve within EC. It is also useful to consider the connection between minimisation based layout processes, evolutionary search, and the ecological nature of biological network structures. This can also have a meaningful influence as a biological metaphor for problem solving; the physical layout position of nodes may have a direct meaningful relationship to the function or interaction of network components.

In a related manner, the function or performance value of vertices within a network can be mapped across a sample of input values, and by inspection the observer can gain insight into behaviour or patterns of network function that might otherwise be too complex to describe or analyse. Such a technique was used in [372] to visualise the function of internal components of Artificial Neural Networks, in particular the contribution of complex weight components on input signals as used in the Micronet architecture developed by Murray [250, 249, 248].

4.3.4 Further Resources

As mentioned, the annotated works of Di Battisa *et al.* [83, 84] are excellent resources into the field of graph drawing, as well as [94] and the more recent book chapter by Tamassia [336].

The online website home of the International Symposiums on Graph Drawing[7] contains links to the individual symposium websites, and links to other useful websites, literature and data resources.

The Graph Drawing E-print Archive (GDEA) online website[8] is an excellent online resource for research materials on this topic.

There are many software products, libraries, toolkits and packages available to assist researchers in the use of graph drawing algorithms. Of note is the GraphViz open source project by AT&T Research[9], the Pajek (Package for Large Network Analysis) tool

[7] http://graphdrawing.org/
[8] http://gdea.informatik.uni-koeln.de/
[9] http://www.graphviz.org/

made freely available (closed source) by Vladimir Batagelj[10], the JGraph open source graph component for Java[11], and the open source JUNG (Java Universal Network/Graph Framework) library[12].

The igraph project[13] is a library by Gábor Csárdi that is well suited to creating, testing and analysing simple, but large, complex networks. The igraph software can be used as a library within the R-project[14] statistical software package [284, 171], as a scriptable Python library, or as a C library. igraph has also been described in a conference paper [61].

4.4 Measurements and Properties

4.4.1 Introduction

It has been observed many times that real-world networks are not the same as simple random models, and so this naturally leads the the question of what mechanisms are responsible for the non-random structures. Further, the real-world observed systems are not simple or regular, but exhibit interesting qualities. There is a need to be able to measure and qualify both real-world and simulated networks. This then enables the development of models to emulate the observed characteristics of real-world networks, and suggest mechanisms for their development. An understanding of the mechanisms enables the exploitation of this knowledge for other purposes, such as simulated evolutionary environments.

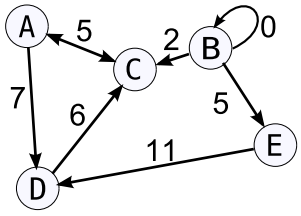
There are several statistical properties of networks that we can measure to gain insight into structure and function. Specifically, this section describes the ideas and formula for determining the *degree* (including *distribution*, *correlation* and *power-law* components), *clustering coefficient*, *motifs*, *characteristic path length*, *connectivity length*, *harmonic mean*, *global efficiency*, *local efficiency* and a definition of *cost*. We also look at other related measures.

Starting with some basic concepts, a graph G is composed of a set N of n nodes and a set M of m edges. G then is defined as $G(N, M)$.

A graph G can be represented as an *adjacency matrix*[15] $\mathbf{B} = \{a_{ij}\}$ which is $N \times N$ in size and where each a_{ij} value represents a connection between two nodes. For a simple edge, this is 1 for a connection and 0 for no connection. If we consider a directed graph (with directed edges) $a_{ij} \neq a_{ji}$, and if we are representing an undirected graph $a_{ij} = a_{ji}$ creating symmetry. This means that for an undirected graph the total number of edges is half the total possible for a directed graph and that the adjacency matrix is symmetric. Usually, graph nodes do not have "self" connections, so $a_{ij} = 0$ where $j = i$.

If we need to describe a weighted graph, it may be convenient to use both the adjacency matrix $\mathbf{B}$ to represent the topology of connections, and an additional *distance matrix* $\{d_{ij}\}$. Where there is a direct connection between nodes, d_{ij} is the weight value of the edge, otherwise it can be defined as the minimum weighted path needed to get from node

[10] http://vlado.fmf.uni-lj.si/pub/networks/pajek/
[11] http://www.jgraph.com/
[12] http://jung.sourceforge.net/
[13] http://cneurocvs.rmki.kfki.hu/igraph
[14] http://www.R-project.org
[15] Also known as a "connection matrix"



(a) simple directed graph

	A	B	C	D	E
A	0	0	1	1	0
B	0	1	1	0	1
C	1	0	0	0	0
D	0	0	1	0	0
E	0	0	1	1	0

(b) adjacency matrix **B**

	A	B	C	D	E
A	0	-	5	7	0
B	7	0	2	14	5
C	5	-	0	-	-
D	11	-	6	0	-
E	22	-	17	11	0

(c) shortest path matrix

Figure 4.5: A simple directed graph represented by an adjacency matrix and a distance matrix. Row headings indicate the "from" vertex, and column headings the "destination" vertex, although this is arbitrary. In this case directed edges are represented; for an undirected graph the adjacency matrix is mirrored down the diagonal axis. We also see an example of a self-linked vertex on the B node. The distance matrix contains the shortest path distance between any two nodes, if a path exists at all and where there are multiple indirect connections. It is only a directed or disconnected graph that has missing distance matrix entries (indicated with "-"). A connected undirected graph will have a complete set of valid path costs in the distance matrix.

i to j. A cache of minimum weighted path cost is very useful for calculations that need to compare path costs. See Figure 4.5 for an example of both adjacency and distant (shortest path) matrix representations.

4.4.2 Degree, Distribution and Correlation

Degree Distribution

The *degree* k_i of node i, also known as the "connectivity", is the number of edges incident to node i and so it is also the number of neighbours[16] to node i. We can determine the minimum degree k_{min} and maximum degree k_{max} that exist in a network, and the *mean degree* $\langle k \rangle$ as an overall measure of edge density. The degree k_i for node i can easily be determined from the adjacency matrix:

$$k_i = \left(\mathbf{B}^2\right)_{ii} = \sum_{j=1}^{m} \mathbf{B}_{ij} \tag{4.1}$$

When considering an entire graph, the *degree distribution* is more meaningful than the degree of single nodes. The degree distribution is the "spread" of connections, and it can be characterised by a *distribution function* $P(k)$ that gives the probability that a randomly selected node has exactly k edges connected to it.

For random graph model, such as the ER model discussed later in Section 4.5.4, the degree distribution is Poisson such that

$$P(k) = e^{-\langle k \rangle} \langle k \rangle^k / k! \tag{4.2}$$

where the mean degree, $\langle k \rangle$, in terms of nodes or a probability distribution p, is given by:

[16] This assumes that multiple edges are not allowed between nodes, and so the graph is not a multigraph. Many real-world networks, with multiple alternative connections between nodes, do not map easily to such a restricted model [259].

$$\langle k \rangle = 2n/N = p(N-1) \approx pN \tag{4.3}$$

It has been noted [9] that in comparison to random networks, most complex and real-world networks have skewed degree distributions following either a power-law or power-law tail distribution of the form

$$P(k) \sim k^{-\alpha} \tag{4.4}$$

where α is the power-law degree distribution exponent value. The mean degree will then take the form of

$$\langle k \rangle = k_{max}^{2-\alpha} \tag{4.5}$$

where the maximum degree will be $k_{max} < N$.

It is reasonably easy to see power-law and exponential distributions by plotting cumulative distributions. A power-law distribution, plotted on a log-log graph, present a straight line, while a straight line on a log-linear (semi-logarithmic) scale indicates exponential relationships.

Because many real-world systems showed a tendency to contain power-law tails, and because this was reasonably unexpected by researchers, power-law distributions were considered an almost "universal" quality of complex systems. (See the reviews of [257, 28].) Many studies then focused on the search for these scale-free networks, and the construction of models (some overly contrived [188]) to generate networks with degree distributions that followed power-law tails.

As Evelyn Keller has pointed out though [188], there are reasons to believe that scale-free architectures are not "universal", nor a common significant indicator of a complex system. Rather, scale-free architectures are particular and should be expected based on the understanding of constraints in the systems in which they occur. This would ideally result in a more grounded approach to scale-free network research.

Degree Correlation Coefficient

Degree-degree correlation $P(k, k')$ is a way of describing that the degrees of two connected nodes are not independent. Put more simply, the number of connections a node has is directly related to the number of connections neighbours have. Erdös-Rényi (ER) random graphs (see Section 4.5.4) do not have any degree-degree correlation, but many real-world networks do. In most cases it is believe this is due to the processes responsible for edge formation.

It is convenient to define degree-degree correlation as the average degree of nearest neighbours to node i, where G_i is the subgraph of nearest neighbours to i:

$$k_{nn,i} = \frac{1}{k_i} \sum_{j \in G_i} k_j \tag{4.6}$$

The average degree of nearest neighbours [273] with degree k (with respect to all other k' degrees) is then

$$\overline{k}_{nn}(k) = \sum_{k'} k' P(k'|k) \tag{4.7}$$

When the degree correlation coefficient and $\overline{k}_{nn}(k)$ both increase as a function of k, a network is said to exhibit *assortative mixing*. Similarly, when the degree correlation coefficient increases as a function of k and $\overline{k}_{nn}(k)$ decreases as a function of k, the network is said to exhibit *disassortative mixing*. An interesting result is that social networks are known to be assortatively mixed [254].

Assortativity Coefficient

Newman has defined a method for a single degree correlation value r by calculating the Pearson correlation coefficient of the degrees at either ends of an edge, which results in a single value that is positive for assortative and negative for disassortative mixing [254, 256]. The full definition is not shown here, but values calculated with this method are shown in Table 4.8.

E_{ij} is the number of edges that connect vertices of type i and j, where $i, j = 1 \ldots N$. Now let $\mathbf{E}$ be a matrix that contains the set of E_{ij} elements. The mixing matrix $\mathbf{E}$ is then normalised by the sum of elements $\|\mathbf{E}\|$ to create a normalised mixing matrix $\mathbf{e}$,

$$\mathbf{e} = \frac{\mathbf{E}}{\|\mathbf{E}\|} \tag{4.8}$$

so that now an assortative coefficient can be defined as

$$r = \frac{\mathrm{Tr}(\mathbf{e}) - \|\mathbf{e^2}\|}{1 - \|\mathbf{e^2}\|} \tag{4.9}$$

where Tr is the trace (or sum of the diagonal elements) of $\mathbf{e}$.

4.4.3 Clustering Coefficient

The concept of clustering, also known as *transitivity* or *network density*, is perhaps best described in social networking terms, where two of your friends are also likely to be friends to each other. This might also be thought of as the *cliquishness* of a node within a network.

Clustering is interesting in complex systems because the amount of clustering is frequently observed to be much greater than clustering for a random graph with the equivalent number of nodes and edges. It seems that *clustering and grouping* are an important part of complex structures, and are defining components of the "small-world" network model. (See the small-world model in Section 4.5.5.)

For a graph G we can calculate a clustering coefficient $C(G)$. We begin with the subgraph G_i for node i, composed of first neighbours for node i. If node i has k_i neighbours, then G_i will contain k_i nodes and can have, at most, $k_i(k_i - 1)/2$ edges. So, the local clustering coefficient C_i for node i is

$$C_i = \frac{number\ of\ edges\ in\ G_i}{k_i\,(k_i - 1)/2} \tag{4.10}$$

This essentially yields a ratio of the number of connected neighbours *versus* the total possible connections in subgraph G_i. Note that *transitivity* is a specifically different definition: the specific ratio of the number of triangles with respect to the number of connected triples (three node paths)[17].

With the local clustering coefficient C_i we can then determine, for the entire graph G, the mean of all C_i ratios giving an overall measure of the clustering $C(G)$. This can be expressed as:

$$C\,(G) = \frac{1}{N} \sum_{i \in G} C_i \tag{4.11}$$

This thesis uses the local clustering coefficient C as defined by Watts and Strogatz [360], with a preference for the notations used by Latora and Marchiori [215]. There are other formulas for clusters, some of them slightly easier to calculate, and some which result in different numerical values. The formula presented here is perhaps the most commonly used and, as Newman has pointed out, it is important that the clustering measure used is clearly defined to allow comparison with other work [257].

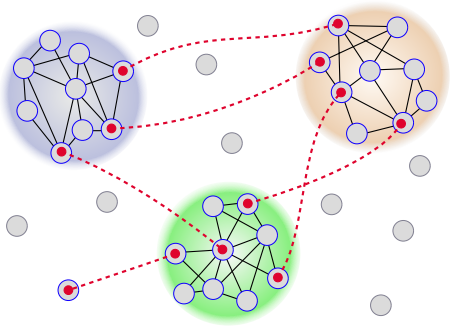


Figure 4.6: Community structure example, where communities are marked by the circle regions, and characterised by a greater number of connections between nodes in each community group. The idea of community can be formalised with either density-based metrics, or similar localised cluster measures. Note that in this example, some nodes are still connected but not part of any community, and real world systems can contain isolated nodes.

It should be noted that the term "clustering" is used for both specific and well defined measures, such as the one presented in Equation (4.11), as well the conceptual notion of, say, general community structure. Such community structures are known to have a higher density of links within particular "groups", though the measures described here do not account for the full range of possible complexities. See Figure 4.6 for a visual example of this idea. Community structure is a difficult but valuable aspect to measure and classify in real complex systems. *Motifs* are a continuation of this idea and are presented in Section 4.4.4.

[17] In other words, of all the unique three node paths that exist within a graph, what is the ratio that are triangles, and hence represents tight clusters of closely connected nodes?

There are interesting open questions as to the value of higher-order degree clustering coefficients, and also low-degree "reciprocal" loops in directed graphs between nodes.

4.4.4 Motifs

Network *motifs* are patterns of interconnection within graphs that occur at a frequency higher than expected for a similar randomised graph with the same degree characteristics. The identification of motifs has been motivated by Uri Alon and colleagues who realised that motifs can be used to identify and classify networks and their functions [315, 235, 225, 234, 186, 10]. Their work has mainly involved the identification of recurring small n sized motifs (in the order of 3 to 5 nodes) in biological systems, but has also been extended to other systems such as information processing networks.

An example of all thirteen possible motif forms for a three node directed (connected) graph are shown in Figure 4.7. Although it can be computationally expensive to look for motifs, there have been very strong results even with small sized graphs of only three and four nodes. This suggests it is well worth the computational resources that can be required.

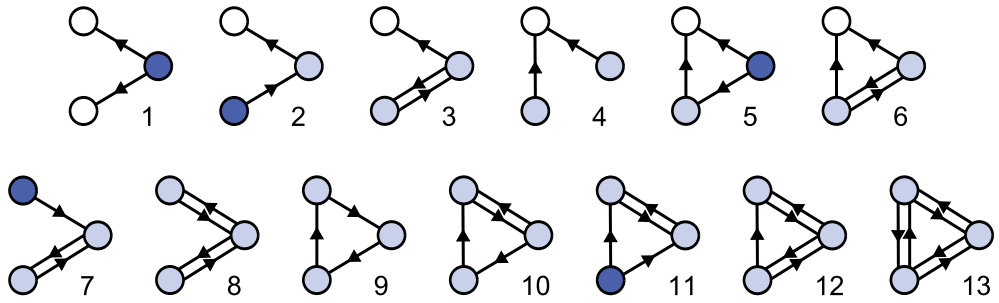


Figure 4.7: The thirteen possible motif forms for a three vertex directed and connected graph. Edge direction has been indicated as well as vertex colour to give a general indication of flow and help identify individual motif forms. The identification of motif numbers in large networks can be directly connected to network formation and function. However, permutation calculation and identification is computationally demanding.

In order to quantify the statistical significance of a motif type M, a count n_M is used to represent the number of occurrences of M in G. A comparison is then made to a random graph G^{rand} with the same degree characteristics as G. A *Z-score* is calculated as a measure of the significance of M in G, expressed as:

$$Z_M = \frac{n_M - \left\langle n_M^{\text{rand}} \right\rangle}{\sigma_{n_M}^{\text{rand}}} \tag{4.12}$$

where n_M^{rand} represents the occurrences of M in the random graph, $\langle n_M^{\text{rand}} \rangle$ is the mean, and $\sigma_{n_M}^{\text{rand}}$ the standard deviation of the motif M occurring in an ensemble of random networks.

Results across different domains have shown that significant motif forms are different. For example, gene regulatory network motifs are different from those of food web motifs or

World Wide Web motifs. Further, the existence of similarities between networks suggest similarities in underlying processes responsible for the formation of the network. One interesting observation is that networks that support information processing seem to be very different to those networks that transfer energy [235, 10].

The discovery of network motif characteristics may be a universal way to fingerprint and categorise networks by the number and type of motifs they contain. As stated by Ron Milo and colleagues, the result "...suggests that motifs can define broad classes of networks, each with specific types of elementary structures" [235]. Hence, motif identification may be very important to many fields including biology, biochemistry, neurobiology, ecology, engineering and many others.

Motif detection has been presented by Wernicke and Rasche [363], and the *Atlas of Graphs* by Read and Wilson contains a large collection of known motif forms [288] and is relevant to the implementation of motif identification algorithms.

4.4.5 Characteristic Path Length

The characteristic path length L is a global measure of the typical separation between any two nodes within a graph G [360]. It is also reported as the "mean vertex-vertex distance" or the "mean path length" in some research.

To generate this measure we must first be able to find the geodesic (minimum) path d_{ij} (edges traveled) between each pair of nodes within the graph. In the example of an unweighted graph, $d_{ij} = 1$ when there exists a direct connection between nodes i and j, and $d_{ij} \geq 1$ provided the graph is *connected*. If all d_{ij} are finite values, then so is L.

In order to determine the entire set of values in $\{d_{ij}\}$ a standard breadth-first search [58] works well, although this can be optimised for specific applications. Weighted networks need to account for edge cost, and so should utilise a best-first heuristic search approach. The characteristic path length L for the entire graph G of n nodes is then:

$$L(G) = \frac{1}{n(n-1)} \sum_{\substack{i,j \in G \\ i \neq j}} d_{ij} \tag{4.13}$$

As listed in Table 4.4 the *diameter* of a graph can be defined in multiple ways: the largest, average or even the minimum geodesic path. The vertex with the lowest total mean path length to all other vertices in the graph can naturally be defined as the *centre* vertex of the graph.

Girvan and Newman introduced a similar quality of *betweenness* as a way to find the central vertex of a graph [130]. A betweenness value is defined as the number of times a vertex participates in the shortest path between all pairs of nodes. Thus, the node with the highest betweenness value is "central" to the graph topology and function. The betweenness value can be easily calculated with a similar method to that used to calculate L. The terms graph "centre" and "betweenness" are summarised in Table 4.6

Graph Term	Meaning
Centre	The vertex with the lowest total mean path length to all other nodes in the graph. A centre vertex is important because the cost to all other nodes will be low, however it does not indicate the utility of the vertex (how often it will be *used by* other vertices).
Betweenness	The vertex that participates the most number of times in the shortest path between all vertex pairs of the graph. A vertex is important because it participates in important shortest paths, not because of the distance or weighting of the paths it is part of. This then is a measure of the vertex utility to other vertices.

Table 4.6: Graph terms used to describe the "central" vertex

4.4.6 Global and Local Efficiency

If G is a disconnected graph (as many real-world networks may become when they degrade), then there exists a d_{ij} that is $+\infty$, and so L is thus ill-defined. Although for many networks this is not a problem, when considering real-world networks, directed networks with disconnected components or other systems that are represented as a disconnected graph, this severely limits the application of L.

It is for exactly this reason that Latora and Marchiori [215, 217] proposed the concepts of *global efficiency* and *local efficiency*, of which both are well-defined even when there exists disconnected components (such that $d_{ij} = +\infty$) in the network. The concept of efficiency is related strongly to the idea that the role of the network is communication between nodes, and the shorter the communication pathways, the more efficient the network. A normalised value to represent efficiency should be in the range of [0,1], where 0 is no communication (very poor efficiency) and 1 for a fully connected graph where every node has a direct connection to every other node (the best possible efficiency).

The measure of global efficiency E_{glob} is defined as the sum of the inverse path lengths between each pair of nodes $\varepsilon_{ij} = \frac{1}{d_{ij}}$. In so doing, when $d_{ij} = +\infty$, $\varepsilon_{ij} = 0$ and so contributes nothing to the overall efficiency. We normalise the sum using an ideal graph G_{ideal} that contains all $n(n-1)/2$ possible edges.

$$E_{glob}(G) = \frac{\sum\limits_{i \neq j \in G} \varepsilon_{ij}}{n(n-1)} = \frac{1}{n(n-1)} \sum_{i \neq j \in G} \frac{1}{d_{ij}} \tag{4.14}$$

In this form we are assuming unweighted edges (or a unity weight of 1 for each edge), however this can be extended to weighted graphs and the sum of inverse distances can be normalised using an "ideal" weighted graph G_{ideal}, as needed for the particular network in question [216]. This is a lot simpler than the modification needed to apply the mean path length L to a weighted graph.

Newman points out [257] that the use of harmonic mean, or "reciprocal of all the reciprocals" type of efficiency measure, is perhaps a more "satisfactory approach" to defining L, and although it has only occasionally been adopted "should perhaps be used more often".

This measure of global efficiency can also be localised to subgraphs. Consider the subgraph G_i about node i, which includes only the neighbours of node i (that are directly

connected to i) and will contain at most $k_i(k_i-1)/2$ edges. Summing and normalising the efficiency of all subgraphs of G gives a measure of the local efficiency E_{loc} within G:

$$E_{loc}(G) = \frac{1}{n}\sum_{i\in G} E(G_i) \tag{4.15}$$

where

$$E(G_i) = \frac{1}{k_i(k_i-1)}\sum_{\substack{l,m\in G_i \\ l\neq m}} \frac{1}{d'_{lm}} \tag{4.16}$$

Note that d'_{lm} is the shortest path length between nodes l and m within the subgraph G_i, and that for a unity weighted graph[18] E_{loc} is normalised.

The measure E_{loc} is similar to the clustering coefficient C, however it is not a direct triangle to triple ratio, nor does it indicate the presence of hierarchy or other specific motif patterns.

4.4.7 Cost

In real-world networks, both engineered and naturally occurring, each edge and node is likely to have a cost, both in construction or in use. Therefore, there are economic and competitive advantages to minimising the cost of a network while maximising the functional performance. As will be shown in Section 4.4.9 many real networks have been found to be low cost while supporting function robustly.

In order to allow us to measure and compare networks, we also need to be able to measure relative cost. We would expect that the efficiency (performance) of a network would increase as the number of edges increases. However, not all edges play the same role within real networks, so efficiency and cost are not so simply related. Indeed, the connection of edges within a graph to critical hub nodes can dramatically improve the efficiency of a network without a large increase in cost.

Using a simple graph G of n vertices and m edges we can define [217, 60] the cost $Cost(G)$ as:

$$Cost(G) = \frac{2m}{n(n-1)} \tag{4.17}$$

This is the number of edges that exist in G normalised by the total number that could possibly exist, and so we have a value in the range [0,1].

Again, it is useful to apply a similar cost measure to weighted networks, and by using a *cost evaluator* function γ we can simply use the distance d_{ij} or weight value of an edge a_{ij} as its cost [216]:

$$Cost(G) = \frac{\sum_{i\neq j\in G} a_{ij}\gamma(d_{ij})}{\sum_{i\neq j\in G}\gamma(d_{ij})} \tag{4.18}$$

[18] Unity weighted graphs, because they describe the topology only, are also known as *topological graphs*.

4.4.8 Other Measures and Properties

Proximity Ratio

Watts and Strogatz [360] used the measures of C and L as a way of characterising small-world networks that are neither random nor like a completely regular system. (See Section 4.5.5 for more on the small-world model.) However, the property of "small-worldness" is qualitative rather than quantitative and requires two measured properties as indicators. Hence, others like Walsh [353] and Latora [215] developed single measures to give a qualitative result.

Walsh suggested the use of a "proximity ratio" μ which is the ratio of C/L normalised by C_{rand}/L_{rand}, where C_{rand} and L_{rand} are taken from an ER style random graph (see Section 4.5.4) with the same number of nodes n and edges m as the graph being considered [353]. This gives a single numerical value, and for systems that are equivalent to a random graph where μ is unity, while for small-world graphs $\mu \gg 1$.

Entangled Networks

It has been suggested by Luca Donetti and colleagues [87, 88] that there exists a family of graphs, which they have named *entangled*, that exhibit a highly "interwoven" topology and contain highly homogeneous structure with respect to degree, node distance, betweenness and loop distance which are all very narrow in value.

Specifically, entangled networks have been characterised as having:

- short average path-length distances;
- small diameter;
- low clustering coefficient (due to the presence of large average shortest loops); and,
- poor modularity.[19]

Conceptually, one might think of an entangled network in a similar manner to the small-world linear-lattice model[20] (discussed in Section 4.5.5), in that a regular topology and long-range connections exist, however the lack of community structure or high clustering and large loops means that the neighbourhoods are sparsely connected. Most connections in the network extend outside neighbourhood regions, and are interwoven or mixed in a homogeneous way that ensures clustering does not occur.

Research regarding the performance of entangled networks, for data communication processes, has shown that while entangled networks have good connectivity and flow performance, they also have excellent robustness against error and attack, and support efficient communication and search processes. It is because of this that Donetti and peers

[19] Modularity is often a measure of "community" structure within a network, and hence poor modularity in this case is an indicator of the absence of any sort of community-like structures.

[20] A linear lattice graph has nodes arranged along a single dimension with regular neighbourhood connections. Linear-lattice models are often defined with a connected boundary condition which creates a "ring" topology. For the classic small-world model additional random connections are also added.

have conjectured that the terms of "optimal" and "near-optimal" should apply to entangled networks.

An open research problem is the development of models to create artificial entangled networks that not only exhibit these distinctive qualities, but that also scale accurately. Methods based on simulated annealing have been used with some success, using a fixed number of nodes and a fixed average connectivity $\langle k \rangle$, while also exhibiting the highest degree of *synchronisability* possible. Synchronisability, in this case, is described as a behaviour of individual dynamical processes at the vertex level, and is closely related to the ability of information to be transmitted across the network, and hence has a dependence on the network topology.

One measure of synchronisability, applied to small-world networks [30], uses a spectra-based technique for the identification of stable synchronised states from eigenvalues derived from the connectivity matrix for a graph.

Bipartite Model

A bipartite graph model is useful to represent systems with two types of nodes and edges that exist only between unlike nodes. Bipartite graphs can be used to model social *affiliation* networks, and are particularly interesting in the study of connections between people acting as board members and individual companies, or the example of collaborations between scientists, or the common films of actors. It is also a way to represent population evolution based on sexual models, where gender is simply a node quality or location occupancy in an ecological model.

Topologically, the distinction of two node types allows the graph to be treated as essentially two different graphs, with two degree distributions each related to the number of nodes of each type. Graph degree distribution generation functions, that can be used to describe the formation and edge distribution of such networks, can then be based directly on the subgraph of a specific node type.

One common method of studying bipartite graphs is to project one set of vertices on the other to give a "single mode" (or "one mode") projection. It can also be useful to separate nodes into layers, where the interaction between layers can be clustered or grouped to identify interlocking relationships. An example of this is shown in Figure 4.8. Such methods have been used to identify "interlocks" of company board members.

The bipartite model can be extended by relaxing the constraint of edge formation between differing node types, and allowing more node types to exist; both changes are useful for modelling real-world networks. Although the additional complexities of connections between node types can make analysis more difficult, it is still very useful to isolate subgraphs based on node type, and to model or measure the degree distributions for each subgraph. This also has application to the study of ecological environments and in particular the formation and interaction of species.

A bipartite model is also directly related to measures of *assortativity* and *assortative mixing*, mentioned in Section 4.4.2 in connection to degree correlation. Assortative mixing, or *homophily* in sociological terms, is the behaviour of individuals to preferentially

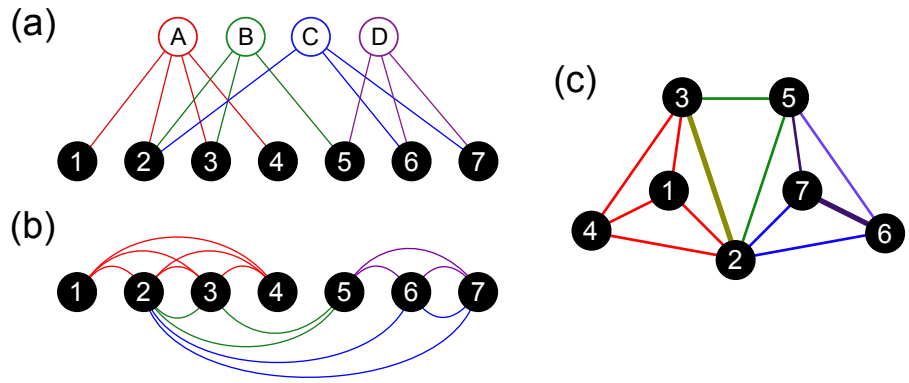


Figure 4.8: Bipartite graph example and its projection to a single mode. Bipartite graphs are represented (a) as two distinct layers of like nodes, with edges between layers only. Such graphs can be used to represent networks such as people and company boards or actors and common movies. In (a) white vertices (lettered A to D) represents people, and black nodes (numbered 1 to 7) represent the common association (such as movies or company boards). A single mode projection (b) starts by converting the common link (in this example people) into undirected edges (using colour to assist). It can be seen that in some cases there are multiple edges between vertices. A planar representation of the projection is shown in (c), using colour to help indicate association, and edge line weight (thickness) to indicate the stronger connection that results from multiple edges.

associate with other individuals that are similar. There is also an important connection to the process of mate selection, exhibited by some species, based on "assortative matching".

4.4.9 Real-World Examples

Now that we have looked at several of the measures used to classify network properties, let us consider some real-world examples. The data presented in Table 4.7 has been reported and analysed in various works by Latora and Marchiori [215, 217, 216]. Here the results are collected into a single table.

	n	m	L	C	E_{glob}	E_{glob}^{rand}	E_{loc}	E_{loc}^{rand}	$Cost$
C.elegans	282	2462	2.65	0.28	0.46	0.48	0.47	0.12	0.06
Film actors	277336	8721428	3.65	0.79	0.37	0.41	0.67	0.00026	0.0002
Internet	6474	12572	3.77	0.3	0.29	0.30	0.26	0.0005	0.006

Table 4.7: Examples of real-world network efficiency and cost. Note small characteristic path length L and high clustering coefficient C, and how this matches the high global E_{glob} and local E_{loc} efficiency respectively. n and m are the total number of nodes and edges respectively. E_{glob}^{rand} and E_{loc}^{rand} represent the global and local efficiency of a random network with the same resources. Note how $E_{loc} > E_{loc}^{rand}$ indicates the increased clustering. This data has been collected from work by Latora and colleagues [215, 217, 216].

We can see in Table 4.7 three different real-world networks: the neural network of the highly studied Caenorhabditis elegans (*C.elegans*), a database of film actors and films (bipartite), and a sample of internet (hardware) architecture. Note in each case we can see comparative global efficiency with respect to a random system, however with greater

local efficiency than expected in a random model. The measures for C, L and $Cost$ are presented also.

Newman has also presented a collection of results in Table II of [257] divided into the groups of *social*, *information*, *technological* and *biological* that were mentioned in Section 4.1.2. A simplified reproduction of the results is shown in Table 4.8 to illustrate typical values.

Network	dir.	n	m	$\langle k \rangle$	L	α	C	r
Social								
film actors	No	449913	25516482	113.43	3.48	2.3	0.78	0.206
company directors	No	7673	55392	14.44	4.60	-	0.88	0.276
math coauthorship	No	253339	496489	3.92	7.57	-	0.34	0.120
student friendships	No	573	477	1.66	16.01	-	0.001	-0.029
Information								
WWW nd.edu	Yes	569504	1497135	5.55	11.27	2.1/2.4	0.29	-0.067
Roget's Thesaurus	Yes	1022	5103	4.99	4.87	-	0.15	0.157
Technological								
Internet	No	10697	31992	5.98	3.31	2.5	0.39	-0.189
software packages	Yes	1439	1723	1.20	2.42	1.6/1.4	0.082	-0.016
Peer-to-peer network	No	880	1296	1.47	4.28	2.1	0.011	-0.366
Biological								
metabolic network	No	765	3686	9.64	2.56	2.2	0.67	-0.240
protein interactions	No	2115	2240	2.12	6.80	2.4	0.071	-0.156
marine food web	Yes	135	598	4.43	2.05	-	0.23	-0.263
freshwater food web	Yes	92	997	10.84	1.90	-	0.087	-0.326
C. elegans	Yes	307	2359	7.68	3.97	-	0.28	-0.226

Table 4.8: A subset of the real-world examples of network statistics taken from published work and collected by Newman [257]. Properties are directed (dir.) network type (Yes or No), n total number of nodes, m total number of edges, $\langle k \rangle$ mean degree, L characteristic path length, α power exponent if degree distribution follows a power law (in/out-degree if directed), C clustering coefficient and degree correlation coefficient r.

Table 4.8 is presented as a sample of real-world networks. The examples cover a spread of domains from organic systems to artificial and technological structures. Although there are significant results from this data alone, what is not shown is a comparison of each system to a random equivalent[21], which is an effective way to identify values that are significantly different.

4.5 Topology Models

4.5.1 Introduction

In an effort to understand the properties of real-world graphs there have been many topology model proposals. It is known that in nature real-world graphs have generic features [360, 27, 207] such as *high clustering*, *short characteristic path lengths* and *non-random degree distributions*. It is also known that some networks are based on hierarchical structures and/or modular community based structures. It is desirable to create simple models that can replicate such known properties, as this can help us understand the mechanisms at play in nature [257]. Understanding these mechanisms also opens opportunities

[21] As already suggested, there is more than one way to define a random equivalent, from the very basic to those that maintain an equivalent degree distribution.

to apply the knowledge to problem solving techniques such as evolutionary algorithms, where graphs can be used and developed.

There are three main approaches to modelling networks:

- create a graph with the required number of nodes and edges in a **single initialisation** phase (using simple rules for topology);
- use an existing structure (regular or otherwise) and use a process of **alteration** (guided or unguided) to re-organise the topology; or,
- use a **development model** that can grow or adapt a network in a step-by-step manner (also guided or unguided).

Indeed, many models use combinations of approaches to achieve their goals.

Perhaps the easiest models to begin with are those with regular or structured topology, created in a simple single stage of development, such as the linear, ring and grid lattice models mentioned. For graphs created with such models there are clear relationships between the structure, size, edges, clustering properties, characteristic path lengths and so on.

Alteration models make it possible to do null-hypothesis comparisons between real-world networks and randomised models based on a subset of the known properties and characteristics. In this way, features that are either characteristic of a system, or are at least distinguishable from random processes, can be identified if they are significant. This includes the identification of motif types and quantities.

Section 4.5.4 looks at important random models, in particular the influential *ER Model* of Erdös and Rényi [103], and the *Configuration Model* [241] for specifying degree distribution in an otherwise random graph. As will be shown in Section 4.5.5, the small-world model of Watts and Strogatz [360] is based on a regular lattice model and uses adaptation to "re-wire" the regular model towards a random model.

The observation that real-world networks often have scale-free degree distributions has prompted a large amount of research, and the development of many models. Almost exclusively, these models are of the developmental kind, and make use of some type of guided incremental addition to a graph so that it will develop with the qualities of interest.

Two important growth models are those of Price [282] (Section 4.5.6) from the 1950's study of publication citations, and the more recent rediscovery of a similar model by Barabási and Albert [27] (Section 4.5.7). Both development style models use a process of guided edge connection based on "preferential attachment" – a process observed in many real world systems. These models are able to create power-law style scale-free graphs. The amount of research targeted towards scale-free power-law models has also drawn some criticism, not because the observation is false but rather that the importance placed upon it is too high [188].

An adaptive model known as *merge and regenerate* [107] is presented in Section 4.5.8. It is similar in its rewiring process to that of the small-world model, however it is able to create scale-free graphs with the qualities of real world systems. Because it uses a

localised development process, not a global information based method, it is more suited to modelling many real world systems.

Some researchers have suggested network models that do not take into consideration localised processes are simply the "first generation". New "second generation" models are needed, incorporating localised processes and information based ideas, in order to further the research field of complex networks.

It would be a daunting task to catalogue all the graph models, especially recent additions, and that is not the purpose of this review. Instead a number of influential and distinctive models are presented that are relevant to the work of this thesis. In Section 4.5.9 some of the important model features are highlighted for comparison.

4.5.2 Regular Models

Regular one-dimensional (1D) lattices are the simplest and easiest models to begin with, and the principles that apply to this singular-dimension case also apply to n-dimensional models. A 1D lattice model is also known as a linear model, and if the boundary conditions allow "wrap-around", the model forms a circle – this is the basis of a circular "ring" model.

Consider that a 1D model can contain n vertices that are each given a location in space. Edges connect vertices along the dimensional axis and the axis denotes the neighbourhood for any specific vertex. So, for a 1D model with average degree $k = 4$, there would be two connections for each vertex in each possible axis direction.

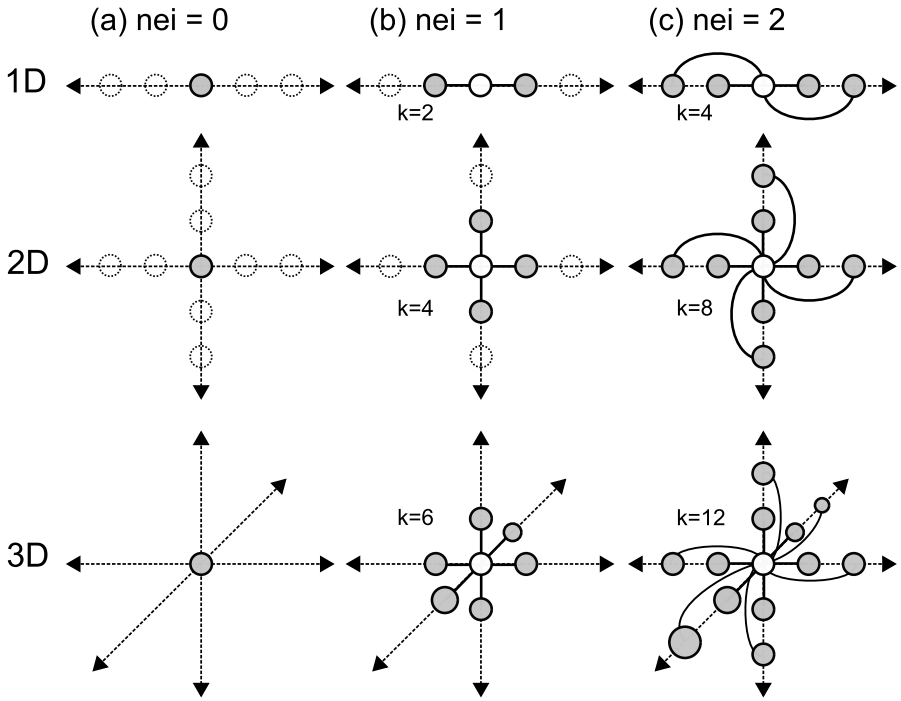


Figure 4.9: Lattice model example for 1D, 2D and 3D cases, where the neighbourhood size (nei) is restricted along dimensional axes. This same process can be generalised to an n dimensional case.

For a two-dimensional (2D) case, the additional axis of freedom allows for more connections, and so a radius of two connection along either direction of each access would

total $k = 8$. A 2D lattice can also use an unbounded "wrap-around" model, which effectively maps to a toroidal space. This regular model can naturally be extended to an n dimensional case, along with the calculation of neighbourhood degree. See Figure 4.9 for simple examples of *neighbourhood size* for 1D, 2D and 3D cases.

A clarification on the term *neighbourhood size*: some models use the term to describe a "radius" style measure extending from the location of each vertex, while other works refer to neighbourhood size as the explicit number of vertices included in a neighbourhood (essentially degree k) such as shown earlier in Figure 3.15. In this thesis the term neighbourhood size is generally avoided in favour of vertex *degree*, as this is the clearest numerical indicator regardless of dimensional lattice structure or other topological features.

Not all regular topology models are restricted to dimensionally orthogonal lattices. Tessellation patterns are a general model that, although most are popularly known for 2D designs and aesthetic appeal, have a useful application as references for regular graph structures. In fact, tessellation models work well as a model for ecological environments, with their inherent physical restrictions to interaction between adjacent cells or locations.

Some regular topology models use a non-homogeneous vertex degree. For example, a tessellation style pattern can be the result of a mixture of two or more vertex types (say types a and b) with different degree (say $k_a = 4$ and $k_b = 8$), but again the overall topology and statistical properties of the model will be known.

The examples of regular structures in Figure 4.10 are tessellations that utilise the edges of polygon shapes as graph edges, and edge intersection as node locations. However, the terms "hexagonal", "square" and so on refer to the polygon shapes that the edges form. As we will see in later chapters, it is often useful to use the tiles as nodes, and the adjacency of tiles as the metaphor for a graph edge. This idea of tile based cell locations is the model used in cellular automata (CA).

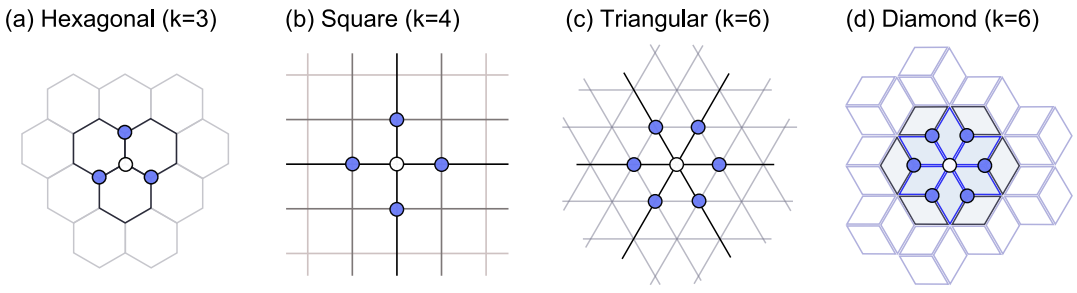


Figure 4.10: Examples of (a) hexagonal, (b) square, (c) triangle and (d) diamond regular graph patterns. All the patterns shown are tessellations, and include a central vertex connected to its immediate neighbours. In these examples the intersection points of tile edges are used as vertex locations. Many similar tessellation models use the tiles as vertices and the adjacent tile faces as edges, and both are valid methods. Note that (a) and (d) could not be formed by a simpler regular lattice model, and that although (c) and (d) have the same degree ($k = 6$) the extended neighbourhood (and hence clustering) is quite different.

4.5.3 Hierarchical Models

It is possible to use a development model based on simple regular structures to construct hierarchical models. For example, a simple hierarchy can be constructed as a tree model. Consider a balanced (homogeneous) tree, where a single root node has a number of children c. As the tree is expanded (grown) each child becomes a parent to c new children leaf nodes. The degree distribution for the graph is uniform with the exception of the root ($k = c$) and children ($k = 1$) nodes. Cycles are not possible in a tree graph. A star (or focal) topology is a special case of tree model, where there is only a single parent node to which all children are exclusively connected.

Another example of a structured hierarchical model, developed using an incremental growth model, is shown in Figure 4.11. In this case the growth model is based on a copying technique, where the entire graph at the last time step is replicated four times and connected directly to the root vertex. The resulting graph is exponential in growth with a strong power-law degree distribution. Even more distinctly, such a model also has a correlation between individual vertex clustering properties and vertex degree. Put simply, new "outer" nodes with few connections are very likely to be connected to a highly connected "hub" node.

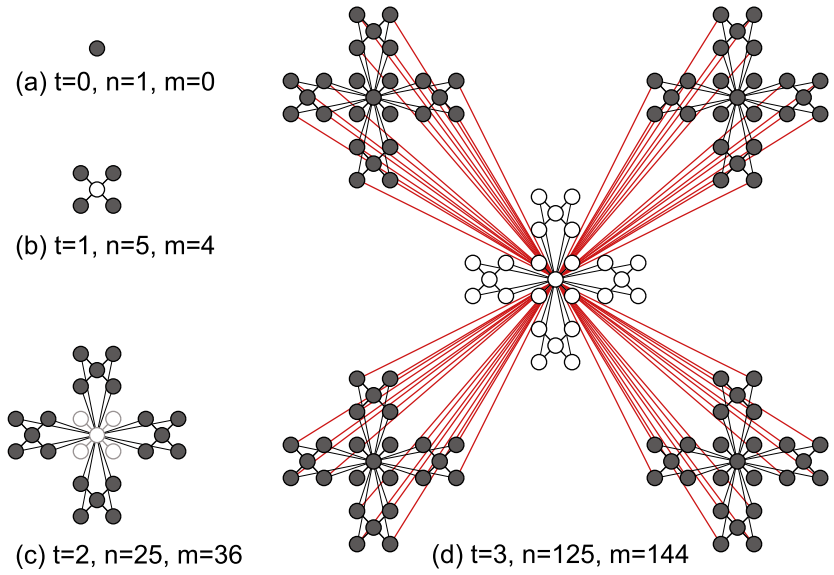


Figure 4.11: Hierarchical graph growth. The initial graph (a) is a single vertex without edges ($n = 1$). At each step $t + 1$, the current graph is replicated four times and connected to the centre of the previous graph. In (b) we see the addition of four more vertices to the graph (total $1+4 = 5$), and (c) and (d) represent two additional steps in the growth. White coloured vertices indicate "old" growth, while darker grey nodes represent the new replicated subgraphs added to the system.

There are examples of hierarchical structures in nature, including genealogies (and classifications of life) and synthesis pathways. In such cases, hierarchy seems to be the direct result of a replication process. However, although hierarchical models are not overly common in real-world systems (their construction tends to be very expensive and vulnerable to attack if there are highly connected hubs), the clustering degree correlation is an

interesting feature to look for, and could indicate the presence of a similar development process when observed in real systems.

For example, Ravasz and colleagues have shown that metabolic networks from 43 distinct organisms are organised into small, highly connected building-block topologies which are combined in a hierarchical manner to form larger networks, with clustering degree following a power-law [286].

4.5.4 Random Graphs

The ER Model

Erdös and Rényi introduced a random graph model [103] that has since been widely used in the study of complex and real-world networks [9]. There are actually two types of ER models that can be used: the first based on a fixed number of vertices, the second based on a fixed number of vertices and edges.

Note that with truly random models, there is no inherent requirements for the graph to be connected (all nodes connected), or an exclusion of connection loops or multiple connections between nodes. It is important to distinguish exactly the type of "random" graph models in use, not just the type of ER model used, as there are significant statistical (and topological) differences in the graphs created with such methods.

Let us consider the first type of ER random graph. We can specify the model as two simple ideas:

- The graph has a fixed number of vertices n.
- Each pair of vertices has the probability p of being connected.

In a concise way, we denote an ER graph G created with $G(n, p)$ to represent both these ideas. Note that the number of edges m is a product of the random formation process. So, based on this model, the average number of edges $\langle m \rangle$ is then

$$\langle m \rangle = pn\,(n-1)\,/2 \tag{4.19}$$

The probability of vertices with k edges is p^k, and the resultant degree distribution is binomial, stated as

$$P(k) = \binom{n-1}{k} p^k\,(1-p)^{n-1-k} \tag{4.20}$$

We see that there are $\binom{n-1}{k}$ possible vertices for k edges, and that the total distribution is both p^k, the probability of connections of degree k, and the probability of no extra edges $(1-p)^{n-1-k}$. For a graph with large n, $P(k)$ takes the form of a Poisson distribution where $\langle k \rangle$ is the average degree. So,

$$P(k) = \frac{e^{-\langle k \rangle}\,\langle k \rangle^k}{k!} \tag{4.21}$$

and,

$$\langle k \rangle = p(n-1) \approx pn \tag{4.22}$$

From the extensive amount of study that ER models have undergone (see the reviews of [9, 92]), we know that when $\langle k \rangle \gtrsim \ln(n)$ nearly all ER networks are *connected* and that the characteristic path length L_{rand} is proportional to the number of nodes and the average degree.

$$L_{rand} = \frac{\ln n}{\ln \langle k \rangle} \tag{4.23}$$

The clustering coefficient C_{rand} is simply the probability p that any two vertices are connected.

$$C_{rand} = p \approx \frac{\langle k \rangle}{n} \tag{4.24}$$

Although both random and real-world networks have similar characteristic path lengths ($L \approx L_{rand}$), for larger networks we see a noticeable difference in the clustering coefficient. See Table 4.7 where clustering is presented in the measures of local efficiency E_{loc} and E_{loc}^{rand}. In real-world networks we see higher levels of clustering (and thus greater local efficiency) than observed in random ER models.

The second type of ER model has an additional restriction of m on the total number of, and so in summary is based on the following ideas:

- The graph has a fixed number of vertices n.
- There are a fixed number of edges m.
- The distribution of edges is uniformly random.

A graph G created with this method $G(n, m)$ is useful because it can match an existing real-world system that has a known number of edges. However, the uniform distribution of edge degree is a big limitation, and other models are needed for null-model comparisons.

Configuration Model

As already mentioned, the ER model creates a graph with a Poisson degree distribution, however this may not be the type of distribution needed to model real networks, or at least the interesting qualities of real systems [360, 9]. The configuration model [241, 242, 257] is an excellent way to model a known or specific degree distribution in an otherwise random topology. The ideas of the configuration model are:

- Define the number of vertices n.
- Define the total number of edges m.
- Define a sequence of vertex degrees $\{k_1, \ldots k_n\}$, one for each vertex. Multiple vertices can have the same degree level, and the sum total of vertex degrees for the graph is m.

- Randomly select nodes and connect vertices while maintaining the allocated vertex degree until all edges have been specified.

The ensemble of graphs that can be created with an equal degree sequence is the "configuration model". There are points at which it is known that graphs created with the configuration model undergo giant-component formation – called a "phase transition" in likeness to physical processes of similar form.

The configuration model is often used as a *null-model*, where a known graph is compared to a random configuration model graph with the same sequence of degree. In this way, a configuration model graph can be used to help identify the distinguishing connections, structures and properties of the real-world graph under observation.

As already mentioned regarding random models, it is possible that graphs created using a purely random connection model will have self-loops (vertices connected to themselves) or multiple (duplicate) edges between vertices. If a null-model comparison is being made with a system that does not allow such connections to exist, then a restricted connection model that matches these restrictions can also be used. However, the inclusion of extra connection rules will impact on the topology of the network, especially for systems of small size, or at the extremes of high and low edge degree values.

4.5.5 Small-World Model

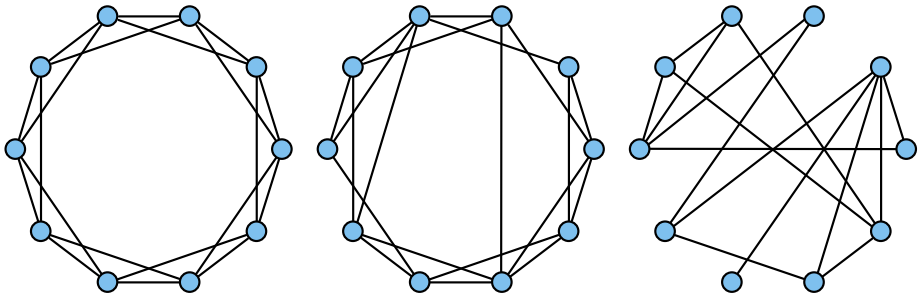
In Section 4.1.4 of the introduction the small-world phenomena was discussed, along with a brief introduction to the work by Watts and Strogatz [360] on small-world network models. It is this work, published in Nature and receiving interest from many different disciplines, that is credited with a large part of the resurgent interest in graph topology research.

The Watts-Strogatz small-world (SW) model is able to create regular or random topologies upon a regular lattice[22], with a probability parameter that controls the degree of order or randomness. The model can be described with the following steps:

- Create a regular lattice of n vertices (ring lattice).
- Each vertex is given k edges connected to the closest vertices.
- Using a constant probability p the end point of each edge is rewired randomly.

As we can see in Figure 4.12, changing the value of p has a distinctive effect on the network topology. By running experiments that varied p, and observing statistical values, specifically the *mean path length* and the *clustering coefficient*, Watts and Strogatz were able to compare their model to real-world networks with appealing similarities. They had created a relatively simple model that matched real-world network characteristics far better than ER models.

[22]The classic case is a simple linear regular lattice with bounding conditions to create a ring lattice. Regular connections are created between nodes within a limited neighbourhood radius. See Section 4.5.2 regarding regular models.



(a) $p = 0$ (b) $p = 0.1$ (c) $p = 1.0$

Figure 4.12: Three examples of the influence of the rewiring probability p in the Watts-Strogatz model. In (a) $p = 0$ and gives the initial (unchanged) regular lattice of $n = 10$ edges each with degree $k = 4$, (b) $p = 0.1$ and shows an example case between order and randomness, (c) is $p = 1$ and the resulting random topology.

In creating networks with this model we see that the graph starts with a single connected graph and that, because of the localised regular connections, clustering is high and the mean-path length will be relatively high (especially on larger sized graphs). Once the rewiring process begins, it is possible for long-distance connections to appear. Conceptually, this means that the benefits of highly clustered local structures remain, while the overall mean-path length is dramatically reduced. Put another way, the diameter of the graph is exponentially smaller than its size: even very large networks can still retain very low mean path lengths. If the rewiring process continues, the benefit of local clustering will be destroyed and the topology will equal a simple ER random graph.

Figure 4.13 shows a recreation of the experiment conditions and data as described by Watts and Strogatz in [360]. In particular, note the small-world effect where clustering remains high (due to the initial regular ring-lattice topology) however mean path length drops low (due to the rewriting and provision of critical long-distance connections). Implementation details of this experiment are supplied in Appendix G.

It is interesting to note that the underlying lattice has benefits and drawbacks. Many real-world systems contain regular, physically limited connections, and in simple cases the regular lattice can be a good match. However, when complex interactions take place (as processes on the topology), the lattice representation may be too removed from the real system, and the model a poor resemblance of the real system.

Small-world (SW) models have been generalised to multiple dimensions, and without the need for specific coordinate systems. In particular, as already mentioned in Section 4.4.3, Jon Kleinberg [204, 205] has worked with 2D lattice models and investigated the nature of search (navigation) with limited local information, as occurs in many real-world examples.

Although this SW model was one of the first models to explain both high clustering and small-world characteristics, the degree distribution is narrow and dependant on system scale. A critical observation, because of its influence to research initiatives, was that real-world networks usually contain a broad spread of degree distribution, and this encouraged many researchers to find mechanisms that could explain or model this.

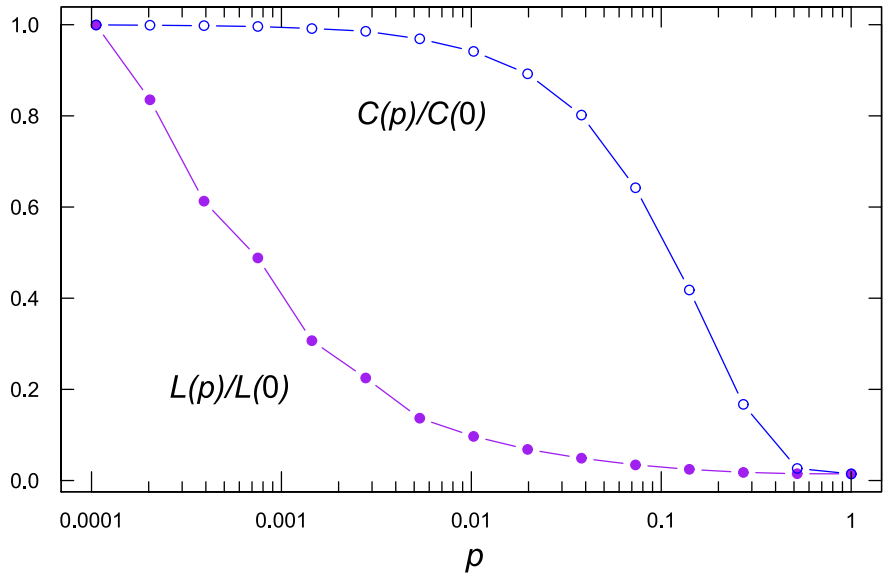


Figure 4.13: The effect of p on L and C in the small-world (SW) model. This figure is a recreation of the model and experiment data as described by Watts and Strogatz in Figure 2 of [360]. The ring lattice model has been used with 1000 nodes, an average degree of 10, and a re-wiring probability p is varied. Result were collected for 20 random realisations, and the values for the clustering coefficient $C(p)$ and mean path length $L(p)$ were averaged and normalised against the lattice model result for $p = 0$. A logarithmic scale is used to illustrate the rapid drop in $L(p)$ while $C(p)$ remains relatively high – the classic "small-world effect".

4.5.6 Price's Growth Model

The work of Derek J. de Solla Price (1922-1983) starts with the work of Herbert Simon (1916-2001) who, in the 1950's [316] showed that models of wealth growth using a "rich get richer" approach resulted in power law increases. Simon named the approach *cumulative advantage*[23] and his work has been influential to many fields. Similar non-network studies of the existence of power-laws in real systems (such as word frequencies and income distributions) were also known [378] as the principle of "least effort" or the *Zipf Law*.

In 1965 Price published [282] his studies of scientific paper citation networks, in which he had observed power-law distributions for both the in and out-degree of citations. Later in 1975 Price published again [283] on the cumulative advantage process he had observed in bibliographic processes, this time with a model to explain the power-law degree distributions.

As Newman has pointed out, the "important contribution of Price's work was to take the ideas of Simon and apply them to the growth of a network" [257]. The growth model can be described simply as the probability that a paper will be cited is proportional to the number of citations it already has. Simon and Price assumed a simple linear relationship, however this need not be the case. The detail of Price's model uses a *master-equation* and

[23] In sociology research this is also known as the *Matthew effect* after a well known biblical text Mat.25:29 "For everyone that hath shall be given..."

is presented in detail in [257]. Some of the model is presented here as a reference for the popular Barabási and Albert (BA) model.

A citation network is considered a directed graph of n vertices, where each publication has a fixed out-degree of papers that it cites at the time of publication. The in-degree of a publication is all the vertices that refer to it, and may grow over time – as has already been mentioned – with a cumulative advantage to publications that already have a high citation degree.

The model allows p_k to be the normalised fraction of vertices with in-degree of k, so that $\sum_k p_k = 1$. Although the model allows for a step-wise addition of new vertices, the rate of addition (number of vertices added each step) can vary over time. The average out-degree of the network $\langle k \rangle$ is held constant over time, and models the reasonable assumption that new publications will cite an average number of old publications.

At the heart of the model is the probability that a new paper (vertex) will cite an existing vertex increases proportionally to the in-degree of the existing vertex.

This creates an interesting problem in that because each vertex starts with an in-degree of zero, how then does a vertex receive its first citation? A simple solution, proposed by Price, is to add an offset constant k_0 to the attachment probability k, and in Price's formulation the value of k_0 was simply 1. It has been shown that this does not affect the power-law exponent value, although this type of assumption does affect the Barabási and Albert (BA) model, and so is an interesting difference between the two models.

Thus, the probability that a new vertex is attached to any existing vertex of in-degree k is given by

$$\frac{(k+1)\,p_k}{\sum_k (k+1)\,p_k} = \frac{(k+1)\,p_k}{\langle k \rangle + 1} \tag{4.25}$$

where p_k is the fraction of vertices with in-degree k, and $\langle k \rangle$ is used to represent the *mean out-degree* of the graph.[24]

By looking at the net change in np_k per vertex, and then rearranging stationary solutions where $p_{k,n+1} = p_{k,n} = p_k$, the probability p_k can be shown to be a power-law relationship of the form α^{-b}, expressed as

$$p_k \sim k^{-(2+1/\langle k \rangle)} \tag{4.26}$$

This tells us that in the limit of large n the degree distribution will have a power-law tail with an exponent value of $\alpha = 2 + 1/\langle k \rangle$, or in the interval between 2 and 3. This power-law tail relationship matches well with published observations of large real-world networks (see Section 4.4.9 and Table 4.8), and is discussed later in Section 4.5.9.

4.5.7 Barabási and Albert (BA) Growth Model

More recently, Barabási and Albert championed the idea that real-world scale-free networks develop as a result of two distinct features: real networks *grow*, and vertices connect to existing vertices using *preferential attachment*. In this regard the model proposed by

[24] Note that the mean out-degree is also directly the mean in-degree $\langle k \rangle = \sum_k k p_k$.

Barabási and Albert[27] is the same as the model of Price, but with one very significant difference – the BA model uses undirected edges.

The use of undirected edges means that in the BA model there is no distinction between in- and out-degree, and while this simplifies the model and avoids the problem of "first" citations, real networks such as the Internet (which the BA model was trying to emulate) are clearly directed and hence it is possible that valuable qualities may be missing from the model.

The model process can be simply stated as follows:

- Start with an initial number of vertices n_0 and no edges (a disconnected graph).
- At each update step, a new vertex is added with m edges to existing vertices, where $m < n_0$.
- The probability that a new edge (from the new vertex) will connect to an existing vertex is linearly dependent on vertex degree k.

Here the common notation of m is used to denote the *edges added per time step*, even though this is inconsistent with prior use of m as the total number of edges in a graph. However, this is consistent within the literature on network growth models.

Like Price's model, vertices are added to the network, with the probability that a new edge attaches to a vertex of degree k given by:

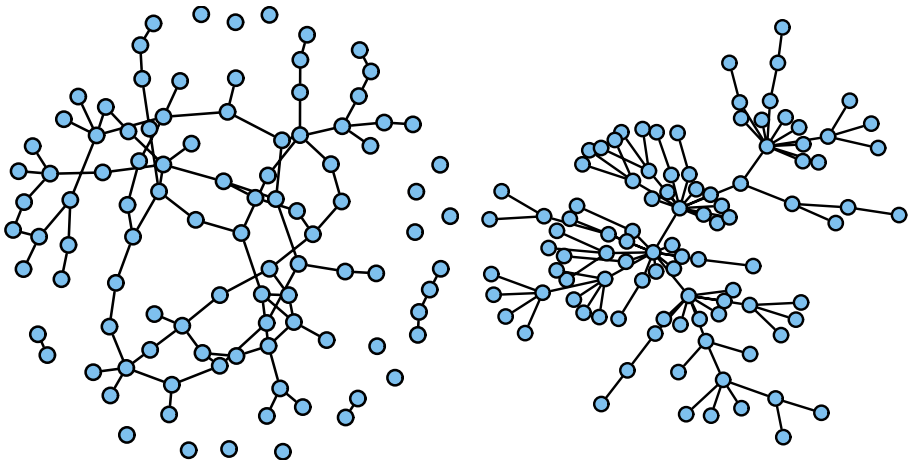
$$\frac{kp_k}{\sum_k kp_k} = \frac{kp_k}{2m} \tag{4.27}$$

Each vertex has a degree m and this value is not altered as the network grows. The $2m$ in the denominator represents the fact that undirected edges contribute a degree to each of the vertices they connect.

We can assume that the BA model uses directed edges, and that the probability of attachment is actually based on the sum of both in and out-degree values per vertex. Although technically valid, this does not make sense for many real networks. For example, in some networks there is a reasonable expectation that underlying development processes make a preferential distinction between in and out edges.

Using the master-equation method (noted earlier in Section 4.5.6 as part of the presentation of Price's growth model, and detailed in [257]), the BA model can be solved exactly in the limit of large graph size [213, 92]. It has been shown by several authors that in the limit of large k the BA model gives a power distribution of $p_k \sim k^{-3}$, with a single fixed power-law exponent of $\alpha = 3$.

The BA model has been extensively studied, and is known to contain other features, some of which are quite distinctly different from real-world networks. For example, there is a correlation between vertex age and degree in real world networks in that the earliest (oldest) vertices are expected to have much higher degrees than that young vertices. This is not the case in many real networks such as the WWW, and so has been a reasonable argument against a BA model. However, even though there is an age correlation in graphs created with the BA model this does not mean that the mechanism of preferential



(a) ER Model (b) BA Model

Figure 4.14: Examples of graphs created using the ER and BA growth models. Both graphs contain 100 vertices and are constructed using incremental growth models. In (a) the Erdős-Rényi (ER) model of $G(n, p)$ has been used with $p = 1/50$, and the layout method is that of Fruchterman and Reingold [122]. In (b) the model of Barabási and Albert (BA) creates a graph using *preferential attachment* for nodes with a higher degree. This particular graph uses a linear growth model, and only one new vertex is added each growth step. The layout method for (b) is that of Kamada and Kawai [183] which is well suited to connected graphs such as this one.

attachment is not a valid primary influence in the development of real networks. It simply means that age based preferential attachment is not a universal mechanism.

Secondly, there is a limitation to the model in that it builds upon an unchanging structure, something very rare in real-systems where an existing structure can easily change. In the WWW for example, web sites that were once active and the focus of much preferential attachment, become "dead" and inactive, or simply disappear.

The BA model can be modified to include an exponential limit [9], such that older nodes lose their "attractiveness" as they get older. This would limit the possibility of nodes with extremely high degree. However a uniform limit process is rarely observed in real systems, which suggests that limit should not be used.

A good model that represents natural network formation should include the notion of "who is connected to whom" – preferential attachment based on *referential* information. Unfortunately, the BA model of preferential attachment is based simply on vertex degree alone.

The contribution of the BA model to complex systems research – particularly the Internet – has been great, and even its limitations have encouraged new avenues of research. Since the introduction of the BA model, there has been a significant focus of research into the development of models that adequately capture the qualities and properties of real-world complex networks.

4.5.8 Merge-Regenerate Models

Merging and regeneration models are also known as "merge and create", "aggregation and injection" or simply "merging" models. As all these names suggest, the central principle of the model is the *merging of nodes* (or groups), and the *addition of new nodes*. The model is an alteration process usually based on an initial random (ER style) graph.

This type of model was suggested in the field of astrophysical systems by Field and Saslow in 1965 [107]. They proposed that a statistical model that included a process of aggregation and injection could be used to generate or develop the formation of structure (stars and interstellar clouds). Their early work is cited by the recent work of Kim et al. [191] and others [298, 297, 267].

Using an initial random graph $G(n, m)$ (ER model) of n vertices and m edges, the merging and regeneration process can be described as an *update rule* for each time step t:

- Select a random node i with degree k_i.
- Select a random neighbour j of node i.
- Merge nodes i and j such that the new node merge will have $k_{\text{merge}} = (k_i - 1) + (k_j - 1) - k_{\text{common}}$ edges, where k_{common} is the number of nodes (hence edges) that were common to both i and j nodes.
- Create and add a new node of degree k_{new} to the graph and attach it to a uniform random selection of existing nodes. The degree k_{new} is picked from a uniform distribution with an average of $\langle r \rangle$.

The update rule process is also shown in Figure 4.15.

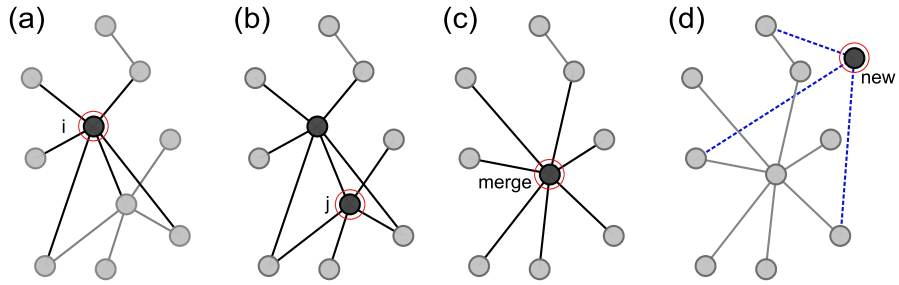


Figure 4.15: A representation of the merge and create model, as described by [191]. In step (a) a random node is selected, then (b) a randomly selected neighbour is nominated for merging, (c) the resulting merged graph. In (d) a new node is added and randomly connected to the network (to retain the overall number of nodes), and with a random number of edges consistent with the mean of the network.

One of the strongest motivations for the development of this type of model is the dissatisfaction of the Price(Simons)/BA models, and similar hierarchical models, that require global information to govern development. It seems more plausible that local methods, driven by optimisation or similar advantages, would be at play in real-world network development, where many development processes are local.

As Kim et al. have discussed in [191], the two prevalent models used to explain broad degree distribution are the preferential attachment models or other models that are "driven" to create hubs against a backdrop of pressure towards a random system. Instead, their merge and regenerate style model is able to create networks with scale-free degree distribution, of the form $P(k) \sim k^{-\alpha}$, using fairly simple localised and real-world processes. (This power-law relationship was discussed earlier in Section 4.4.2 and shown in Equation (4.4).)

The motivations raised also align with the criticisms raised by Keller [188] that the exaggerated focus of preferential attachment style (BA) scale-free models has limited the view of the real mechanisms at play in biological (and other) systems.

Merge and create style models correspond well with the ideas of development processes in real-world systems. For example, it is likely in a computer networking context to replace two switching devices with a single device of equivalent capacity as a consolidation of resources (and expense). Similarly, new switches can be added to the network based on demand. Another example is that of social network, where a social connection between two people also means that the connections between the two people may be shared, such that both individuals will share many links and thus take direct paths rather than indirect paths when searching or transferring information. The consolidation of links between friends represents a natural process of optimisation, where individuals take direct rather than indirect paths.

Takemoto and Oosawa have created a model that merges "building blocks" (essentially small motif structures) rather than individual nodes [335]. This is similar to the idea of hierarchical metabolic network blocks (such as the ones presented in [286] and cited as an influence to their model) or the merging of communities in social systems. Their model also produces power-law degree distributions and clustering spectra and high average clustering coefficients independent of network size. Their model for the exponential rule is "tuneable" by adjusting the ratio of merged nodes to that of all nodes in building blocks. It is reported that the model is also like the BA under specific conditions [335].

4.5.9 Comparing Topology Models

This review of topology models has considered *regular* (Section 4.5.2) and *hierarchical* models (Section 4.5.3), *random* graph models including the ER and *configuration* models (Section 4.5.4), the *small-world* (SW) rewiring model of Watts and Strogatz (Section 4.5.5), the *preferential attachment* growth models of Price (Section 4.5.6) and Barabási and Albert (BA) (Section 4.5.7), and the *merge and regenerate* style models of Kim and others (Section 4.5.8).

These models are now compared using some of the distinctive measurements and properties described earlier in Section 4.4. Of particular interest are processes of development, distinctive structural properties, and the resultant degree distribution and clustering to vertex degree.

As an initial comparison, Figure 4.16 presents degree distribution and cluster coefficient

relationships for three reference graph models. This comparison is similar to that shown graphically in Box 2 of the review by Barabási and Oltvai in [29].

It can been seen from (1a) of Figure 4.16 that the ER model has a Poisson degree distribution (typically near the average degree $\langle k \rangle$) and (1b) no correlation between clustering and vertex degree. The BA model demonstrates the notion (2a) of a "scale-free" power-law degree distribution and uncorrelated clustering. The hierarchical model shows not only a power-law degree distribution (3a), but also a very distinctive correlation of clustering to vertex degree (3b).

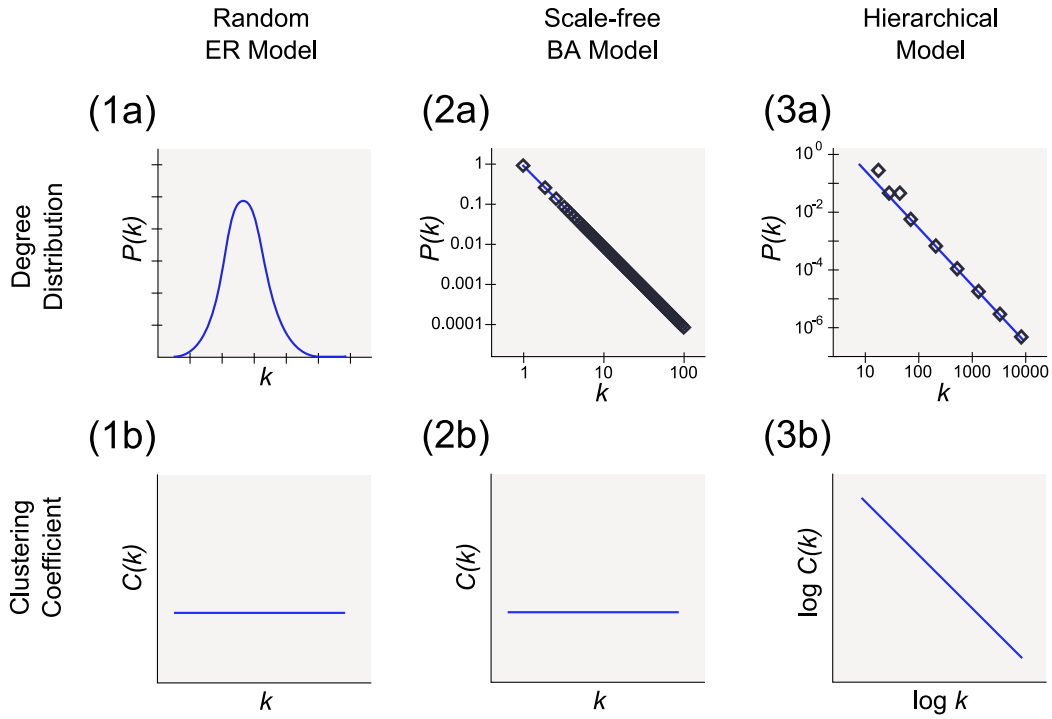


Figure 4.16: Graph models compared: the (1) ER random model, the (2) BA scale-free model and a simple (3) hierarchical model.

The comparison points can be extended to cover all the models discussed in this section:

Regular models are commonly based on lattice topologies, and because of their "regularity" all have a clear relationship between size and topology features across scale. If all vertices have the same degree (or a regular pattern), then the degree distribution of $P(k)$ with respect to k is a simple constant (or a set of constants). As there is a fixed degree distribution, the clustering coefficient is also fixed by the topology of the system. Regular models are a useful basis for other models, but do not display interesting features with respect to degree distribution, cluster or mean-path lengths.

Hierarchical models, although based on a regular pattern of structure, are quite different from simple regular models. The repetition of structure as a growth process creates a direct correlation between degree distribution and vertex degree. Consider that a hierarchical model that connects new subgraphs to an existing "hub" vertex will create strong correlation between the probability of vertex degree $P(k)$ and k,

and possibly a distinct affect on clustering with respect to degree (as newly added "isolated" and sparsely connected vertices are attached to existing highly connected "hubs"). See Figure 4.16 (3a and 3b) for an example of these relationships for the hierarchical model described earlier in Figure 4.11 of Section 4.5.3.

Random models, such as that described by the ER model, are likely to have a clear relationship in the distribution of vertex degree related to the random distribution used in formation. Also, without a specific guided process to influence connections based on existing degree or clustering, there is likely to be no correlation of clustering properties with vertex degree or any other factor. The random example in Figure 4.16 shows a Poisson style degree distribution (near the mean value of k for the system), and the clustering coefficient is uncorrelated and evenly distributed across all values of k.

Small-world models, like that proposed by Watts and Strogatz, are based on a regular topology, with fixed degree distribution and clustering properties, and the model is altered to include random re-wired edges. This means that for small values of p (the rewiring probability) the resulting degree distribution will resemble that for a regular topology model. As the value of p is increased towards $p = 1$, degree distribution approaches the random model case. As already presented and discussed (Section 4.5.5), it is the properties of high-clustering and low mean path length that are the distinctive real-world style qualities of the small-world model. However, features like degree distribution are linked to system scale – they are not scale free as many real-world graphs tend to be [257]. This is a strong motivation in the investigation of scale-free models.

Scale-free models, such as the "cumulative advantage" model of Price and the "preferential attachment" model of Barabási and Albert, are quite unlike the simple regular or random models. As can been seen in the representation of Figure 4.16, the BA model has the distinctive *scale-free* relationship between degree distribution and degree (a straight line when presented on a log log chart). Note also that there is no correlation between clustering and degree, and this matches the fact that the BA preferential attachment model is only based on degree, not measures of clustering or community style importance.

Merge-regenerate models are most similar in output to scale-free models, however development is more similar to the rewiring process of the small-world models, but focused on node removal and addition. Development is a localised process that does not depend on global information – in major contrast to the more popular preferential attachment models. Merge-create style models do create networks with scale-free degree distributions and real-world like clustering coefficients. Their power-law slope is also "tuneable" through growth rate parameters (merging rules).

As a summary of the network models and properties described, Table 4.9 contains

a list the models, indicates the nature of the degree distribution and the relationship of clustering to vertex degree.

Model, Process & Features	P(k)	C(k)
Regular		
▷Regular, typically homogeneous configuration. (All nodes with same degree). ▷Good basis for other models	Constant (or set of constants)	Constant
Hierarchy		
▷Repetitive (exponential) pattern of growth. ▷Very strong hubs. Clustering correlated to degree	Linear log-log (even)	Linear log-log
Random $G(n, p)$ (Erdös-Rényi)		
▷Random probability p of edge between nodes. ▷Famous and highly studied.	Poisson ($\langle k \rangle$ centred)	Constant
Random $G(n, m)$ (Erdös-Rényi)		
▷Fixed n and m. Uniform edge formation. ▷Uniform distribution limits application.	Poisson	Constant
Configuration (Random)		
▷Uses specified degree distribution. ▷Excellent for **null-model** comparisons. Interesting phase transitions.	As specified (set of values)	Constant
Small-world (Watts-Strogatz)		
▷Regular lattice with random rewiring. ▷Low L, high C - "small-world" effect.	From constant to Poisson	Constant
Scale Free (Price)		
▷Directed growth model with (linear) *cumulative advantage* attachment. ▷Models scale-free topology.	Linear log-log	Constant
Scale Free (Barabási-Albert)		
▷Undirected growth model with *preferential attachment*. ▷Avoids "first" edge issues. Highly studied.	Linear log-log	Constant
Merge-regenerate		
▷Initial ER $G(n, m)$ model. Merge two neighbours, add new node. ▷Local process, not global.	Linear log-log	Constant

Table 4.9: Summary of network models and properties. $P(k)$ represents the type of degree distribution, and $C(k)$ indicates the relationship of clustering to degree.

4.6 Networks and Processes

4.6.1 Introduction

The purpose of this section is to briefly discuss some the key process ideas as they related to network topology.

4.6.2 Utilisation

Networks are used to perform a functional role. Functions and processes that occur on a network are affected by topology. Processes that directly alter the network will certainly affect topology. This is especially interesting in the context of topology formation and the study of network resilience.

From phase transition models we are able to measure and predict critical stages in network formation. Percolation models help us to understand how the "occupancy" of nodes and edges affects a network's performance. This is a very useful way to model network resilience.

4.6.3 Navigation

The process of searching on a network, or network "navigation", models many real-world tasks and functions. Indeed, Milgram's social networking experiments [232] were not just about the network of connections, but also the ability of people to successfully use information they had to utilise the connections available. Again, understanding and making use of complex systems relies on not only the topology, but the functional manner in which the topology is used. Efficient topologies are of little consequence if a process is unable to effectively navigate to good connections.

Rosvall, Sneppen and others [298, 297, 86] have proposed a model of network formation and adaptation that is based on information transfer. They have shown that when the model is adapted based on utilisation, and imperfections exist within the information used to guide utilisation, it is possible (and likely) for scale-free topologies with hierarchical features to develop. Their model is particularly interesting, because the process of lost information (*lossy*) is a very real-world phenomenon. To have a model that presents real-world features that is also based on a realistic issue of information loss makes this an excellent candidate for many simulations.

4.6.4 Evolution

Through observation, measurement and experiment, it is fairly certain that evolution and topology are inextricably linked: it is the nature of the real-world. Consider that:

- Network topologies are formed as a result of evolutionary (adaptive) processes.
- Network topology is the environment upon which evolution occurs.

There are two key theories behind the observations that both technological and biological networks contain non-random structure, usually scale-free [91, 26, 259]. First, networks are almost always the result of a developmental or "growth" process, in which nodes and edges are added to the existing network. Second, edges are formed with a bias such as "preferential attachment".

Within the domain of software systems and development, measuring the topological changes – such as the clustering and degree correlation of systems – during system growth and evolution has provided tools for detecting change [348] and an understanding of the resilience (or resistance) of software "classes" to change [347] as software systems of significant size grow. The concept of topology in software applies at both a design level and to run-time code interaction ("coverage"). As mentioned earlier, the identification of network motifs within software code is another area of analysis that is still being explored in new ways [10].

There is a considerable interest (such as [174, 376, 227]) in the modelling of protein interaction networks. Proteins are represented as vertices, and edges are the reactions (or interactions) between them. From an evolutionary view, genes that code for proteins are duplicated during reproduction, and hence processes such as mutation or gene-duplication – if altered or adjusted – could well explain some of the linkages and dependencies that are observed [257, 36]. This also adds weight when genes are either co-located on the genome or their expression is tightly coupled.

Scale-free topologies exist in many real-world systems, yet a model of preferential attachment does not reflect the real-world process responsible in all systems. Considering the domain of evolutionary processes, what mechanisms exist that could explain cumulative advantage? Although the exact details may not be essential for artificial models, the underlying process might be significant.

Work has been presented by Jain and Krishna [175, 176] of an evolutionary system of simple chemical models capable of forming interacting "species" and increasing complexity over time. The network of interaction is a weighted graph, where positive links indicated catalytic interaction, and negative values represent inhibitory influence. The model is able to demonstrate spontaneous growth, connectivity, cooperation and complexity. It also demonstrates catastrophic events, such as the loss of critical "building-block" species, and the "rediscovery" of critical components.

The work is distinct from the earlier (1979) "hypercycle" model of macromolecular evolution proposed by Eigen and Schuster [100] which requires specific reproduction mechanisms and suffers from internal disruptive influences. Both models are interesting in that they provide a basis for modelling evolutionary processes as topologies, in particular the notions of interacting species evolving (via mutation or reproduction) in a closed system.

4.6.5 Biology and Genetics

It is an accepted concept within the fields of biology and genetics that a fuller understanding of biological and cellular systems can only occur through a framework of knowledge around cellular networks and the topology of interacting components [36, 29]. Reductionist models have provided useful knowledge of molecules and individual cell components. However, this does not give the desired overall picture and a model of the complex interactions of all components – such as DNA, RNA, small molecules and proteins. This is needed to understand the entire cellular behaviour.

In biological systems there are many interactions between proteins, metabolic processes, signalling and transcription regulation. There are *networks of networks*; a single system cannot be isolated and fully responsible for even a single cell's function. It is the topology and the dynamic properties of such networks of networks that ultimately defines the behaviour of an individual cell.

Evidence from 43 different organisms, across all organism domains (*eukaryotes*, *bacteria* and *archaea*) indicated that cellular metabolism follows a scale-free topology [177, 350, 351]. This confirmed the notion that most metabolic substrates are involved in a limited

number of reactions, but an important few participate in many reactions and are critical "hubs" for metabolic function.

Featherstone used a technique of fragmenting the observed "picture" of gene expression networks, via mutations and genome-engineering technique, to enable the isolation of gene functions [105]. This also assisted the identification of a gene expression network topology, and the presence of scale-free topology and small-world qualities. The work was motivated by the fact that most phenotype changes are the result of many genes. The scale-free organisation "helps make organisms resistant to the deleterious effects of mutation, and is thus highly adaptive".

In a study of gene expression data from different cancer types, Agrawal [3] showed the existence of power-law features, and the presence of small-world behaviour. The work also discussed the results with respect to evolutionary processes which is relevant to artificial evolution models also.

In a similar way to gene expression networks, the physically based interaction of protein-protein networks has also been shown to contain scale-free topology [36].

Wuchty [376] applied the then recent discoveries of small-world measure and scale-free topology to existing data on protein interaction. Results showed systems with a "high degree of local clustering" accompanied by a few long-distance connections. Again, the observations were valid across different organisms, and this repeats the suggestion that interaction complexity in organisms is an indicator of evolutionary progress of organisms; there is an increasing network complexity from single cell systems to multicellular systems. The author clearly states, however, that the scale-free and small-world models can only be a "rough" approximation of the real systems, and the processes used to construct the models may not be the ones responsible for the biological systems.

The work of Maslov [227] looked at the specificity and stability of protein interaction networks. Results indicated that highly connected nodes tend to link to less connected nodes (proteins), which is the "disassortative mixing" property discussed earlier (Section 4.4.2) as it relates to degree distribution.

Although research indicates the presence of scale-free topology in biology, it is not ubiquitous. A problem with the scale-free measure is its lack of specificity: it does not indicate the "strength" or a cause that underlies its existence.

The transcription regulatory networks for *S. cerevisiae* and *Escherichia coli* present a mixed case of scale-free and exponential form. Results show that transcription factors (out-degree) regulate only a few genes, and overall matches a scale-free model. However, the in-degree process of transcription factors interacting with genes matches an exponential model. From research findings it seems that most genes are regulated by a small number of transcription factors only [235, 315]. Another finding, related to the identification of motif forms, is that cellular networks have a disproportionate number of highly connected nodes, and that such highly connected nodes are a distinctive quality of the processes they represent.

There is evidence in metabolic networks of "ultra" small-world effects, such that there

are a very small number of reaction paths between metabolites and the entire network can be affected by changes in metabolite concentrations easily [177, 350, 351].

Wagner has also suggested [349] evolutionary constraints on the evolution of gene expression. It is interesting to note that a parasitic bacterium has the same mean path length as a highly developed multicellular organism. This strongly suggests that there are mechanisms at play in evolutionary processes that encourage or maintain a short mean path length in the system.

There are many biological examples of the network principles discussed in this chapter. In particular there are many examples of complex networks that exist between the components – such as molecules, DNA and proteins – that form large organisms. There is also strong indications of the role that evolution plays in the formation and development of such complex structures. This supports the work in this thesis that combines the ideas of complex system topology and applies this to the field of evolutionary computation.

4.7 Summary

This chapter has examined current research and models of graphs and complex systems, including a number graph measurements, properties, and models of graph development.

Appendix C contains a detailed survey of many instances of the topology model presented and discussed in this chapter. The survey supports the selection of appropriate topologies for use in the investigations presented later in Chapter 6.

The overall objective of this chapter, and Part I of the thesis, is to support the development of an ecosystem model of evolutionary computation. This model is presented in Chapter 5, and is able to support different organisational structures and topologies, including the models presented in this chapter.

Part II

Investigations within Ecosystem EC

Part II

- Chapter 5 presents an ecosystem framework that can be used for different organisation models of evolutionary computation (ESEC).
- Chapter 6 shows, with a series of investigations, how the ESEC framework can be applied to an organisational level of a single species population, how the topology of the population can be specified and how topology properties influence evolutionary outcomes.
- Chapter 7 considers in detail the opportunity for further research with the ESEC framework, by presenting both community and ecosystem organisations. A community is an extension of a single population model to a system composed of nested populations and multiple interacting species. Similarly, an ecosystem can be composed of nested populations, communities and entire ecosystems. Of particular interest is the opportunity to investigate the influence of interaction topology between nested systems.
- The thesis conclusion in Chapter 8 summarises results and considers further avenues of research based on the ESEC model and `esec` package.

Chapter 5

An Ecosystem Model for Evolutionary Computation

5.1 Introduction

This chapter presents the composition[1] of concepts and models presented in Part I of the thesis, in particular: ecological models and organisational levels (Chapter 2), the application of evolutionary adaptation as a means of computational search (Chapter 3), and the properties of topology to support and influence processes and activities (Chapter 4) such as those in ecological systems and in particular evolutionary adaptation.

After presenting a condensed summary of the essential and relevant aspects of ecology, evolutionary adaptation and topology, this chapter proposes an architecture of systems that can be used to implement an ecosystem model of evolutionary computation (ESEC = EcoSystem Evolutionary Computation).

Consideration is given to related work in Section 5.4, and key questions for the ESEC model are listed and discussed in Section 5.5. Section 5.6 supports later empirical investigation by presenting appropriate methods to be used when comparing performance.

The ESEC model is also detailed and used in later chapters to investigate some open questions and to validate expectations of the model and its capabilities.

5.2 Ecosystem Evolutionary Computation

5.2.1 A Composition of Models

The intent of this section is to clearly present the meaning of an "ecosystem model of evolutionary computation" (ESEC) as it is used in this thesis; ecologically based organisation, computational search based on evolutionary adaptation, and explicit consideration of topology and complexity.

An ideal outcome of the ESEC model is that in addition to the successful application of a biological metaphor (evolutionary computation) other useful concepts from ecology and

[1] The term "composition" is used in the sense of bringing together a number of different components in an integrated manner.

topology fields are also included, and possibly exploited to advantage in some applications. In order to list potential advantages and disadvantages, it is constructive to consider each of the areas presented in earlier chapters with their use in the ESEC model composition.

5.2.2 Ecology, Ecosystems and Organisation Scale

The fields of ecology and ecosystems contain many useful models and defined systems of interaction that may be of assistance to an artificial application to evolutionary computation. Chapter 2 presented many interesting and relevant concepts including the biological origins of evolutionary adaptation.

To begin, consider the model of energy and matter flow within an ecological system; energy flows into a system, the value of such energy is transferred and used to created changes in matter, and as the system is open, the energy is lost. The more exchanges of energy, the greater the losses and the reduction in usable energy value. Based on this it can be observed that within ecological systems there is always a "cost" and a finite system (of organisms and resources) can only support a finite number of organisms and processes. Not all system configurations are sustainable, and successional phases of change in configuration are common, while relatively stable systems of minimal change are the exception.

From these initial points we can consider and apply models of limited energy and resources to evolutionary computation (EC) as a "quota" system, whereby different components of the overall system (be they individuals or groups) have limited resources. Similarly, successional changes of system composition might be allowed or artificially applied to EC in order to support or encourage particular adaptations (such as robustness, competitive and cooperative behaviours).

There are various levels and types of interaction within an ecosystem. For many EC models this is specifically limited, but an ecosystem model should support community models of interaction between individuals and species. In the biological world, competition and specialisation to exploit niche opportunities are the normal. Cooperation and symbiosis form between individuals and species as a necessary complexity. Despite the cost in energy or resource use, symbiosis is required to enable species survival and stability. Although individuals within groups can have an influential or leadership role in localised environments, ecosystems are inherently decentralised and group level behaviour is emergent.

In many biological examples, an organism's life can be characterised by three activities with the environment and other individuals: foraging, reproduction and survival. Foraging and survival are purely self-serving acts, while reproduction is a drain on individual resources, yet vital for a species' longer term survival. All three activities may motivate an individual to change its location in the environment (mobility).

As a general model, mates for reproduction are selected from a local environment, with selection based on a direct proportional relationship to fitness. Offspring are likely to remain in the local environment of parent individuals, particularly when juveniles require

the support or resources of adults. The support of "weak" children provides an island style of niche.

Given that individuals typically avoid risk, the likelihood of aggressive conflicts with other individuals is inversely proportional to the availability of resources. When resources are scarce, there is more pressure on survival. Aggressive conflicts (when they occur) are resolved probabilistically in proportion to the relative "strengths" of individuals or groups.[2]

A life cycle model (Section 2.3.3) incorporates the basis of many useful abstractions that are applied to evolutionary computation. They include the components of populations, individuals and traits, the processes of fitness, survival and reproduction. In order for fitness-based selection pressure there needs to be trait variation within a population. Trait variation is the "raw material" that enables selection pressure to change the successional composition of a population.

Biological examples of the life cycle model, and of fitness-based selection pressure, present decentralised processes, competitive and cooperative interactions (at several levels), adaptation and evolution. Behaviours of an individual are the product of inherited capability and the developmental and current environment – including interaction with other individuals of many species. Some individual and group behaviours emerge as a specific result of environment and interaction; aggregate behaviours are not specifically prescribed.

Evolution involves three distinct organisational scales: "group", "individual" and "trait" (variation) components. As discussed previously (Section 2.4.9), a classic model of evolution primarily considers the "group" to be a population of "individual" organisms whose composition includes "traits" (genetic material). Macro-scale evolution and micro-scale evolution models consider different organisational scales for each component. It is appropriate to consider different or multiple concurrent evolution scales within an ecosystem model for evolutionary computation.

To consolidate, the features and processes of biological ecosystems discussed so far are:

- Decentralised: No leader, "overlord" or grand-coordinator of individual behaviours. Systems scale and survive within the possibilities of shared environment resources.
- Autonomous: Individuals have local influence (work) within local conditions (environment). Independent behaviour and interaction result based on local situations.
- Energy driven and resource limited: Life cycles and food chains. Limited resources demand appropriate solution minimisation (as a competitive necessity), while open systems require an input of energy to sustain activity.
- Organism activities: Foraging, reproduction and survival – all of which can motivate individuals to move. Reproduction and survival are proportional to fitness. Survival pressure is inversely proportional to resource limits.

[2] A "pack" of small animals working together are able to hunt larger prey that would be impossible for a single small individual to successfully attack.

- Evolution: The availability of trait variation, selectional pressure and successional change.
- Symbiosis: Individuals adapt[3] to live with others (or die). Adaptation can result in divergent niches, layers of complexity, exclusion and extinction of groups or species.
- Emergence: Group behaviours can emerge as a result of a collective interaction, including the capability for group level adaptation and behaviours.
- Scale: Evolution concepts are not bound to a specific organisation scale. There are multiple scales of evolution, including micro and macro models that may adapt concurrently. It is typical for the rate of evolutionary change within micro scale systems to be much faster than rates within macro scale systems.

5.2.3 Evolutionary Computation

A review of the origins and ideas of simple evolutionary algorithms (EAs) and the field of evolutionary computation (EC) were presented in Chapter 3. The chapter showed how, as an applied biological metaphor, evolution concepts and processes can be applied to problem solving. EAs have proved to be a robust adaptive search technique suitable for many domains, and have specific features that make them suitable for a niche of problems where other classic techniques are less suited.

Key components for evolutionary computation, and in particular an ecosystem model of EC, were identified and discussed. (See Section 3.2.2 and Table 3.3.) These components are:

- **representation** of individuals, population and communities;
- **initialisation** of individuals and other structures;
- **evaluation** of individuals and interactions;
- **selection** of individuals in many different contexts;
- **variation** operations acting on existing individuals;
- **migration** between groups of individuals; and,
- **termination** criteria as a practical necessity.

Representation should include the specification of a search domain (or search landscape), individuals and traits, and groups of individuals at small (neighbourhood) and large (population and community) scale. Additionally, an ecosystem model should support representation of composite organisations which specify subsystem components.

Individual representation could include models and processes of interaction, including reproduction. However the important features of evaluation, selection, variation (including reproduction) and migration are separated. The representation of groups of individuals ranges from simple populations of single species individuals to communities of multiple interacting species.

[3]This includes inter-generational adaptation of traits as well as intra-generational changes such as learning and behaviour modification.

The evaluation of an individual is based within a specific context. Although the overall search domain is specified for an EA, a search vector within the domain may be composed in many ways; from a single species individual to a complex composition of several species individuals. Because of this, the measure of success within the search domain is applied to the contributing individuals based on a model of relative contribution – the evaluation context. The "fitness" measure of success for an individual within its population or community can require that the search domain measure be scaled, weighted and aggregated to create a specific "fitness" value.

Selection applies to many different contexts within an evolutionary process. This includes the selection of individuals for reproduction (which can be divided into roles for the primary "parent" and partner "mate"), the selection of offspring survivors (natality), the selection of adult individuals for removal (mortality), and migration selection (for both emigration and immigration) for individuals that are displaced or copied between semi-isolated populations or communities.

Although the context to which selection is applied varies, mechanisms for selection can be shared and applied to each context. The strength of selection mechanisms vary from weak to strong. (See Section 3.2.5 for a discussion of several well known selection mechanisms.) In practice, the use of multiple strong selection mechanisms results in rapid trait convergence while too little selection strength results in poor search efficiency.

Variation is a vital component of all evolutionary processes, on which selection (via evaluation) is able to operate. The two established mechanisms of variation are recombination and mutation, of which the specific implementation depends on representation of individuals and in some cases the search domain. The performance of an EA can depend greatly on the ability of variation operators to provide useful results, and there are many domain specific operators, as well as their associated configuration parameters. (See Section 3.2.6 for illustrative examples of binary and permutation variation operators.)

The abstraction of migration as a distinct component is an essential feature of an ecosystem EC model. A migration protocol enables individuals to be moved or copied between groups such as populations and communities. Migration in this sense is not being used to describe movement of an individual within a single population, but this is not a specific limitation. A classic and well cited example that utilises inter-population migration is the "island EA", a distributed EA (dEA) described earlier in Section 3.4.6.

Termination, as a component of the generational evolutionary process, is a practical feature, often specifically linked to search domain performance or practical resource limitations and specific problem objectives.

The order and dependency of components need to be carefully considered with respect to the use of a topology for population, community or composite ecosystem structure. In particular, when using a topology to limit evaluation, or selection (directly or indirectly via evaluation), a graph instance forms a parameter of such functions (Section 3.2.10).

New offspring individuals may be added to a population either incrementally and individually, in a so called "steady state" manner, or as a combined "generational" replacement influence. As a generalisation of these two extreme models, the "gap" model supports the

description of both extremes, as well as an intermediate level of overlapping "gap" where a proportion of the existing population is replaced by a group of new offspring.

As shown in the review of common (or "classic") EA dialects (Section 3.3), most are single panmictic population models of single species individuals whose traits are applied simply (mapped directly) to a search domain. Two important "structured" EA are exceptions to the panmictic models, namely distributed EA (dEA) and cellular EA (EA) models.

Structured models tend to be suited to parallel implementations and the use of a gap model can impact on the suitability. Some approaches to concurrent implementation work best with fine-grained distributed EA models while other approaches are more practical when using coarse-grained island populations. It is a design objective that the ecosystem EC model presented should easily support any model of parallel architecture, abstracted from specific hardware decisions.

5.2.4 Topology, Complexity and Efficiency

Simple and Complex

Graphs and topology have been identified as important parts of real-world complex systems, including many levels of ecosystems. Within established evolutionary computation relatively simple or structured topologies have been applied to population and community topologies. Based on the observation of complex topologies in many "ideal" natural and artificial systems, and the relationship of effective and efficient properties of such complex systems, it is important to be able to include both simple and complex topologies within the ecosystem EC model. This enables investigation of the relationship topology and evolutionary adaptation have, and is a key focus of later chapters.

Topology Survey

Of the many graph models discussed in Chapter 4 only a representative selection need to be included and explored in detail, ranging form simple to complex. As an important supplement to the thesis, Appendix C provides a detailed survey of graph topologies that are used or related to investigations in later chapters, including simple, deterministic and complex models.

The survey includes properties and details for a number of graph types, organised into groups. Order begins with full graphs, deterministic regular lattices, tree and star graph models, and then moves on to stochastic random Erdös-Rényi (ER) models, the small-world growth model of Watts and Strogatz (WS), the Barabási-Albert (BA) scale free growth model, and the Merge-Regenerate (MR) model.

For each graph there are three common assessment and presentation forms within the review:

- Table of properties and statistics for the specified topology type for three (or more) standard graph sizes.

- Two dimensional layout image (or sample of images if presentation is variable) to visually represent the graph model topology.
- Sets of histogram plots for vertex degree and path length; one set for each standard scale.

Additional details, summaries and relevant comments are provided to assist in the comparison of topology instances. Each table of properties and statistics presents the following details which are all discussed in Chapter 4 and in the topology glossary (Appendix A):

- Number of vertices n.
- Number of edges m.
- Components of the graph (if a graph is disconnected).
- Diameter of the graph.
- Girth of the smallest cycle (if possible).
- Density of the graph, normalised.
- Mean vertex degree $\langle k \rangle$.
- Mean path length L.
- Clustering coefficient (transitivity) C.
- The normalised global efficiency of the graph E_{glob}.
- The normalised local efficiency of the graph E_{loc}.

For a single deterministic topology the values of a single graph instance are used. If a graph is stochastic, a sample of graph instances (usually 30) is created and the average values used where appropriate. All graph models are treated as undirected.

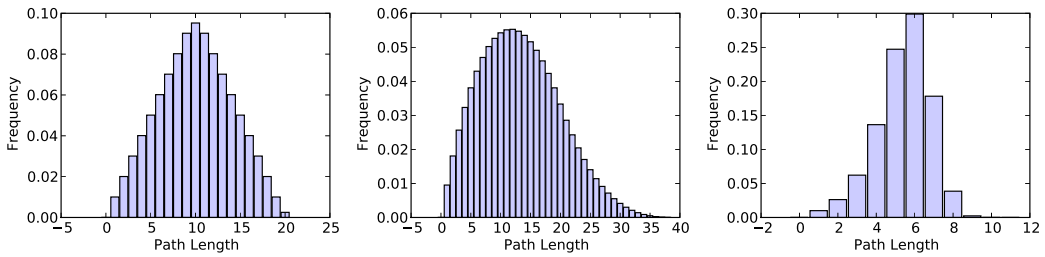
The potential complexity and variance of graph model parameters make a complete survey of all possible graph instances impossible. It is hoped, however, that the survey provides a supportive and comparative explanation of the known and expected properties of the graph instances selected for use in experiments. This kind of survey is missing from other studies utilising complex topologies with evolutionary computation.

Select Topology Summaries

Lattices of various neighbourhood configurations form the majority of the topology survey content. This has been done to consider a wide selection of the common lattice topologies considered with existing EC literature. The review includes analysis of the circular and non-circular forms, as well as novel "hollow" lattice forms. Also considered is the influence that rewiring has on each base lattice.

Overall details for regular lattices, hollow lattices and rewired lattices are all presented as summary tables. Regular lattice properties can be directly attributed to size and neighbourhood configuration and degree distribution; properties such as diameter, density, mean path length (distribution) and clustering can all be determined analytically. Lattice graphs are always a single component unless rewired in which case an isolated component may form.

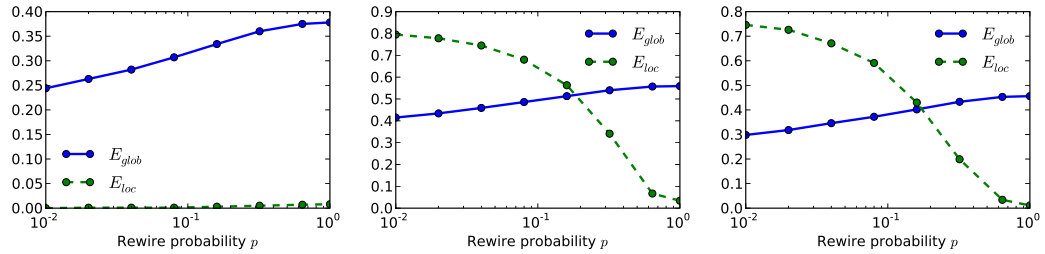
In general, the path length distribution of bound lattices have wider and larger distribution of values in comparison to equivalent circular lattices, which show a distinctive triangular form. As expected, rewiring a lattice (circular) graph reduces the width and shifts the mean value of its mean path length value distribution. See Figure 5.1 for examples of histograms for an L.k4 lattice graph of size $n = 400$, in response to rewiring. The rewiring process and its influence on a regular lattice was discussed in Section 4.5.5 as it forms the basis for the Watts-Strogatz small-world model. Rewiring creates vertices with degree values that do not exist in a simple regular lattice. The influence of rewiring can been seen in a vertex degree histogram, and as a consequence as a change to mean path length distribution.



(a) Circular (b) Bound (c) Rewired

Figure 5.1: A comparison of mean path length histograms for circular, bound and rewired L.k4 lattice graphs of size $n = 400$.

The main properties of interest influenced by rewiring are mean path length and local clustering (Figure 5.2). As the degree of rewiring is increased, the mean path length decreases (global efficiency), while the local clustering property (local efficiency) initially resist the influence of rewiring but is eventually disrupted.



(a) L.k4 (b) L.k12 (c) L.k6

Figure 5.2: Comparing the influence of rewiring on E_{glob} and E_{loc} on three different base lattices. In each case the global efficiency increases, and for the L.k12 and L.k6 instances the local efficiency is reduced.

The main influence on variations, within the observations made of rewired lattices, is the initial lattice density and local cluster configuration. If initial density is low then as rewiring takes place it is possible for isolated components to form. This applies to a biological example, in that sparsely connected organisms (low density) can be greatly affected by minor changes to the environment that easily isolate individuals or sub-populations. Sim-

ilarly, if high local efficiency is important the process of rewiring may introduce additional local connections.

Tree graphs have been included in the investigations as they are a simple deterministic hierarchical growth model, and so have a similar base form to scale free growth models involving stochastic factors. Pure balanced tree models display exponential growth properties. As the tree sizes for investigations are selected to match other lattice topology sizes all tree examples are unbalanced. Star graphs are a special case tree graph, where there is only one parent and all other vertices are child leaf nodes.

Because of the lack of cycles, trees have zero girth, clustering and local efficiency. Diameter is essentially two times the number of growth steps required to form the tree.[4] The mean path length and the graph diameter decrease in proportion to the number of children added each growth step, as more children allocated to each node increases the degree and width of each hierarchical tree level. The star model is simply an extreme case of the tree model, where the number of children added in step one is $c = n - 1$.

For the Erdös-Rényi (ER) $G(n, p)$ graph models, the average density and clustering coefficient values are well known to be essentially the same as the specified edge probability p. As such, this simple model for graph formation does not require an understanding of any other process. One important feature of ER graphs is that, for small graphs, the low "density" (the number of edges present compared to the number of potential edges in an fully connected graph) often results in isolated subgraph components. For an evolutionary process this implicitly means the population diversity is also subdivided.

Isolated components can be a useful thing for some simple search domains that require little diversity and rapid convergence is desirable. This is also a fundamental limitation for applications that require diversity. It is also a problem for a termination component if it relies on population trait "convergence" criteria – isolated components in a deceptive search landscape are less likely to concurrently converge.

The Watts-Strogatz (WS) small-world model starts initially with a 1D circular lattice and a prescribed overlapping regular neighbourhood size. As already observed and stated several times, a rewiring process reduces mean path length and increases global efficiency while ideally retaining good local efficiency. When rewiring is continued to an extreme level the graph resembles a single component random graph.

A single configuration of the Barabási-Albert (BA) preferential attachment growth model is used, with the power relationship set to $p = 1$. In this basic form the cycles are not created (as in tree graphs) and so the girth and clustering values are zero. Overall cost is quite low, while robustness to random vertex removal is high. Such hierarchical graphs are highly susceptible to targeted removal of critical vertices.

With an underlying ER model as its base, the Merge-Regenerate (MR) model initially matches the equivalent ER profiles. As a number of merge and regenerate steps are applied, the MR graph features are more likely to resemble those of a tree or star graph. It is possible, as in ER graphs, for some vertices to remain isolated. The overall mean path length remains relatively low (compared to simple ER instances) due to the concentrating

[4] In this case diameter is the number of steps required to travel from one leaf vertex, through the root vertex to another leaf.

nature of "merges" and the reconnection of new vertices to the existing components. Of the models selected, the MR model is considered only superficially. There are several parameters that influence the MR model. A deeper analysis is beyond the current scope of investigation concerns.

Expectations

As already outlined, there are several aspects of EC to which non-trivial topology might be applied. However, as a basis for presenting reasonable expectations of the influence complex topology may have to the adaptive process of evolution, consider a simple EA scenario with a single population.

For simple search domain applications, a simple population topology is likely to be best. In this type of search domain the search process can afford to be greedy; rapid convergence and loss of diversity do not have a detrimental effect on either solution quality or the search time. A full graph population topology, where all individuals have direct access to all other individuals, enables the search process to greedily exploit the current "best" solution.

For some search domains, particularly those with deceptive features, rapid population trait convergence is known to be detrimental. A population topology that helps to isolate multiple concurrent search properties is likely to support successful search. In such cases a regular lattice might be excellent, offering structured isolation for different concurrent niche variations.

Conversely, lattices can also slow progress down. For example, if two essential traits may be topologically distant the search will require numerous successive generations before a successful "mixing" of the two traits can occur. The size of a lattice and its density (based on neighbourhood topology) also add parameters to the EA paradigm. Such parameters can be sensitive to qualities of the search domain and are also difficult to know *aprior* in practical applications.

In a similar but less prescriptive manner, a simple random topology might provide suitable topology subgraphs that support semi-isolated specialisation (as in the case of a lattice), while still allowing trait mixing and overall search vector competition. It is conceivable that there are search domains where neither extreme of simple or structured topology is best, and that instead there is a niche where random topology performs best.

To retain some lattice specific properties, a regular lattice with a variable degree of rewiring provides a "tuneable" topology, which can then be adjusted for good a search domain specific performance. Such a topology provides a range of lattice isolation and global mixing can be adjusted. Rewiring creates a "phase transition" between dominant local efficiency exploitation and global efficiency (connectivity) for explorative abilities. It might be worth considering if there are applications for which the WS small-world topology may be the "right niche" for the domain, in light of the WS topology having a 1D lattice with dense overlapping neighbourhoods (greater than that found in 2D lattices).

5.2.5 An Organisation of Systems

Figure 5.3 presents the ecosystem organisational levels used by the ESEC model to support evolution and topology. There are three main organisational structures: the *population* (containing individuals of a single species), the *community* (containing multiple interacting species populations), and the *ecosystem* as a composite organisation.

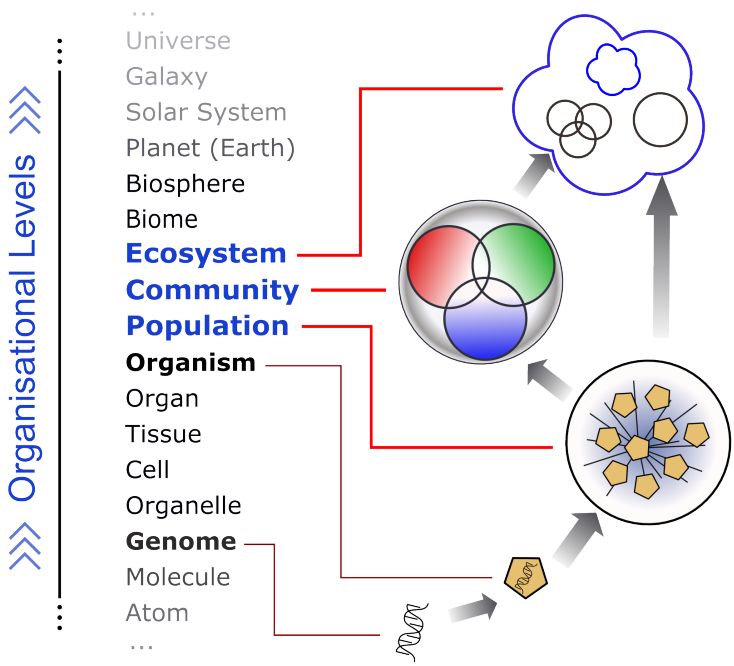


Figure 5.3: Organisational levels used as basis for ESEC components. The three main organisational levels are the *population* (containing individuals of a single species), the *community* (containing multiple interacting species population), and the *ecosystem* as a composite organisation.

Individuals are the essential evaluated entities within this model. As presented in classic models of EC, individuals can be simple values, encoded and expressed (mapped) values, or a complex network of genes to phenotype traits. An individual may also be the result of a formation (composition) within a community and environment context. Accordingly, the relative value of an individual and their traits is a function of the environment including the community.

Populations contain individuals of the same species that are interacting among themselves. There may be multiple populations of the same species type within a community or ecosystem, hence a population in this model specifically describes a type of membership and group isolation. A population may be the result of an initial colonisation or migration process. Within a population it is possible to have trait related substructure such as juvenile individuals, breeding adults and non-breeding elders.

A species defines a group of individuals that are similar enough to reproduce, and so this also identifies potential competition between potential mates for reproduction. The definition of a species can also be common among different populations or communities, such that species individuals can migrate between organisational groups and integrate.

Communities have a specific environment, population subsystems and species. Rules govern the interaction, occupancy, boundaries, limits of individuals, time and resource allocation including space or territory based competition. The community environment can be the search domain. Individuals of several populations may interact (cooperatively or competitively) in order to form a solution vector for the search domain space. Species within a community coevolve and coadapt.

An ecosystem also includes an environment, and may contain communities, population, and nested ecosystems. A collection of species can be specified and shared between subsystems. Individuals of subsystems interact based on rules, and may migrate between compatible subsystems according to protocol. The notion of an energy quota can be used to limit resource allocation with which to adapt. Subsystems may adapt concurrently or serially based on a specific order. The environment does not need to be static.

Each organisation structure is considered in more detail. Chapter 6 looks at simple single species population models, with a variety of population topology models on a selection of well known search domains, in order to investigate a number of population topology related questions. In Chapter 7 both community and ecosystem organisational structures are considered with respect to opportunities for further research, in particular those focused around the influence of interaction topology between nested subsystems.

Although it is a deliberate goal of the ESEC model to contain a broad model of concepts and functionality, it is not able to contain all possible variations. It is hoped, however, that by creating a considered and inclusive model it will enable, within a specific context, the comparison of similar algorithms and systems, making it a valuable framework for research beyond the initial scope of the work presented within this thesis.

5.3 A Python Package: `esec`

5.3.1 Package Objectives

The ESEC model has been implemented as a Python programming language[5] package, named `esec`, and contains Python modules specifically designed to enable research into ecosystem based models of evolutionary computation. Even though the `esec` package has extensive support for different population, community and ecosystem models, it is also a robust and flexible platform for simpler traditional models of evolutionary computation.

The primary aim of the `esec` package was to support the strong topology related research objectives of this thesis, and while implementing the package and accomplishing the primary goal it became clear that other users could benefit from the features implemented. The `esec` package support a wide number of graph topologies through the use of the igraph[6] library. It is hoped that by providing the `esec` package under a liberal license that others, including undergraduate and postgraduate level students and researchers, will find it useful and a valuable contribution to ongoing research.

[5] See `http://www.python.org`.

[6] igraph is a free (GNU GPL) and well respected software library written in C for creating and analysing graphs. See `http://igraph.sourceforge.net`. A Python interface makes for simple integration with the `esec` package.

Appendix E contains a detailed presentation of the package features, architecture, language choice, dependencies, design motivation and integrated testing employed for the `esec` package. The package code is heavily documented at module, class and method level using integrated "docstring" features that are part of the Python language design. Package documentation is available in HTML format, and included as CDROM documents. Testing and verification of package features is extensive; software quality is an integrated part of the software development and maintenance life-cycle.

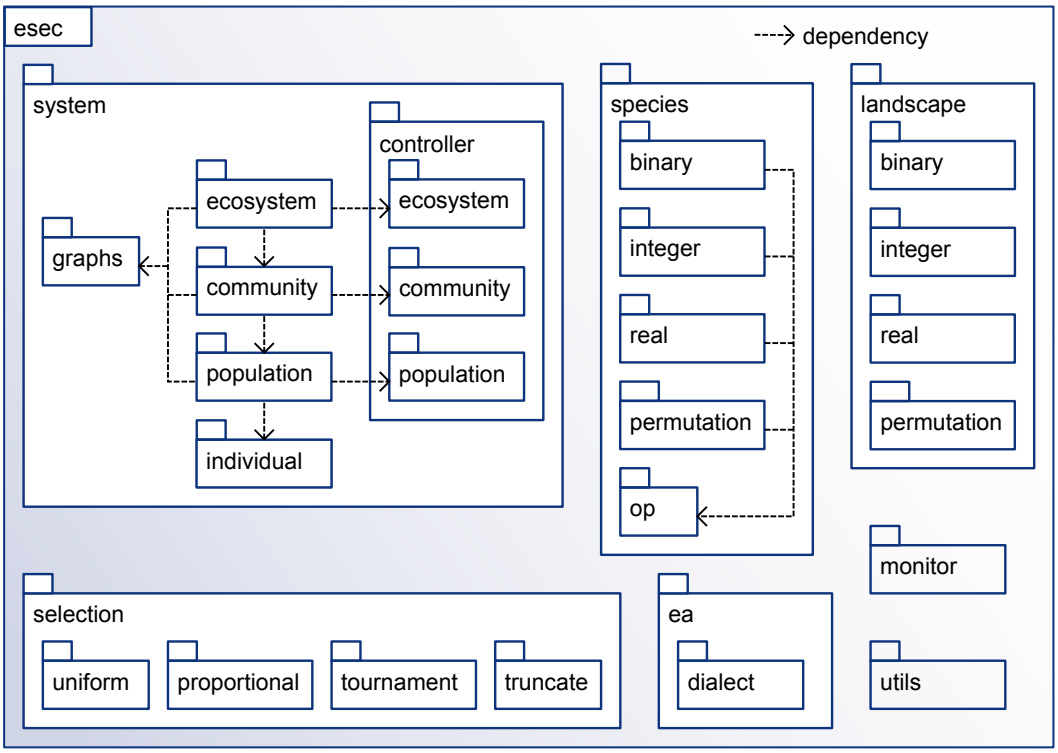


Figure 5.4: Relevant modules of the `esec` Python package. Not all modules of `esec` are included, and only some dependency relationships are shown which are particularly relevant to the organisational structure of the ESEC model.

Figure 5.4 shows the modules of the `esec` Python package, as well as some of the strong dependency relationships that are particularly relevant to the organisational structure of the ESEC model. The package modules correlate directly to the configuration tree data structure levels used to specify the configuration of a system.

5.3.2 Configuration

Syntax Dictionary

To create a flexible and supportive configuration format, `esec` uses a Python dictionary[7] data structure. There are two stages to configuration: firstly specify configuration syntax (within `esec`), and secondly specify configuration values (by the user).

A Python dictionary data type is simply written with a pair of curly braces "`{...}`". Dictionaries contain uniquely named elements and values. By convention within `esec` all

[7] Also known as a "map" or "associative" container in other programming languages.

name "keys" are strings. Keys must be unique, however key names can be repeated within nested dictionaries.

When a configuration dictionary is used by `esec` all named keys and values are checked against matching syntax dictionaries. A full syntax dictionary is a composition of abstract base and concrete final dictionaries, many of which are selected as the configuration details are processed. For example, the specification of a species genome type is used to select the syntax used for recombination or mutation configuration details.

Listing 5.1: Syntax and value specification for the key1 element

```
syntax = { 'key1': int, ... } # specify allowed key and value type
config = { 'key1': 123, ... } # value of specified type
```

An example of syntax and value specification for a simple element is shown in Listing 5.1. A small set of simple types and values used to define what type of value a configuration element might hold is shown in Table 5.1.

Syntax Type	Description
`str`	String
`int`	Integer number
`float`	Floating point number
`bool`	Boolean true/false
`dict`	Nested dictionary of details (unspecified)
`list`	Nested list of values (unspecified type)
`None`	None
`(...)`	Explicit set (tuple) of values (strings)
`[...]`	List of alternative value types
`{...}`	Nested syntax dictionary (specified)

Table 5.1: `esec` supported syntax dictionary types and values

The syntax can also specify that the configuration value must be one from an explicitly stated set of values (typically a set of string identifiers). As an example `{'type': ('binary', 'real', 'integer')}` is used as part of species configuration to indicate that the species "type" must be one of the set of stated strings ("binary","real" or "integer").

The element syntax also supports alternative value types using a list, written in Python using square brackets `[...]`. For example, `{'selection': [str,dict,None] }` indicates that the "selection" element value may be a string, a nested dictionary, or explicitly "None".

Named keys of the syntax dictionary are required in a configuration unless an element is indicated as optional, by placing a "?" at the end of the key name. For example `{ 'unique?': bool }` indicates that the "unique" key and value does not need to be specified as it does not apply in some situations.

Structure

In order to apply the `esec` package to a search domain, an `esec.Application` class instance is created and provided with a configuration dictionary. The package provides special handling of dictionary objects as configuration dictionaries, which enables the configuration syntax to be specified by various classes, default values to be stated, validation of provided details against the known syntax, and compositional overlay of successive configuration dictionaries. This approach provides many benefits. For example, a configuration dictionary from one experiment can be used as a base for other experiments which need only specify new or alternative details which are overlaid on top of the initial details.

The `Application` class instance contains several components: a *system* (which may contain a group of systems), the specification of a single or multiple *species*, the search domain *landscape* to which species are applied, and *application* level details which are used to control run and termination conditions for experiments. Listing 5.2 shows the basic dictionary structure used for all configurations. Internal details for each nested dictionary may depend on other settings. For example, the exact components within the "`system`" configuration, and the number of species defined, depend on the system type.

Listing 5.2: Structure and syntax for an EA ESEC configuration

```
cfg = {
    'EA': str, # string - name identifier
    'system': {
        ... # population, community or ecosystem
        ... # may include topology for breeding, survival etc.
    },
    'species': {
        ... # genome, recombination and mutation details
    },
    'landscape': {
        ... # search domain details (environment)
    },
    'application': {
        ... # seed, run stop conditions etc
    },
}
```

Species

The species configuration (Listing 5.3) begins[8] with the definition of its type (such as classic binary, integer or real gene values), initialisation method (which is typically random, and may be seeded or based on a distribution of values), and the method of species evaluation. In many cases evaluation is simply the substitution of the gene values as a vector to the search domain landscape. Other forms of mapping (such as binary to real values) or complex expression mechanisms can also be used.

As part of the species configuration, suitable recombination and mutation details may also be needed. For models that do not use a specific form of recombination (such as crossover) a simple clone operator is used instead; recombine and clone operators are

[8]The order of elements within a configuration dictionary does not (by definition) matter, however for the purpose of discussion a specific order is often useful.

typically exclusive of each other. Some aspects of the recombination operation imply an interaction topology between individuals of the same species, as in the selection of a parent and other mate individuals. It is the role of the system to supply appropriate individuals, based on the system configuration, that the species reproduction operations may need.

Listing 5.3: Structure and syntax of species configuration

```
'species': {
    'genome': {
        'type': str, # 'binary', 'real' etc
        'init': str, # eg. 'random',
        'eval': str, # 'simple', 'cached' etc
    },
    'recombine': {
        'type': str, # eg. 'one_point' crossover
        'rate': [float,str], # float range [0.0-1.0]
        'parents': int, # no. used in operation
        'offspring': int, # as a result
        'parent': {
            'selection': [str,dict], # order based?
        },
        'mate': {
            'selection': [str,dict], # parent breeds with
        },
    },
    'mutate': {
        'type': str, # genome specific, eg 'bitflip'
        'rate': [float,str], # float value or string
    },
    'clone': { # recombine alternative
            'rate': [float,str],
    }
},
```

Landscape

The role of the landscape configuration is to define the specific parameters of the search domain, abstracted from the concerns of any underlying system or species. Some landscapes are simply incompatible with the representation abilities of some species (an invalid configuration), and some systems may be compositions of multiple species.

With respect to object oriented programming ideas it is the landscape's role to evaluate solutions, however it does this without specific knowledge of individuals or their particular composition of traits. The trait values of one or many individuals are passed to a shared landscape instance for evaluation using an adapter or facade style of software pattern; the landscape is simply called upon to evaluate a vector of trait values.

When an individual is called upon to present its internal trait values as a landscape suitable vector, the individual makes use of a shared species object. The species object knows how to manage any required complexities and can also make use of the container system for collaboration if needed. In this way it is possible to for multiple species to be applied to multiple-objective landscapes via system level collaboration.

Listing 5.4 shows the basic configuration structure for a landscape. Specific landscapes may require unique parameters, and so are not general, but there are often similar

parameters shared among groups of landscapes. The most common shared details are "parameters" (dimensionality), range values (as upper/lower "bounds"), output offset and landscape inversion (invert "true"/"false").

Listing 5.4: Structure of landscape configuration

```
'landscape': {
    'name': str, # unique string identifier
    'type': str, # domain type eg. 'IVP', 'BVP'
    'seed': int, # optional integer value
    'invert': bool, # output inversion
    'offset': [int,float], # float/integer value
},
```

When several species are sampled and combined from several populations for evaluation, a specification of the inter-species interactions is required. It may be a simple model that uses representative sample individuals (ie. random or best), and partitioning of populations to localised interactions, or a complete and typically expensive exhaustive combination of each individual with all other combinations from other species.

For classic minimum optimisation problems it is often important to specify a stop resolution limit for use as a termination condition.[9]

The dimensionality n of many classic real-value test functions is set as a means of specifying landscape difficulty. Although any explicit value for n could be specified, as a convenience several sizes can be specified using a short notation of $n\#$ where "#" is the value of n.

A large selection of established and classic test problem domains are available within the `esec.landscape` module. For a complete listing see the `esec` documentation which is generated directly from the Python source code. All of the benchmark problem domains described in Appendix B have been included within the current `esec` package.

System

Population systems contain details for population topology and the breeding, survival and replacement of individuals. See Listing 5.5 for the syntax required for a population. Note that the value of the system type is not always a `'population'` as there can be specialised subtypes which alter the exact configuration of elements used.

[9] The `esec` package is designed to maximise solution value. The objective in many classic optimisation problems is to minimise to a zero value. In these cases the landscape is inverted and the resolution limit is set to a small negative number.

Listing 5.5: Synatx for population system configuration

```
'system': {
    'type': str, # eg. 'population' or descendent
    'topology': [str,dict], # ie 'panmictic' or details
    'size': [int,'#topology'], # value or determined
    'breed': {
        'size': int, # children created
    },
    'replace': {
        'group': str, # who to replace. eg 'parent'
        'selection': [str,None],
        'compete' : bool, # True = replace "only if better"
        'per_gap': int, # gap interval
    },
    'survive': {
        'group': str, # 'offspring' = non-overlapping competition
        'selection': [str,dict], # eg 'best', or dict details
        'size': int, # no. of survivors
    },
},
```

The top level syntax for both community and ecosystem system types are identical. (See Listing 5.6.) Communities and ecosystems are composed of nested subsystems, where the `'interact'` policy details interaction topology and settings. In the case of a community, the subsystems are specifically populations, while an ecosystem can be composed of population, community and nested ecosystems. It is possible to specify multiple instances of each of the specified subsystem configurations, using a list of integer values for `'count'`.

Listing 5.6: Structure of community and ecosystem configuration

```
'system': {
    'type': str, # 'community' or 'ecosystem'
    'systems': list, # list of subsystems
    'count?': [int,list], # repeat subsystems?
    'interact': dict, # interaction policy details
    'quota?': [list,None], # quota per subsystem?
    'order': str, # eg. 'sequence'
},
# Multiple species
'species': list, # list/nested list of subsystem species
```

In keeping with ecological principles an allocation of energy can be specified using quota values. The list of quota values is mapped to each subsystem on a per-specification or per-instance basis depending on the `'count'` specification. Quota integer values are used as a simple weighting of time given to each subsystem, with several options for the order in which the quota of energy resources allocated. It is possible for subsystems to operate concurrently, however this is not detailed or explored in the work presented here.

Application

An application instance can be given settings to control a number of "run" simulations using the configuration. (See Listing 5.7.) For each simulation run the system is reset then executed until one of the specified termination conditions are meet. Progress and success

reporting are presented as specified by the `report` value. Each run can be seeded using the optional integer random seed value (`rand_seed`) plus an integer increment value so that each run is seeded uniquely.

Listing 5.7: Structure of application configuration

```
'application': {
    'rand_seed?': int,  # eg. 12345,
    'run_count': int,   # eg. 5,
    # termination conditions?
    'run_stop': str,    # eg. 'gen+res+fixed',
    'gen_limit?': int,  # generation limit
    'birth_limit?': int,# new individuals
    'eval_limit?': int, # evaluation calls
    'res_limit?': float,# eg. -0.01, -0.0001 etc
    # progress monitor / end of run report?
    'report': str,      # eg. 'brief,best',
},
```

The exact conditions of termination are specified with a string `run_stop` value which is composed of tag substrings joined by the "+" character. There are currently five valid tags, four of which use other application keys to set limit values; `gen` for generation limit (in `gen_limit`), `birth` for birth limit (in `birth_limit`), `eval` for evaluation limit (in `eval_limit`), `res` for resolution limit (in `res_limit`), and lastly `fixed` which is a population diversity convergence test as defined by the particular species involved.

5.3.3 Batch Experiments and Reports

To facilitate investigations, a batch specification and execution script (`run.py`) allows a collection of unique experiment configurations to be specified (as a list), and each configuration to be run multiple times according to the settings of the `application` section. Each configuration is tagged using labels and given a unique batch identification. The batch result data is stored and conveniently tagged, which also facilities filtering and presentation of results. If required a subset of the batch configurations can be run (or re-run), and configuration settings overridden, using command line arguments to the execution script.

Batch result files can be automatically compressed into single summary files. There is also an integrated report generation process using a companion script (`report.py`) that uses the batch file details. Comparison methods and result presentation supported by the reporting script is discussed in Section 5.6.

Batch configuration files are written in Python allowing the full programming language to be used. This includes being able to import an existing configuration and modify it without restating all details. As the extension and alteration abilities can obscure the exact configuration details used when running a particular application instance, the complete configuration details used by each configuration are saved with batch result data.

Programatically, a batch file must define a "`batch`" function which returns a `list` of configuration details (dictionaries). A defined report function ("`report`") is used by the report generation script (`report.py`). Similarly, support for comparing results across multiple batch file results is provided by defining "`multi_batch`" (which supplied a list

of how the batches are compared) and "`multi_report`" (which specifies how the reports are to be shown) functions within the batch file.

See Appendix F for a listing of the HTML and PDF reports generated by batch configuration details, and included as CDROM documents.

5.4 Consideration of Related Work

5.4.1 Introduction

With respect to the ESEC model, there is a wide range of work within the field of evolutionary computation that is related. Of particular relevance are models that used specific ideas from ecology, such as interaction structures and topology in relation to populations, groups and individuals.

Evolutionary algorithms are based on the ecological ideas of population based variation, genetic encoding, reproduction and recombination, and fitness-based selection pressure. The majority of EA models, and in particular early origin and canonical models, utilised simple genetic representation, interaction and population structures. Indeed, for simple search domains there are clearly strong disincentives to create or utilise a more complicated or complex model of the essential evolutionary algorithm concepts.

Section 5.4.2 first considers explicit niche schemes that can be applied to simple single population models. Next, Section 5.4.3 looks at the range of established structured population models, including the prevalent distributed and cellular EA models. As the ESEC model uses the notion of *community* to contain multiple interacting species, Section 5.4.4 looks at relevant coevolution EA models.

5.4.2 Explicit Niche Schemes

The development and specification of explicit niche behaviour within a single population model is an elegant use of biological metaphor applied to EA search. Whenever an EA based search uses a global selection method (based on a simple full-graph population topology), it is possible for "lethal" suboptimal solutions or trait components to be propagated excessively and effectively stifle genetic variation and exploitation of diverse regions of the search space. Explicit niche schemes are a popular way of limiting the influence of lethal traits, as well as a means of encouraging the exploitation of unique niche search space regions.

The two dominant methods of explicit niche scheme are fitness sharing [142] and crowding [74], and as mentioned in Section 3.2.5 of Chapter 3, there are various selection-based techniques that can be used to create or maintain niches [354, 233, 155, 54, 229].

In fitness sharing the reproductive "fitness" of each individual is adjusted in proportion to the number of individuals with membership of (within a small "distance") the same "niche" location. The distance is ideally based on a genotype measure, but if this is difficult or not possible the fitness value can be used. Parameters need to be specified for the weighting of the proportional relationship (typically linear), and for a distance threshold (share radius) which essentially specifies the number of niches that will be supported [98].

Because fitness sharing utilises fitness proportional selection, the greater the relative fitness of a particular niche location (in the solution space), the more individuals that can be supported by such a niche. This supports the notion of exploitation of known fit location of the search space, but can also increase the risk of lethal traits and overall premature convergence.

Crowding is a diversity preservation technique originally proposed by De Jong for steady-state ("gap") models [74]. By identifying "similar" individuals (based on genome details rather than fitness value) offspring replace the most similar members of the current "parent" generation. Deterministic crowding is an improvement proposed by Mahfoud [222], which is based on likelihood that offspring are similar to their parents, and so offspring compete with their parents for survival. In this way offspring still replace an individual that they are similar to without the need for a large number of genome comparisons.

Both fitness sharing and crowding use an individuals' properties to modify interaction between individuals. This effectively creates a dynamic interaction topology as an emergent structure within a simple panmictic population topology; the complexity of the interaction topology does not need to be prescribed using a top-down approach, but emerges from a bottom-up process. Distinctively, while fitness sharing is a fitness-proportional exploitation technique, crowding is a means of maintaining diversity within a population and ideally supports exploration.

5.4.3 Structured EAs

Established Models

Of ecosystem models and population topology models, variation of population topology is the aspect most explored early in the field. (See reviews such as [263, 7, 51, 6].) The selection of various population structures has undoubtedly been related to the emergence of distributed computation hardware. Well cited reviews and classification models of structured evolutionary algorithms (suitable for parallel implementation), have already been presented and discussed in Section 3.2.10 and Section 3.3.6. The established structured EA (sEA) models include the following:

- **Global:** This includes distributed evaluation "master/slave" models and master-directed sub-population "breeding/evaluation". Both model types do well when communication is costly (and should be avoided), while facilitating the beneficial use of distributed evaluation.
- **Coarse-grained:** Typified by sparsely connected subpopulation "islands" ("demes") with infrequent migration. This model that works well when distributed resources are of mixed capacity (in terms of processing or communication ability).
- **Fine-grained:** A regular and typically spatial distributed ("cellular") population model with localised overlapping neighbourhoods. This creates very high commu-

nication requirements between neighbours, and so applies nicely to shared memory vector processing architectures.

- **Hybrid:** A model that combines structured models at two-or-more levels.

Global

A global and coarse-grained structure works well when computationally expensive operations can be algorithmically divided. The well-established trade off [51] for both models is communication cost; the benefit of parallel execution must compensate for the cost of transferring information from a master to slaves, or between coarse-grained subpopulation groups. Coarse-grained models require less communication between distributed groups than a single population model distributed by a master-slave arrangement.

Coarse-grained

The island model or "island migration" model (discussed previously in Section 3.4.6) can be described as a coarse-grained parallel EA, where multiple "island" populations (or "demes") evolve concurrently with infrequent communication between islands.

It is not surprising that interest in this type of distributed EA grew in the 1980s with the availability and development of parallel computing hardware, and with the ideal goal of evolving diverse solutions as an advantage for multimodal search domains [98].

Island models [243] have been proposed and investigated in many different forms, and there have been investigations related to questions such as appropriate interaction models, island sizes and convergence behaviour [367], and the potential for efficient search using cellular (fine-grained) island EAs [251]. See the work of Cantú-Paz [49, 50, 51, 52] which has looked specifically for accurate and efficient parallel EA models, in particular variations in migration policies and selection pressure, and the more recent contributions of [320, 319] for analysis of island models.

An island EA supports the early work of Eldredge and Gould [101] and the theory of *punctuated equilibria* as discussed earlier in Section 3.2.10 in its relation to EAs, and also Holland's early formulation of the GA as a trade-off between exploration and exploitation [166].

Coarse-grained models, for a single species, are represented in the ESEC model as an *ecosystem* organisation composed of multiple (common species) *populations*. Each population evolves is an isolated sense, and any migration between populations is effected by the ecosystem and its rules.

Fine-grained

There are many names and classifications given to "fine-grained" population topology models. See the classic work of "parallel" and "fine-grained" EA models of Mühlenbein [245, 247], Gorges-Schleuter [144, 145], Manderick and Spiessens [224, 327, 328], and Davidor [65, 66].

In all cases a single population topology is divided into a number of spatially arranged and overlapping neighbourhoods[10] [98, 19]. Each location in the population habitat can be considered a "cell" of the overall "cellular" structure. The term "diffusion" is perhaps one of the most appropriate [98], as it describes the limited (diffused) progressive transfer of traits, even when a trait is strongly selected.

As a general model, within a generation each individual in a fine-grained cellular structure is given the chance to compete and reproduce only within its localised neighbourhood. This includes operations for the selection of parents, the generation of offspring and the selection of survivors. Survivor offspring may replace the original parent cell individual or alternatively compete for survival with individuals of the surrounding neighbourhood.

The order that individuals (or alternatively "deme" groups) are selected and updated can be varied. Updates can be performed asynchronously, particularly on hardware that supports this. As long as localised operations have a scheme to resolve concurrency issues, this is another useful distributed processing model.

Selection of the cell to be updated is commonly done using uniform random or a simple sequential order. Mate selection for reproduction with the selected "parent" is based on a localised selection scheme. Well known global selection schemes such as tournament or fitness proportional selection can be easily adapted to a local cell environment. Methods that involve global sorting or ranking are typically avoided due to the cost of repeating such calculations with generation-gap (steady-state) replacement.

It is common for the central parent individual to be replaced, but it is also possible to apply additional selection pressure for survival. For example the parent may only be replaced by the offspring if the offspring is better or equal to the parents' level of fitness. Similarly, a neighbourhood or mate individual may be replaced by the offspring, especially if there are multiple offspring created during reproduction.

Davidor developed [65] and presented [67, 66] an "Ecological Genetic Algorithm paradigm" (ECO GA) that is based on a cellular fine-grained GA, with some additional specific ecological principles. The five ecological principles are restated here:

1. An organism's life is characterised by three activities; foraging, reproduction and survival.

2. Mates for an organism are selected from a local environment, and in proportion to the potential mate's fitness.

3. Offspring remain in the immediate vicinity of parents.

4. The frequency of aggressive conflicts is inversely proportional to availability of resources.

5. Aggressive conflicts are resolved probabilistically and in proportion to the relative strengths of opponents.

[10] Some authors describe each neighbourhood as a subpopulation. This thesis does not use the term "subpopulation" in this sense to avoid confusion with community subpopulations.

The ECO GA model uses a cellular population, a steady-state (variable gap size) reproduction model for mate selection, reproductive fertility is based on the relative fitness ratio, and replacement is a local random target where fitter children are always retained, and weaker children have a fitness proportional chance of survival. It was observed in [65] that the probabilistic survival of weak children allows "island" niches of diversity to develop, and in this regard is somewhat similar in effect to preselection mechanisms and crowding schemes [74].

Fine-grained population structure does not need to be restricted to a simple 2D lattice topology. The common use of a toroidal structure eliminates non-homogeneous connections while also reducing the mean path length between cells. Non-homogeneous topology, such as neighbourhood size and the number of cell connections (degree), create interesting environments and different localised competitive strengths.

Localised selection pressure is generally weaker than an equivalent global scheme [78]. Sharma developed a well cited series of investigations [300, 299, 301] on structured (lattice) topologies, including the influence of various selection methods with different neighbourhood topology sizes. One of many interesting results presented is that propagation time of traits across a grid seemed to relate to the grid size rather than the connection distance, suggesting that mean path length measure of a topology may not be a dominant factor influencing evolutionary search processes.

Alba and Troya [8], working along similar lines to Sharma and earlier work [144], looked at cellular EA models (cEA) and proposed an adjustable parameter for the ratio of neighbourhood size and global lattice size. Their results showed that the ratio between radii, and the use of a dynamic change in radii during evolution, did influence search outcomes. The radii could be adjusted to alter effective selection pressure and optimise the population topology to suit different search domains and different stages of evolutionary progress.

Fine-grained structured EA are represented in the ESEC model as a single *population* organisation. The topology for the population can be specified and altered, which in turn influences the interaction and pressure for each individual.

Hybrid

Hybrid models are almost universally implemented in an attempt to utilise good features from single or fine-grained models as well as coarse-grained models. Opportunistic Evolution (OE) [333] is a good example in that it uses an adaptive asynchronous global model to control and adapt coarse-grained islands. Each island (typically a specific machine) performs a "mini" evolution process for a period of time weighted to justify the communication and setup costs. As discussed by the authors, the advantage of this approach is that unlike a basic global model of distributed evaluation, where evaluation time needs to be long enough to justify communication costs, the OE adaptively performs enough evaluation on each island to justify the costs. This model is also interesting in that it represents, and adapts to, ecological concepts of resource limits (ie. quota).

There is good evidence that the advantage of structured models is greater than simply

a divided computation workload. For example, results have suggested [355] that breaking a population into smaller evolutionary portions is an advantage for compositional problems, where a full solution is formed by smaller sub-solutions. This requires that sub-solution parts can be discovered independently, at least partially, in order for selection to be effective. It has been reported also that compositional problems are specifically amenable to island models [319] and coevolution.

Hybrid models can be represented in the ESEC model by the *ecosystem* organisation, which in turn is able to contain populations, communities (of different species populations) and (possibly complex) nested ecosystems.

5.4.4 Communities of Species

Introduction

While most EA models adapt individuals of a single species (as is appropriate for many search domain applications), the examples of biology are almost exclusively filled with multiple species in competition or cooperation with each other. Species coevolve and coadapt, inhabiting niches that are influenced by the ecosystem composition. Interactions are not a simple panmixia, but a mixture of mostly local and infrequent extended interaction. Organisms of complex biological species are rarely static; mobility is key to individual and species survival.

As presented in Section 5.2.5, the ESEC model defines a *community* as having a specific environment, population subsystems and species, as well the necessary rules that govern interactions between and within species. As individuals of each species ca ninteract, a process of coevolution takes place.

For clarity in this thesis the term "species" is reserved only for a group of individuals that are structurally similar and able to interbreed with each other. This is in contrast to the general EC field, where it is not uncommon for the terminology of "species" to be used to describe similar subgroups of individuals that can emerge within a single structured EA population.

Fitness sharing and crowding are two explicit niching techniques discussed previously in Chapter 3 and Section 5.4.2. As niche techniques manipulate the interaction between individuals using either genotype or phenotype details, and can so isolate individuals into different groups, explicit niching effectively creates subspecies within a population. However, this type of multiple-species evolution is not treated in the *community* organisation of the ESEC model, but rather as part of the *population* organisation and implemented using established fitness evaluation and selection.

In a related manner, fine-grained population models (such as cellular or lattice topology models) are also often discussed in terms of diversity and species, as localised regions are able to evolve and specialise traits in relative isolation. A single population notion of subspecies formation and diversity is contained within the ESEC *population* organisation as discussed earlier.

There are many interesting EA models that have used explicit communities composed of multiple unique species, often coevolving. The next subsection considers some of the

earliest models of coevolution used to develop classifier rules. Following this the models of deliberate competitive and cooperative coevolution are considered in more detail.

Coevolving Classifier Rules

One of the earliest examples of community EAs is an artificial coevolutionary search approach by Holland and Reitman [170, 167]. In their model a population of stimulus-response rules for a *classifier system* were evolved. Similar approaches to rule evolution were also developed by Smith [323]. Later developments in the field presented richer rule and classifier models, such as the work by Grefenstette and colleagues [149, 151].

Competitive Coevolution

One of the most well known examples of artificial coevolutionary search is that of Hillis [162, 163], developed to evolve minimal sorting networks. The initial population model used a 2D lattice grid population topology containing 64536 individuals (256×256). When using only a single species to represent networks, Hillis noted that the evolutionary search process converged prematurely. A second "parasite" species was introduced which represented test cases for the "host" sorting networks.

The fitness of each host was a measure of how well they were able to sort the local parasite examples, and conversely parasites were rewarded for exposing flaws in a host's sorting ability. Both host and parasite species started at a simple evolutionary level of complexity, and as coevolution progressed an artificial "arms race" of incremental (and distributed) development discovered some excellent, and human competitive, sorting network results. Another noted benefit of the model is that it avoids evaluation time by not testing trivial test cases.

Other early models of competitive coevolution include Holland's *Echo* models [168, 169] which use predator-prey ideas, and also Ray's *Tierra* model, which was created as an "evolutionary approach to synthetic biology" [287] and included features such as host and parasite species.

Rosin and Belew [295, 296] created competitive coevolutionary models and applied them to a number of domains, including evolving strategies for competitive game play such as tic-tac-toe, nim and go. In [295] their approach used two species to represent opponents of a game. Unlike lattice based competition selection, Rosin and Belew created a *shared sampling* approach which selects a sample of opposition individuals from the previous generation as competitors for the current generation, with a bias for those individuals that proved difficult to defeat. A *competitive fitness sharing* function is used to reward individuals based on the number of opponents they defeat.

There are many other examples of evolving game playing strategies. For example the early work by Reed et al. [291], Pollack and Blair coevolution of a backgammon player [276], and Fogel and Chellapilla's well known report of the checkers playing program "Blondie24" [112] which evolved weights for an artificial neural network based on interaction with online human players.

Cooperative Coevolution

Husbands created a distributed cooperative coevolution model and applied it to multi-criteria and multi-constraint job shop scheduling (JSS) problems [172]. Job-shop problems can be examples of complex multi-objective constraint satisfaction problems where a number of "jobs" need to be performed, and each requires a number of unique "machines" for a set number of operations. Different jobs require a different number of operations. A common coevolutionary approach is to create a different species for each job and use a lattice model to distribute, evaluate and coevolve the species. As job species can conflict when combined, Husband's approach was to use an "arbitrator" species, which also evolved, to resolve issues.

The system was able to evolve both high quality solutions (schedules with a low overall time) in a relative short evolutionary time and with a robust level of genetic diversity. Although the exmaple is a very problem specific there are a number of interesting high-level results. Two examples are that "take-over" by single good species can and will occur, and that the diffusion lattice does assist in diversifying the search.

The division of a problem domain into individual species is a common approach for cooperative coevolution. However, as Bull and Fogarty noted in [44], when inappropriate divisions of the solution representation are made search performance is adversely affected in relation to problem complexity.

In [271] Paredis developed a *symbiotic* (cooperative) two-species model and applied it to a deceptive problem (see [134]). One species represented a full solution, while the other species represented a permutation. Full graph population structures were used for each species, and a simple random selection approach used to form collaborations for evaluation.

De Jong, Potter and colleagues [78, 278, 281] developed a novel and well cited model for Cooperative Coevolution. The stated objective for the model was to search for solutions to complex problems. As Potter explains in [278], a fine-grained model alone "is not sufficient" to support coadapted subcomponents, which is one of the key features and possible advantages of a coevolutionary model.

The model used an explicit "divide-and-conquer" strategy where a search domain solution is divided into a number of sub-species (modules), evolved independently and evaluated using temporary collaborations. The fitness of each sub-species individual is based on its collaboration performance. Later work [281] also investigated dynamic speciation and species extinction; new species are created in response to ecosystem stagnation, while species providing only minor or insignificant contributions are removed.

While there are many possible collaboration strategies and Potter and De Jong initially considered two alternatives: a *greedy collaborative* approach where only the best of other collaborating species is used, and a *less greedy collaborative* approach where the best of one species collaborates with other randomly selected individuals from each species. Their work found that the greedy approach was less robust than the less greedy approach, in particular for domains where there is a lot of interaction or interdependency between sub-species. As Kirley later commented [196] an important criteria for the selection of an

effective collaboration model is that it "preserves and integrates information coming from each species". Conceptually, a greedy approach is narrow and greatly restricts the range of information that can be integrated from multiple species.

The cooperative coevolutionary model of De Jong and Potter is well known and has been successfully applied to a range of domains [78]. Other applications included the development of neural networks [279], concept learning using an antibody metaphor [280], and later string covering problems and the evolution of cascade networks. Potter's thesis [278] states a number of questions and further research directions, in particular the consideration of alternative collaboration, ecological relationships and speciation models.

Juillé in [179] and also Juillé and Pollack in [180, 181, 182] reported several successful applications of an "incremental" coevolutionary approach to develop solutions for "complex systems", including classifier network solutions for the intertwined spirals problem, and coevolving an "ideal trainer" for use with cellular automata rules.

Moriarty and Miikkulainen [244] created an adaptive cooperative coevolution model and used it to create neural networks as an alternative to standard evolutionary algorithms. They state that their work was based on the common hypothesis that a coevolutionary approach could be more efficient than single species search, as different species can search in parallel for components of the problem domain.

Kirley [193, 196] developed a spatial cooperative coevolutionary model (SCCA), using a cellular lattice where each lattice cell contains multiple species (cohabitation), and collaborations are limited to each cell neighbourhood. Sub-population species are evolved in parallel rather than in a sequential or "round-robin" manner. In this way individuals are pressured to both cooperate with other species, and to compete with members of their own species.

Two approaches to collaborator selection for each cell ("site") were investigated: cell only species collaboration (denoted SCCA 1), and local neighbourhood collaboration where a random species of the cell is selected and the best neighbour for each other species is selected (SCCA 2). Although results for classic function optimisation problem did not indicate a particular significant advantages for SCCA (1 or 2), a more difficult NP-complete job shop scheduling problem indicated that SCCA 2 was the preferred collaboration strategy, and overall results were comparable to other evolutionary algorithms.

The spatial cooperative model has been used by Kirley and colleagues as a basis for other ecosystem based extensions including meta-population models, dynamic disturbances, extinction (catastrophe) events and speciation processes [146, 197, 198, 194].

Watson and colleagues [356, 357], and in particular the work presented in Watson's PhD thesis [355], investigated how methods based on abstract symbiotic processes influence evolutionary search. In their model simple low-level modules or "building blocks" ([166]) are used to form high-level complex "aggregations". The work presented a number of symbiotic processes for combining modules, explicit approaches to ensure that modules coadapt and cover complementary parts of the problem domain, and module level credit-assignment methods of fitness. This is in contrast to methods that need explicit subcomponents (such as [152]).

Coevolution has also been applied to robotics for the development of behaviours and controllers for both simulated and physical constructs. (See [317, 261, 260].) In particular, see the work of Nolfi and Floreano [262], and related work on modular component development and evolution (with applications to robotics) by Calabretta and colleagues [45, 46, 48, 47].

An *endosymbiotic* evolutionary algorithm (EEA) was proposed by Kim et al. [192] as an extension of earlier work by Bull and Fogarty [44]. The endosymbiotic model is based on observations in cellular biology where relatively large and complete eukaryote cells appear to be "hosts" containing smaller eukaryotic cells that have been consumed by host cells, and yet survived to become an integrated part of the overall cell. The EEA model supports the independent coevolution of prokaryote-like sub-components, the "horizontal" transfer of genes from prokaryotes to be combined as "complete" eukaryotes solutions, and the endosymbiotic level evolution of the complete eukaryotes.

Kim et al. compared their EEA model to existing symbiosis models on classic problems, including the well known NKC test domain ([187]) and a string matching problem composed of substrings, with good results for the EEA over other EA in most cases. Interestingly, the authors classify existing symbiosis models into two categories that relate directly to the population topology distinctions used in this thesis: separated and population-based symbiotic evolutionary algorithms (SPA), and separated and neighbourhood-based symbiotic evolutionary algorithms (SNA).

Coevolution and Success

Paredis [272] investigated the "Red Queen" principle in relation to earlier coevolution results. The "Red Queen"[11] principle was proposed by biologist Leigh Van Valen in 1973 [346] regarding the requirement for continual competitive development in evolutionary ecosystems. This principle suggests that species need to continually change in order to maintain a fitness relative to the system it is co-evolving within. Examples of this are seen in the "arms race" between competing species for a shared resource, and the interdependent predator-prey cycles.

The net effect of an arms race may be static relative fitness (the "Red Queen" dynamic), increasing fitness, or an overall decrease in fitness by all species, such that the entire system may converge to a sub-optimal state. For example, in a dense forest area, trees competing for light must expend large resources just to begin competition.

As a criticism of existing coevolutionary models, Werfel, Mitchell and Crutchfield [362] investigated in closer detail the coevolutionary success of Juillé and Pollack [180, 181] and accredited the success to *resource sharing* rather than the coevolutionary mechanisms.

Pagie and Mitchell [270] presented a comparison of evolutionary and coevolutionary search results. They looked at coevolution using either one or multiple populations, such as competition within a population ([296, 12, 317]), or where an individual's fitness is based on comparitive context with individuals of other populations (such as [162, 180, 272, 268]).

[11] The title is in reference to a comment made by the Red Queen character, in Lewis Carroll's "Through the Looking Glass", who says "... in this place it takes all the running you can do, to keep in the same place"[53].

They restated the theorised benefits of several coevolutionary models: more efficient evaluation of evolving solutions [162], possible auto-adjustment of selection gradient [182], and the open-ended nature of coevolution [296, 106]. Successful coevolutionary applications to using spatial (cellular) structures include work such as [162, 172, 268]. Other unsuccessful coevolutionary applications are typically characterised by "red queen dynamics" [272, 314, 269, 182] and mediocre stable states [276, 182].

Pagie and Mitchell also noted from published works that the use of continuous mixing techniques can approximate the behaviour of a non-spatial (lattice) model. This result is also supported by the population topology investigation presented in Chapter 6 where random graphs performed similar to lattice topologies for many problem domains.

In their own investigations, Pagie and Mitchell state that coevolution is hoped to lead to an "arms race" condition of continual improvement in search. However it can also lead to a Red Queen dynamic model of moving without advantage, unnesscesary speciation or poor quality (mediocre) stable states. Later work by Mitchell and Williams [369], and also with Thomure [240], continued to investigate the success and role of "space" or spatial topology with respect to coevolutionary learning (search).

5.4.5 Summary

The ESEC model presents three organisation levels to represent existing and new ecological EA models: population, community and ecosystem. This section has considered work related to the ESEC model, and discussed how the ESEC model classifies, and can be used to represent, a range of existing models. In particular this includes explicit niche schemes, structured EAs and community EAs of coevolving species.

Chapter 7 looks at the ESEC community and ecosystem organisational models in more detail, which between them support the range of classic coevolutionary and coarse-grained structured EA models.

5.5 Key Questions

There is long history of biological and natural systems providing inspiration and influence for evolutionary computation.

Topology clearly has a role in natural examples of evolution at many different levels. To describe complex systems of interaction, the topology is often used as an abstracted "top-down" model of connections. However the origin of topology in most (if not all) natural systems is an emergent and adaptive "bottom-up" process of interaction and localised processes.

As a metaphor for artificial models, both top-down and bottom-up approaches are useful in particular contexts. For example, understanding what types of low-level interactions create top-level structural features can indicate the nature of system growth, emergence features, expected complexity and robustness. Similarly, being able to specify a top-level topology with particular qualities can be used to influence bottom-level interactions. The top-level abstraction is a means of specifying potentially complex system-level interac-

tion without the need for complex local interaction rules or computation; interactions are specified by the established topology.

Evolutionary computation also has a long history of models that have used both local bottom-up models of interaction, as well as top-down models of population (and inter-population) interaction [98, 77, 113]. (See Chapter 3 and Section 5.4.) The majority of topologies used are simple. The influence of topological complexity is likely to be highly sensitive to the specific processes (evolution) and objectives (problem domain) an EA is applied to.

There are several areas of questioning, inspired by ecology and complex topology models, that we can consider with respect to evolutionary computation and the ecosystem model presented here.

Questions arising from ideas within the biological origins of ecology:

- Does a model of limited resources and energy affect evolution?
- How does varying levels of resources, and survival pressure, affect solution complexity, interaction complexity, niche and speciation development, and interdependency?
- Should the majority of interactions be localised?
- How influential is mobility to the vitality or diversity of a population?
- Do complex environments require incremental competition for successful adaption?
- Are episodic successional evolutionary paths useful or occasionally essential to the development of robust and/or complex solutions?
- Will the presence of an age-influenced life hierarchy (such as juveniles and adult models) be useful or essential in the support or development of specific complex solutions?

The field of topology also presents many questions, especially with respect to simple compared to complex structure:

- Does complex topology influence EA search outcomes? If so, in what way?
- Does a solution to a complex search domain have any relationship or sensitivity to a complex population or evolution topology?
- How does simple structured topology compare to similarly resourced random or complex topology?
- What is the influence of a dynamic topology?
- Would a dynamic topology modified by local fitness based processes influence EA search outcomes?
- Are there niches where topology specified by stochastic parameters (as in rewired lattice structures) provide opportunity or support for processes that are not possible in simpler topology?

- Does a model of "occupancy" influence outcomes, in particular with respect to density regulated competition or significant disturbance events?
- Are motif structures important, and if so for what type of processes or outcomes?

To explore all of these questions, and many other similar and related topics, is outside the scope of this thesis, but the ESEC model presented does provide the capacity to explore most of these issues in detail. The wide range of open questions is a natural consequence of support for ecological processes, dynamic interaction and complex topology models.

It is important to distinguish between the importance of a specific domain solution (discovered by an instance of an EA search), as opposed to algorithm performance qualified by its typical efficiency and efficacy.

Almost all of the questions listed could be restated and prefaced with the words "In what way is the process and outcome of evolutionary search influenced by ...". The terms "what way", "process" and "outcome" need to be qualified, and in each question there would be additional terms that also need to be defined in order to construct a clear investigation and discuss results meaningfully:

- "what way" relates specifically to the "process" and "outcome" terms, and in particular it is desirable to be able to qualify any "good" or "bad" distinctions. For example, does a particular topology structure slow down, speed up or have no effect on the required evolution time with respect to a different topology type?
- "process" can relate to several of the processes involved in evolution, but in particular those related to population diversity, including mixing, convergence, localised niching, speciation support, and so on.
- "outcome" is specifically related to the quality of the best solutions found, but this may also related to the frequency (histogram) or population fitness, success rates of simulation runs, as well as success rates of mating.

5.6 Comparing Performance

5.6.1 Introduction

There are many measures and methods that have been used to compare the performance of different evolutionary algorithms. It is important to construct appropriate experiment scenarios, collect relevant data, and to make suitable comparisons that are relevant to experiment goals. Any use of statistical measures should enhance presentation and provide support for confidence and probability of outcomes.

5.6.2 Measurement and Concepts

For many of the experiments presented in later chapters, an EA application is applied to a number of different search landscapes and application parameters (such as population topology details) are varied. In general terms a number of simulation runs are performed

for each specific configuration and results collected. Complete reports (in HTML and PDF format) are generated that contain a number different tabular and graphical presentations of the results. All experiment reports are provided as electronic appendices.

Each report presents an initial set of summary tables and these can be used to compare a number of broad measures. These include two common measures of EA search success: Average Evaluations to Success (AES) and Success Rate (SR) values.

AES is the average number of evaluations each evolutionary algorithm (EA) instance (run), from a sample group of runs, required to achieve a search "success" result. Success is defined as a solution found that matches, or is within a value tolerance of, a known ideal solution value. If a single EA search run did not achieve a success result, due to either the population becoming "fixed" to a single genotype or a maximum limit is reached (such as evaluations or generations), then the result does not contribute to the AES value.

Success Rate (SR) percentage values are presented along side AES values. This gives both a qualification of the AES value, and an overall measure of success frequency. When SR results are less than 100% it is a fair indication that either the search domain is difficult or the EA instance has limited capability or resources.

It is interesting to compare groups of successful search run data. Many statistical tests assume normal distributions, and so a null hypothesis normal test can be done to see if, for a group of AES results, either the skew and/or the kurtosis differs from that expected in a normal distribution. This is presented as the two-tailed p-value ("norm") test, where a value less than a 0.05 is treated as an indication that the distribution is not normal. The result can often be seen in the skew or outliers features of AES data distribution, and in some cases can be an indication that a hard limit (such as the maximum number of generations allowed) is limiting the results.

It is also worth considering "fixed" run results. As a basic measure, the number of times a run becomes "fixed" can be compared to the number of "success" and "limit" results. A Mean Best Fitness ("MBF") value can be calculated as the average of the best value from all successful and fixed EA search runs, and excluded limit results. Although some "limit" fitness results are quite good (and perhaps better than "fixed" results), it is difficult to justify limit results in the MBF value as the search has been stopped in an arbitrary (and so less meaningful) sense.

A "Best Fitness Histogram" is a simple list that presents the range and frequency of best fitness values. This gives an indication of the variability of result value quality. Presentation is simply a comma separated set of fitness values (formatted) and their frequency values (separated by a colon), limited to a practical number of five instances. Greater than this, a number is used to indicate how many additional fitness value types were not displayed.[12]

[12] For example, "(+4)" indicates that four additional unique fitness values are also known but not shown in the list.

5.6.3 Box and Whisker Evaluation Plots

A box and whisker plot (also known simply as a whisker plot or boxplot) is used to visually and easily compare distribution of values, and in this case the number of function evaluations an EA run takes until success.

Graphically, data distribution is presented as a column with a box that extends from the lower to the upper quartile of the data, with the median (AES) indicated with a line. Whiskers extend up and down to show the range of data values that are not considered outliers, and flier (outlier) points are plotted as "+" character past the whiskers. Outliers are defined as values more than 1.5 times the quartile values.

Whisker plots provide an excellent overview of data distribution, especially when a number of distributions are compared as columns with a shared scale.

5.6.4 Mann-Whitney U Test Comparison Matrix

The Mann-Whitney U test [226] (MWU) is a well known non-parametric rank-sum test to assess if two independent samples come from the same distribution. Unlike the Student t-test which requires normal distributions, or the Wilcoxon Test [368] which requires equal sample sizes, the Mann-Whitney U Test does not have such restrictions and yet performs almost identically.

Interestingly the t-test has been widely used to compare EA search results in EC literature. However, as noted by the normal distribution test values from results data collected for this thesis, many sets of results should not be considered normal distributions. Although it has been shown that the t-test is fairly robust to a non-normal distribution, it is clear that the Mann-Whitney U test is the more appropriate and robust statistical method for non-uniform data distribution comparison.

In order to compare the success evaluation data for groups of results, a half matrix of MWU test p-values can be presented. Sample size greater than 20 is preferred (and required for the normal test) but > 5 is suitable. Of the p-values presented in reports, values < 0.05 are (arbitrarily) considered significant (and marked in bold and blue colour), although values near this threshold are also likely of interest. A p-value less then the threshold is an indication that the null hypothesis (that both sets of values are from the same distribution) should be rejected.

5.7 Closing

This chapter has presented an ecosystem model for evolutionary computation (ESEC) based on the observations and models of ecology and ecosystems, the established paradigm of biological evolution as a metaphor for evolutionary computation (EC), and supporting the explicit inclusion of structural topology to influence interactions at many different levels. ESEC is a compositional model that supports three distinct organisational systems: populations, communities and ecosystems.

A Python programming language package, named `esec`, has been developed to support the evaluation of the proposed ESEC model. The objectives and structure of the package

and its internal modules align closely with the ESEC model. The specification of `esec` configuration syntax and settings was described in general, and at a high level of system abstraction, in preparation for the work presented in later chapters.

Several key questions that apply to the ESEC model were presented and briefly discussed, along with general consideration of how experimental results are collected, presented and compared.

Chapter 6 describes and specifically investigates the influence of population topology on evolutionary search outcomes. Chapter 7 describes and considers the ESEC model, using the `esec` package, at community and ecosystem levels.

Chapter 6

Population Organisation

6.1 Introduction

The purpose of this chapter is to explore single species population topology models using the ecosystem model for evolutionary computation (ESEC) presented in Chapter 5. By using a range of population topologies, from simple to complex, it is hoped to better understand the relationship and influence topology can have on evolutionary search outcomes.

A representation of the composition of a population organisation, as defined within the ESEC framework, is presented in Figure 6.1. The role of the population organisation is: to define and contain a single species collection of individuals, the interaction topology and model. A population defined in this way is a subsystem component of later community and ecosystem organisation models discussed in more detail in Chapter 7.

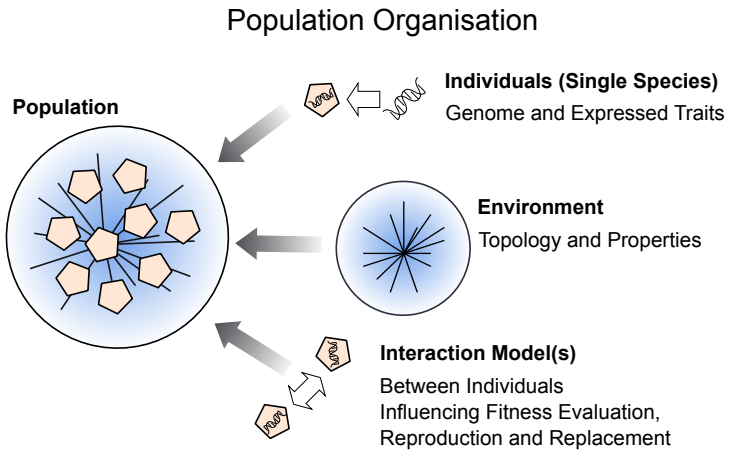


Figure 6.1: Population organisation model within the ESEC framework

In Chapter 3 the sections on representation of a population (Section 3.2.3) and component influence (Section 3.2.10) both considered various population topology models. In particular, those models used in parallel EAs (Section 3.2.10) and structured EAs (Section 3.3.6). Population topology can, in principle, be set to any type of graph: simple

panmictic and cellular lattice models, random, hierarchical trees and complex small-world growth models.

Chapter 5 described in some detail the ecological motivation for EA models to include localised processes, and discussed related work based on ecological principles (Section 5.4). The discussion there included "niche" creation schemes (such as pre-selection, crowding and fitness scaling), different forms of structured topology, and structured population models that consider multiple-interacting species.

A number of population topology related questions are presented in Section 6.2.1, and these are investigated using the ESEC framework and the `esec` software package. The configuration and result of experiments are presented and discussed in Section 6.3.

One of the strongest outcomes of this work is that specific topological properties can and do influence evolution progress and outcomes. Because of this, a better understanding of the influence of topology is clearly desirable.

Although there are some indicators from the results that suggest similar problem domains (with similar levels of complexity) perform better on particular population topology types, the results are limited in scope. Suggestions for further research and open questions are presented in closing.

6.2 Investigation Scope

6.2.1 Objectives

A number of population and ecology related questions can be considered using the ESEC model of population organisation. All of the questions considered here relate to evolutionary search outcomes as a measure of variable influence; some are variations of topology, others are variation of processes in the context of topology.

For structured population topologies, one consideration is the order in which individuals (or "cells") are selected and replaced by new offspring individuals. The "update order" in many cellular EAs is a simple linear sequential order of each cell location. Various other update orders can be conceived, such as a "spiral" pattern. It is also easy to consider a reverse order for any pattern based update sequence.

The following is a list of questions addressed by experiments related to either variations of topology, or variation in process:

- In a broad sense, does population topology type influence evolutionary search? If so:
  - Is the sensitivity of evolutionary search to topology consistent or different among a range of different search domains?
  - Is there a difference in search outcomes between simple lattice, random and complex population topology?
  - How do lattice and complex topologies compare to full graphs?
  - What is the general influence of topology size on outcomes?

  - Is the influence of topology size different on different topologies?

- Are evolutionary search outcomes affected by the topological differences of circular or non-circular lattices, and if so, to what degree?
- Within a structured population model, can update order influence search outcomes? If so:
  - What is the influence of a simple sequential line update order and is this altered by topology?
  - Does the analogy of a spiral update order have an influence, and how does it compare with simpler line sequences?
  - Does fitness based order, forward and reverse, influence search comes?
- Will an ecological model including delayed “juvenile" competition influence search outcomes, and if so is this altered by population topology?
- To what degree does a level of lattice rewiring influence evolutionary search processes, and how does this compare to other influences such as scale and edge density?

To investigate these questions Section 6.2.2 presents a selection of appropriate population topologies, and Section 6.2.3 a selection of real and binary valued search domain landscapes. The experimental work, and the results collected, are presented in Section 6.3.

6.2.2 Selected Population Topology

As an initial base for comparisons, 23 topology types were selected, each of size $n = 100$ vertices (or nearest possible size). Each topology type is indicated by an abbreviated tag name. The topology types selected, and many others, are covered in detail in the Topology Survey of Appendix C. There are several major topology types and a simple notation convention:

- Lattices (“L”): Regular circular two dimensional (2D) lattice models of various degree (“k”). This includes a selection of “hollow” lattices (“h”). For example, “L.hk4” is a hollow lattice of base degree $k = 4$.
- Trees (“T”): A rooted tree growth model where a number of new children c are added each step until the required number of vertices is reached.
- Erdös-Rényi (“ER”): A random graph model $G(n, p)$ where p is the probability of a connection between two vertices in n. For example, the “ER.01” instance is an ER model where $p = 0.01$.
- Watts-Strogatz (“WS”): A small-world model based on a lattice graph with a probability p of random edge rewiring. In this case a one dimensional circular lattice of neighbourhood size three is used, resulting in a vertex degree of $k = 6$. For example, “WS.01” has a rewiring probability of $p = 0.01$.

Later investigation of scale influence include topology sizes of $n = 400$ and $n = 900$. This is based on an equivalent 2D lattice axes scale change from 10×10 to 20×20 and 30×30 respectively. In cases where other topological features need to be preserved (as in the circular honeycomb lattice) the nearest feasible size is used.

Specific investigations into the influence of rewiring use base lattice topologies and apply different degrees of rewiring.

There are three other minor topology types including a "Star" (the natural extreme limit of a tree topology), the Barabási-Albert ("BA") scale-free preferential growth model of power $p = 1$, and a Merge-Regenerate ("MR") instance of mean degree $k = 5$ for regenerated vertices.

Table 6.1 is a summary of the topology labels and definitions used in the base experiment.

Lattice (Circular)	
L.k4	Vertex degree of $k = 4$ (N,S,E,W)
L.k8	Vertex degree of $k = 8$ (k4 + NE,SE,SW,NW)
L.k12	Vertex degree of $k = 12$ (k8+ extended N,S,E,W)
L.hk4	Hollow, base degree of $k = 4$ ($\langle k \rangle \approx 2.96$)
L.hk8	Hollow, base degree of $k = 8$ ($\langle k \rangle \approx 5.92$)
L.k6	3-axes planar lattice, $k = 6$
L.hk3	Honeycomb (hollow 3-axes), $k = 3$
Tree	
T.c2	2 children per parent vertex
T.c3	3 children per parent vertex
...	...
Erdös-Rényi	
ER.01	Vertex pair connection probability of $p = 0.01$
ER.02	Vertex pair connection probability of $p = 0.02$
...	...
Watts-Strogatz	
WS.001	Rewiring probability of $p = 0.001$
WS.01	Rewiring probability of $p = 0.01$
WS.1	Rewiring probability of $p = 0.1$
Individual Topologies	
Star	Simple star topology (or "tree" where $c = n - 1$)
BA.p1	Barabási-Albert growth model (power $p = 1.0$)
MR.5	Merge-Regenerate model (mean degree of $k = 5$)

Table 6.1: Base experiment topology labels and summary details. The complete series of Tree and ER topologies are truncated (indicated by "...") for brevity.

Three different base lattice configurations are shown in Figure 6.2 where each central vertex example (black filled circle) is surrounded by a neighbourhood (blue circles) size of $k = 4$ (Von Neumann), $k = 8$ (Moore) and $k = 12$ (Extended Moore) vertices respectively.

The term "hollow" is used to described a standard base lattice configuration that has had a periodic selection of vertices removed. The result is a lattice that contains several species of vertex degrees. Figure 6.3 shows (non-circular) examples of hollow lattices, based

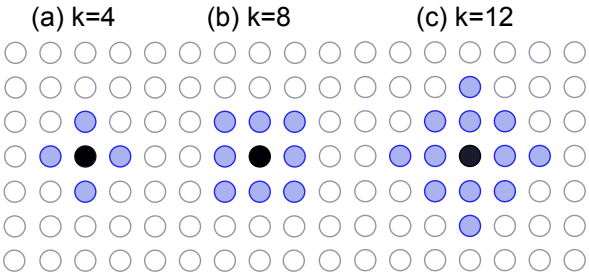


Figure 6.2: Neighbourhoods for (a) $k = 4$, (b) $k = 8$ and (c) $k = 12$ lattices. Also known as the Von Neumann, Moore and Extended Moore neighbourhoods respectively.

on the standard $k = 4$ and $k = 8$ 2D lattices, designated L.hk4 and L.hk8 respectively. In the case of L.hk4, not only are there vertices of degree $k = 4$ but also $k = 2$ create a hybrid of separate and connected topological properties. Similarly for the L.hk8 example, the base vertex degree of $k = 8$ is removed entirely, and replaced by vertex species of degree $k = 6$ and $k = 4$.

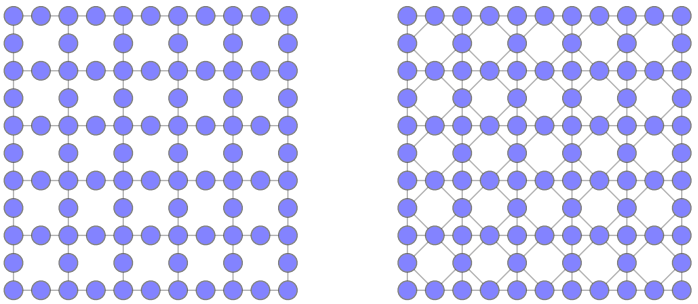


Figure 6.3: 2D layout for L.hk4 and L.hk8 lattices showing hollow features. The L.hk4 instance has a $\langle k \rangle \approx 2.96$ making it the sparest lattice of the group. For L.hk8, $\langle k \rangle \approx 5.92$.

As another variation of standard 2-axes planar lattice configuration, it is possible to use a 3-axes planar layout, both in a base configuration where vertices are of degree $k = 6$ as well as a hollow variation, that results in the organically familiar "honeycomb" shape. Examples are shown in Figure 6.4. The honeycomb lattice is interesting in that all vertices have a uniform degree of $k = 3$, a rather low edge density, which means that measures of "local efficiency" are also low (or zero depending on the measure). Honeycomb lattices also have a rather large girth size (smallest cycle size through unique edges) of six. The honeycomb lattice topology was included in the investigation because it was thought that low density and large girth properties may lend themselves nicely to encouraging spatially unique species development. Result presented later in Section 6.3 suggest that this might be the case for some problem domains.

Tree topologies were selected as a basic hierarchical growth-based model for comparison with other growth-based topology models, such as the BA model. As already noted, the star topology also falls under the scope of tree topology. Note that although a parent-child value c is specified, trees are unbalanced in all the configurations selected here. It would be

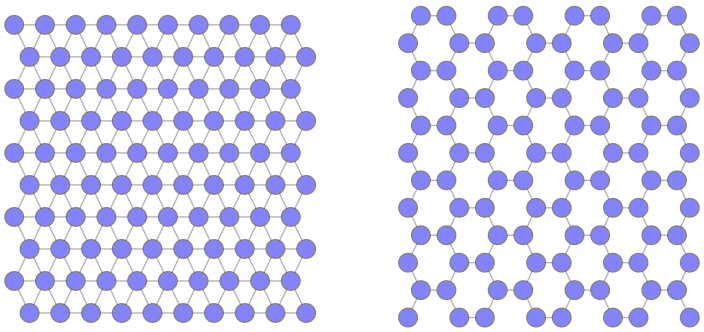


Figure 6.4: 2D 3-axes layout for L.k6 and L.k3 (honeycomb) lattices

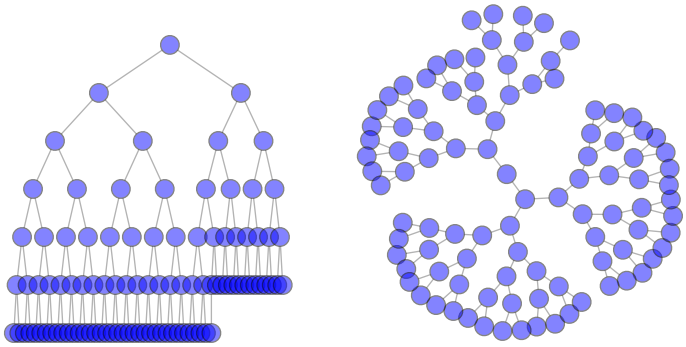


Figure 6.5: Tree and force-based layout for T.c2 graph

reasonable to expect some operation differences between balanced and unbalanced trees, given that unbalanced trees contain a more diverse range of vertex degrees, and degree distribution can influence topology based processes. Figure 6.5 shows an example of the T.c2 topology both as a hierarchical tree layout and a planar force-based layout algorithm.

The small-world topologies created using the Watts-Strogatz model start with a regular lattice, with a specific regular neighbourhood degree, and rewired according to a probability value p. The greater the level of rewiring, the greater the level of random disruption to the regular lattice. As a regular lattice has a relatively high level of local efficiency and large mean path length, the rewiring process can increase the global efficiency (by reducing the mean path length) while still retaining the majority of lattice based local efficiencies. Figure 6.6 show three specific WS topologies created using rewiring probabilities of $p = 0.001$, $p = 0.01$ and $p = 0.1$ respectively. The disruption caused by rewiring can be clearly seen.

Figure 6.7 shows examples of the BA.p1 and MR.5 topologies. Note the hierarchical (tree-like) structure of the BA preferential attachment growth model, and the random and densely clustered form in the MR case that results from the merge-regenerate process (including some characteristically isolated vertices).

An extensive survey of standard measurements and properties based on these topology configurations, and other related variations, is presented in Appendix C. The survey should also provide an indication of appropriate configuration values for interesting topological

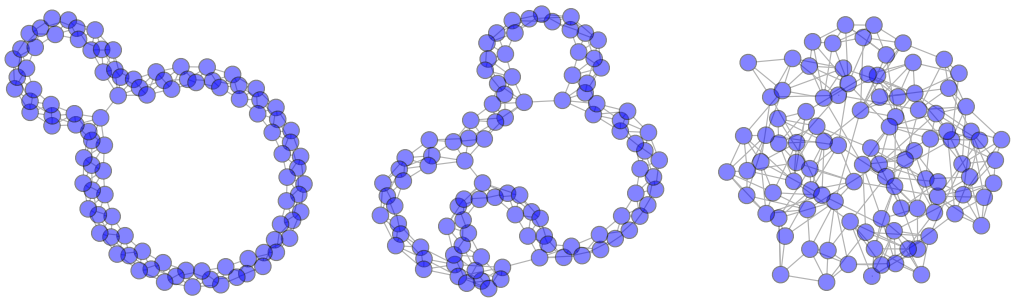


Figure 6.6: Force-based layout for WS.001, WS.01 and WS.1 topology instances. Note the level of disruption caused by the increased levels of rewiring.

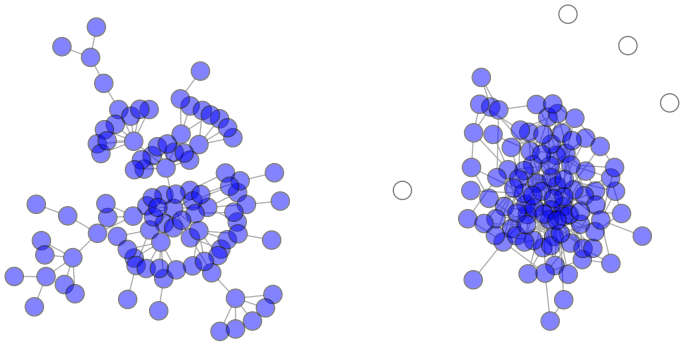


Figure 6.7: Force-based layout of BA.p1 and MR.5 topology examples

properties. For example, there are indications of phase transitions in local efficiency values due to rewiring levels. This could be an interesting direction for future investigations.

6.2.3 Selected Problem Landscapes

A large selection of established and classic test problem domains are available within the `esec` package. For a complete listing see the `esec` package documentation. All of the benchmark problem domains described in Appendix B have been included within the `landscape` module of the current `esec` package.

Both real value and binary value problem landscapes of various qualities have been selected for this series of investigations. Noting that the `esec` package always maximises fitness values, some classic minimisation problems are inverted (typically indicated with an "i") for use. Many real value landscapes have a dimension parameter n which can be increased to create more difficult high-dimensional problem instances.

In the base configuration, binary genomes are used to represent real values. This was done to minimise the influence and number of additional configuration symbols and parameters required when using real value representation. Also, real value genome representation introduces value constraint concerns. The objective of investigations was to better understand the influence of topology, rather than the identification of overall optimal EA configurations for each problem landscape selected. However the exclusive use of binary genomes does limit the generality of results, and presents an open extension

to the work considered here. An investigation was conducted to see if the influence of topology was also observed when using real value genomes. Results are not presented in detailed in this chapter, but are included in the electronic appendices. They confirm that the influence of topology is consistent to that observed using binary genomes.

Real Value Landscapes	
Sphere (Sph)	
Sph.n3i	$n = 3$, inverted, unimodal
Sph.n10i	$n = 20$, inverted, unimodal
Sph.n3	$n = 3$, multimodal, asymmetric bounds
Sph.n10	$n = 20$, multimodal, asymmetric bounds
Rosenbrock's Valley (Ros)	
Ros.n2i	$n = 2$, inverted, unimodal
Ros.n20i	$n = 20$, inverted, irregular modality
Frequency Modulated Sound (FMS)	
FMSi	$n = 6$, inverted

Table 6.2: Labels and summary details for the selected real value landscapes. Note that binary genomes are used to represent real values in the base configuration. The search objective in each case is to find the maximum optimal value. Sph.n3 and Sph.n10 are considered multimodal because in these instances there are multiple deceptive regions in the search space.

Binary Value Landscapes	
Minimum Tardy Task Problem (MTTP)	
MTTP.20i	$n = 20$, inverted
MTTP.100i	$n = 100$, inverted
Whitley's Deceptive 4-bit Problem (WD4B)	
WD4B.5	$n = 5 \times 4 = 20$ (bits), deceptive
WD4B.10	$n = 10 \times 4 = 40$ (bits), deceptive
Subset Sum Problem (SUS)	
SUS.100	$n = 100$, inverted
SUS.1000	$n = 1000$, inverted
SUS.1000e	$n = 1000$, inverted, even values only

Table 6.3: Labels and summary details for the selected binary value landscapes

Table 6.2 is a summary of the real value, and Table 6.3 lists the binary value landscapes selected for the base investigation. Other landscapes have been selected as part of specific investigations presented later.

The base species selected for the experiments is a binary genome. As a consequence the real value landscapes are represented using mapped binary-to-real tables within the specified constraints of each real value landscape range of values, which also excludes unconstrained exploration which can easily occur using real value genome representation.

Real Value Landscapes

- Sphere ("Sph") (Appendix B.3.2). Four instances designated as "Sph.n3i", "Sph.n20i", "Sph.n3", "Sph.n20" are used. The first two are used as easy non-

deceptive unimodal problems, while the second two are inverted with an asymmetrical value range of $(-4.5, 5)$ to create highly deceptive instances.

- Rosenbrock ("Ros") (Appendix B.3.7). Two inverted instances "Ros.n2i" and "Ros.n20i" of this classic non-convex unimodal problem with subtle and deceptive features were used.
- Frequency Modulated Sounds ("FMS") (Appendix B.3.13). A single inverted instance "FMSi" of this highly complex multimodal problem with six parameters, which exhibits strong epistasis, was used.

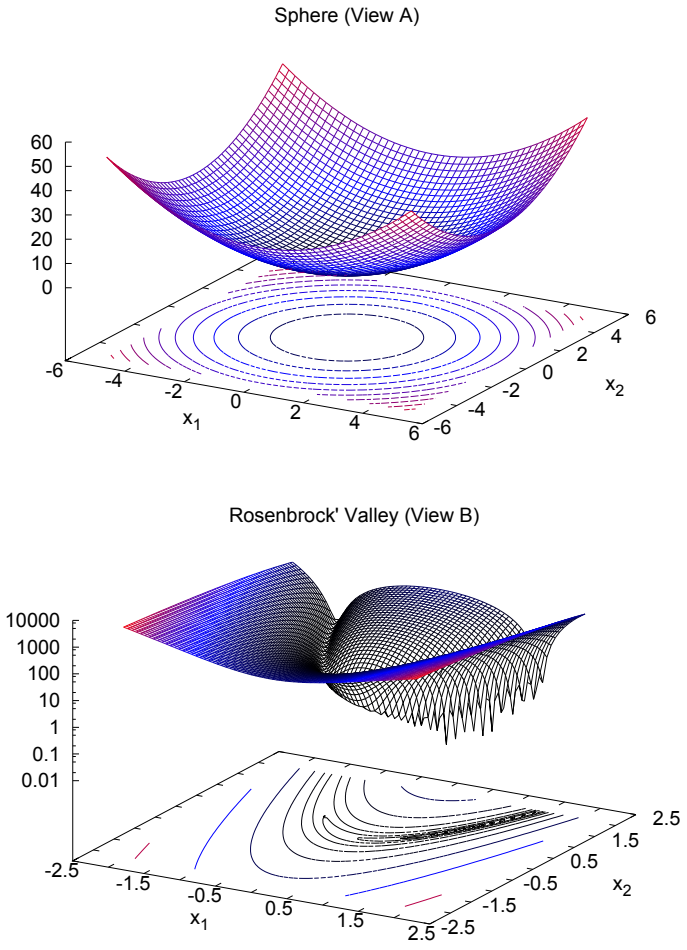


Figure 6.8: The 2D Sphere (De Jong F1) and Rosenbrock real value functions shown in standard minimisation form.

Figure 6.8 shows 3D surface representations of the 2D Sphere and Rosenbrock functions which are classically used as minimisation problems.

Binary Value Landscapes

- Minimum Tardy Tasks Problem ("MTTP") (Appendix B.5.10). This is a single task processor scheduling problem, in which the objective is to find an allocation of

tasks which is both feasible and minimises the total penalty of unallocated "tardy" tasks. Two inverted ("i") instances of this minimisation landscape are used, namely "MTTP.20i" and "MTTP.100i" for 20 and 100 tasks respectively.

- Whitley's Deceptive 4-bit Function ("WD4B") (Appendix B.2.5). Two maximisation instances, "WD4B.5" and "WD4B.10", using 5 and 10 4-bit components each were used. This creates roughly medium and hard difficulty problems respectively, however each 4-bit component is clearly separable. A graph representation of the deceptive 4-bit encoded space is shown in Figure 6.9.
- Subset Sum Problem ("SUS") (Appendix B.5.7). Three inverted minimisation instances of "SUS.100", "SUS.1000" and "SUS.1000e", where "100" and "1000" denotes the size of the subset, were used. (The "i" has been removed for brevity.) The "e" instance contains only even numbers in the set, and is considered easier.

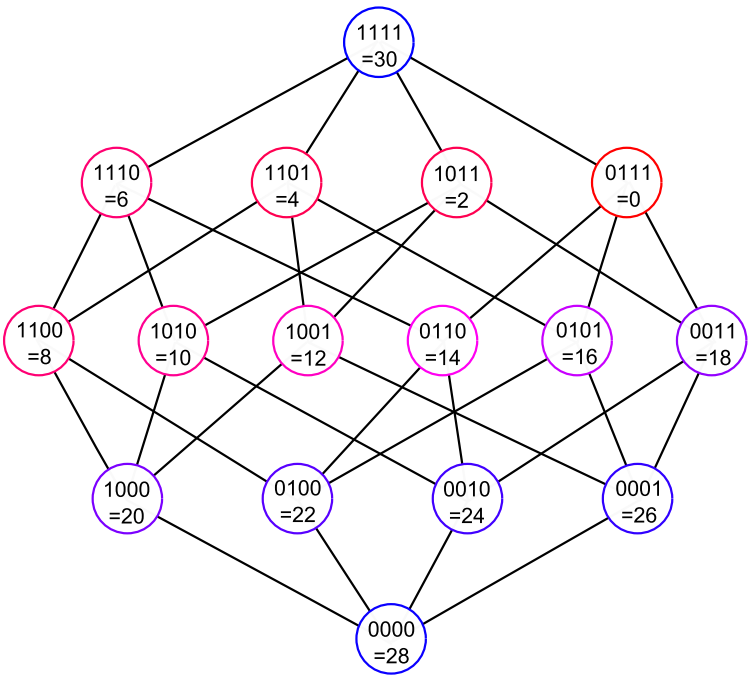


Figure 6.9: Graph representation of the encoding used in Whitley's 4-bit deceptive (WD4B) function. Both the binary string values and the maximisation values have been shown, with colour added to help represent the deceptive nature of the domain.

Ideally, this set of problem landscapes presents a range of simple and hard domains, including small and large dimensionality, unimodal to multimodal, simple to deceptive, and easily separable to highly epistatic. It is with deliberate intent that some of these instances be too difficult for frequent EA success, as the thresholds of success can be used as indicators of influential changes in search performance due to changes in the topology.

6.2.4 Result Comparison Methods

A discussion of methods appropriate for comparing results from evolutionary algorithm search was presented in Section 5.6.

6.3 Experiments

6.3.1 Introduction

The scope of questions related to population organisation and the influence of topology have already been described, along with an initial selection of population topologies and search domain landscapes selected. In order to support a range of investigation variations, a common "base" experiment configuration is used as a reference for later experiments.[1] (See Section 6.3.2 where this is discussed in detail.)

After first considering the influence of different population topology instances on EA search performance (on the range of search domains), a specific line of inquiry regarding topological scale is undertaken. This includes the influence of scale (topology size) for a selection of structured populations as well as full topologies, for comparison.

Regular lattices provide a highly localised neighbourhood for EA processes such as relative fitness assessment and parent, mate, offspring survival and replacement selection. From the review of topology (Chapter 4) and within the survey review of topologies (Appendix C) it can be noted that mean path length profile differs significantly between circular and non-circular (bound) lattices. An investigation compares the influence of circular and bound lattices.

When considering other work on structured EAs with lattice "fine-grained" topology, there have been several suggestions regarding the order that topology locations (cells) are visited. (Update orders are described in more detail in Section 6.3.5.) One general argument is that orderly line based reproduction and replacement may have some advantage over random update order. Update order is also the consideration of selective replacement pressure; the influence of strong elitist pressure may interact with variations of lattice update order. Some novel update orders are also suggested as an interesting extension of existing arguments, including fitness-based order.

Ecological principles suggest that both fitness and competitive replacement are delayed events, and that a species population can contain an internal structure based, for example, on age or interaction type. In biology new offspring rarely compete immediately with adults of the same species, and offspring survival relates to competition among a different set of species than those faced by an adult individual. A simple single species model of juvenile support is considered through delayed replacement; offspring are allocated to the population but do no compete for survival until after a period of delay.

As observed in the properties of natural complex systems, small-world models have been proposed that are based on rewired lattices. Depending on the degree of rewiring applied, and within stochastic variation, such models can contain both efficient local "neighbourhood" characteristics, as well as efficient global performance without the extreme cost of dense or fully connected graph topology. To better understand how the processes of

[1] The base configuration is implemented as an `esec` batch configuration file designated "`batch01a_base`", and later `esec` batch experiments are able to directly import the configuration as a basis for refinement or extension. Similarly the report configuration of the base file can be used to generate reports for other batch experiments.

EA search may be influenced by such rewiring, a number of lattice population topology instances are selected and rewired to various degrees.

Appendix F contains a listing and descriptive overview of the `esec` batch configuration files created and associated with this chapter. Appendix D lists the location of summary report files (and images) as both HTML and PDF documents included on the CDROM. Batch files and selected results are included here as they relate to the investigations and topic groups, and so the order of presentation can different from the order implied by the batch file naming convention.

6.3.2 Topology Influence

Base Investigation

The `batch01a_base esec` batch configuration file specifies a structured EA (sEA) using an ESEC population level of organisation. As presented in Section 5.3.2 the overall configuration contains the "system" architecture, a single "species", search "landscape" and other "application" (investigation) related settings (Listing 6.1). The configuration type is specified as "sEA" and the "system" architecture is of type "population" (or the subtype "structured").

Listing 6.1: Top-level ESEC Structured EA (sEA) configuration

```
cfg = {
    'EA': 'sEA', # Structured EA with specific topology
    'system': {
        ... # population topology, breeding and survival
    },
    'species': { ... }, # genome, recombination and mutation
    'landscape': { ... }, # search domain details
    'application': { ... }, # batch, seed, run stop conditions etc
}
```

System topology is specified (Listing 6.2) with details for one of the set of listed topologies (Table 6.1), and used by the batch in combination with each search landscape instance (Table 6.2 and Table 6.3). The standard default topology size is $n = 100$, or as near as possible within constraints of each topology type. Default topology settings include the update "order" of uniform random sample (URS) without replacement, and undirected (bidirectional) edges. Topology "size" is either stated explicitly (such as $n = 100$) or derived as a consequence of dimensional parameters (as in a 10×10 regular lattice).

Individual solutions for a search landscape (one of the selected real or binary value landscapes) can all be represented by binary genomes. The default species configuration (Listing 6.3) requests that simple uniform random genome initialisation be used, and that evaluation be "binary". The binary evaluation operator is able to map between binary genomes and binary, real or integer value landscapes. For real value landscapes a

Listing 6.2: Base ESEC sEA "system" configuration

```
'system': {
    'type': 'structured', # "Structured Population"
    'topology': {
        'type': None, # To be specified (graph name)
        'order': 'URS', # Default uniform random sample
        'directed': False,
        ... # Other topology settings as needed
    },
    'size': '#topology', # 100 or determined by topology
    'breed': {
        'size': 2, # min. due to crossover
    },
    'replace': {
        'group': 'parent', # The parent
        'selection': None, # Because the group value is enough
        'compete' : True, # True == "only if better"
        'per_gap': 1, # Interval between "replace" update(s)
    },
    'survive': {
        'group': 'offspring', # non-overlapping competition
        'selection': 'best',
        'size': 1,
    },
},
```

resolution parameter specifies the number of genome bits used in order to quantise a finite range of real values.[2]

Listing 6.3: Base ESEC sEA "species" configuration

```
'species': {
    'genome': {
        'type': 'binary',
        'init': 'random', # Simple uniform random bits
        'eval': 'binary' # Map genome <--> landscape
    },
    'recombine': {
        'type': 'one_point',
        'rate': 0.8,
        'parents': 2, # used in breeding (not neighbourhood size)
        'offspring': 2, # due to type
        'parent': {'selection': 'order' }, # set by topology
        'mate': {'selection': 'binary_tournament' } # within neighbours
    },
    'mutate': {
        'type': 'bitflip',
        'rate': 'one',
    },
},
```

Although real value genomes can be selected for real value landscapes, real value genome operators typically introduce a number of additional domain sensitive parameters. As investigations here focus on the influence of topology on EA processes, a simple

[2]This is different from the application resolution value which is used as a "threshold" above which best solution of the EA must achieve

quantised binary genome assists in the objective. Species settings need to be adequate for the search domain, however adjusting species settings, and particularly genome details, is to be avoided when investigating other concerns.

Reproduction in the base configuration is frequent (0.8 = 80%) 1-point (`one_point`) crossover with infrequence ("one"[3]) "bitflip" style mutation, which results in reproduction needing two parent individuals and creating two offspring. The number of offspring created is stored in a group specified by `system.breed.size` which must match the minimum needs of the reproduction operator. For clarity, the primary individual selected to reproduce is termed "parent" and any other individuals used in recombination are termed "mate". In this way the configuration for selection of the parent is specified as "order" (meaning that it depends on the system topology settings) and mates are selected using a stochastic binary tournament. Mate selection is restricted to "neighbours" as defined by the specific topology.

Selection and replacement of new offspring into the population is determined by the system settings for "replace" and "survive". In both cases the "group" value indicates the target of selection; "offspring" selection retains the "best" child, and the target of replacement is the sole "parent" which must compete for survival.

Application settings specify the number of runs (typically 30 or 50) performed for each configuration and the stop criteria for each run. Default settings will stop a run on reaching a generation limit, fixed population (ie. genetically uniform as determined by the species) or on discovery of a good solution (with resolution limits). All runs are seeded with a seed number offset by the run number.[4]

Base Results

Table 6.4 and Table 6.5 present the combined result summaries of all topology types, and all problem instances (real and binary landscapes respectively). Results are presented as average evaluation to success (AES) and the success rate (SR) percentage. It is notable that the Ros.n20i and FMSi problems are difficult for all population topologies. Figure 6.10 is presented as a representative example of the distribution of evaluations to success (ES) for two landscapes.

With respect to AES, the overall best performing topology for real problems is L.hk8, while for binary domains the denser L.k12 performs best for the WD4B instances, and the sparse L.k4 does best for the SUS.1000e domain. For easy real value search domains such as Sph.n3i, Sph.n20i, Sph.n3 (deceptive), and the easy binary value search domains of MTTP.20i and the SUS group, the success rates are almost uniformly 100%.

The inverted sphere landscapes (such as Sph.n3i) have a smooth unimodal maximisation surface, and are certainly not considered difficult problem instances. All of the topology instances were able to achieve a 100% success rate (SR). It is interesting to see in the box and whisker plots that for these two simple problems, Sph.n3i and Sph.n20i

[3] "one" is converted to $= 1/L$ where L is the number of bits in the genome.

[4] Unfortunately the igraph library, used by the `esec` package to create many of the topologies, does not currently support a method for seeding the generation of deterministic graphs.

	Sph.n3i		Sph.n20i		Sph.n3		Sph.n20		Ros.n2i		Ros.n20i		FMSi	
L.k4	2357.3	100%	22893.7	100%	6073.5	100%	42186.0	2%	4544.3	74%	-.-	0%	-.-	0%
L.k8	2099.8	100%	20695.6	100%	5782.0	100%	38397.0	2%	3413.4	80%	-.-	0%	-.-	0%
L.k12	2017.9	100%	19892.6	100%	5611.9	100%	37356.0	4%	3302.8	64%	-.-	0%	-.-	0%
L.hk4	2013.1	100%	19986.1	100%	5125.6	100%	39296.0	2%	4075.8	74%	-.-	0%	-.-	0%
L.hk8	**1817.4**	100%	**17321.0**	100%	**4676.2**	100%	**32743.0**	2%	3497.7	62%	-.-	0%	-.-	0%
L.k6	2207.9	100%	21776.2	100%	5854.8	100%	41229.5	4%	4126.5	78%	-.-	0%	-.-	0%
L.hk3	2267.9	100%	23661.5	100%	6360.1	100%	44204.0	2%	3636.2	74%	-.-	0%	-.-	0%
T.c2	2886.5	100%	31664.9	100%	8056.0	100%	-.-	0%	4558.4	90%	-.-	0%	-.-	0%
T.c3	2908.2	100%	31814.9	100%	7823.5	100%	-.-	0%	4740.5	84%	-.-	0%	-.-	0%
T.c4	2837.0	100%	32452.5	100%	8122.2	100%	-.-	0%	7573.7	86%	-.-	0%	-.-	0%
T.c5	3042.3	100%	33025.9	100%	8285.8	100%	63983.5	4%	6150.6	84%	-.-	0%	-.-	0%
T.c6	2981.9	100%	33643.7	100%	8708.4	100%	64697.0	4%	5909.6	82%	-.-	0%	-.-	0%
ER.01	2658.5	100%	34894.1	100%	8356.9	100%	83281.3	22%	8162.5	100%	-.-	0%	-.-	0%
ER.02	2698.2	100%	28733.9	100%	7461.7	100%	**85141.0**	2%	10618.9	98%	-.-	0%	-.-	0%
ER.03	2444.6	100%	24321.8	100%	6585.0	98%	-.-	0%	7922.9	86%	-.-	0%	-.-	0%
ER.04	2364.8	100%	23415.4	100%	5884.1	100%	42298.0	2%	4134.5	78%	-.-	0%	**24642.0**	2%
ER.05	2120.9	100%	21985.9	100%	5759.3	100%	39036.0	2%	3542.0	74%	-.-	0%	-.-	0%
WS.001	2208.3	100%	24776.7	100%	6468.7	100%	48582.1	16%	4524.3	88%	-.-	0%	-.-	0%
WS.01	2224.2	100%	24290.2	100%	6453.2	100%	46083.0	8%	3675.1	84%	-.-	0%	-.-	0%
WS.1	2194.2	100%	21633.6	100%	5902.2	100%	40558.0	6%	**2809.3**	64%	-.-	0%	-.-	0%
Star	**4241.9**	100%	**44887.4**	100%	**13859.2**	82%	-.-	0%	**21102.0**	60%	-.-	0%	-.-	0%
BA.p1	3014.1	100%	34603.4	100%	8608.4	98%	-.-	0%	7944.8	84%	-.-	0%	-.-	0%
MR.5	2305.1	100%	23037.9	100%	6196.4	100%	42253.0	2%	3826.6	66%	-.-	0%	-.-	0%

Table 6.4: Base experiment results for topologies of size $n = 100$. The Average Evaluations to Success (AES) and Success Rate (SR) % results are shown for each real value landscape. Maximum and minimum values for each AES column group are formatted bold and coloured red. When an SR value is 0%, the AES value is shown as "-.-".

	MTTP.20i		MTTP.100i		WD4B.5		WD4B.10		SUS.100		SUS.1000		SUS.1000e	
L.k4	1597.0	100%	58083.7	18%	4385.9	76%	12051.8	12%	3994.5	100%	4243.6	100%	**3621.5**	100%
L.k8	1363.2	98%	57469.6	14%	4033.4	76%	10829.5	4%	4105.8	100%	4400.7	100%	2468.1	100%
L.k12	1460.1	100%	43468.4	20%	**3197.7**	76%	**5621.0**	2%	3781.3	100%	5118.0	100%	2669.9	100%
L.hk4	1496.7	98%	40520.3	6%	4791.3	78%	12146.0	2%	4055.6	96%	4741.0	100%	2682.4	100%
L.hk8	**1289.1**	98%	**29066.7**	6%	3787.2	68%	-.-	0%	4130.0	98%	5885.9	98%	3178.1	100%
L.k6	1516.2	100%	29696.7	6%	4209.9	84%	9236.0	10%	4964.5	100%	5815.7	100%	3440.0	100%
L.hk3	1613.1	100%	66200.5	16%	4892.0	90%	11623.8	12%	**3616.5**	100%	4439.7	100%	2727.6	100%
T.c2	2125.7	100%	76261.0	4%	10397.7	92%	**35862.9**	14%	4984.7	100%	4001.2	100%	2814.9	100%
T.c3	2034.5	100%	**89831.0**	8%	12835.8	84%	23262.1	14%	4706.6	100%	4953.3	100%	2442.8	100%
T.c4	2196.3	100%	65693.0	4%	16000.0	80%	25534.0	6%	3974.1	100%	4878.8	100%	2583.4	100%
T.c5	2283.4	100%	64191.0	2%	16310.3	76%	29626.3	6%	4063.6	100%	**3817.7**	100%	**2061.9**	100%
T.c6	2039.5	100%	75401.0	8%	15434.6	70%	30294.5	4%	4135.3	100%	3994.5	100%	2506.5	100%
ER.01	2021.6	100%	78143.5	12%	8379.2	90%	25823.6	10%	3776.8	100%	5902.4	100%	2891.8	100%
ER.02	1844.1	100%	72180.5	8%	10160.6	86%	18370.3	6%	**5012.5**	100%	4951.8	100%	2661.8	100%
ER.03	1635.6	100%	70109.3	8%	5294.0	84%	9896.0	6%	3939.8	100%	4819.9	100%	2828.3	100%
ER.04	1655.7	98%	61657.0	20%	4680.2	76%	14678.3	6%	3811.9	100%	5955.6	100%	2866.0	100%
ER.05	1637.8	100%	47182.7	20%	4321.7	80%	-.-	0%	4756.8	100%	6403.1	98%	2972.9	100%
WS.001	1668.2	96%	72542.2	10%	6126.8	90%	23660.1	26%	4661.2	100%	4616.7	100%	2685.4	100%
WS.01	1630.5	100%	47938.6	10%	5785.4	90%	17960.2	26%	3665.7	100%	4968.7	100%	2870.5	100%
WS.1	1621.4	100%	47674.6	20%	4987.1	94%	12056.5	4%	4738.3	98%	5241.4	100%	3083.3	100%
Star	**5156.0**	70%	-.-	0%	**35692.0**	6%	-.-	0%	4074.7	98%	4878.8	100%	2568.4	98%
BA.p1	2333.9	100%	61929.0	4%	22530.3	78%	31410.3	6%	3675.5	100%	4823.3	100%	2459.2	100%
MR.5	1644.6	98%	62343.1	14%	4484.4	64%	-.-	0%	3824.6	100%	**6414.8**	100%	2813.7	100%

Table 6.5: Base experiment results for topologies of size $n = 100$. The Average Evaluations to Success (AES) and Success Rate (SR) % results are shown for each binary value landscape. Maximum and minimum values for each AES column group are formatted bold and coloured red. When an SR value is 0%, the AES value is shown as "-.-".

(Figure 6.10), the regular lattice landscapes clearly have the best AES performance, with the hollow L.hk8 doing best.

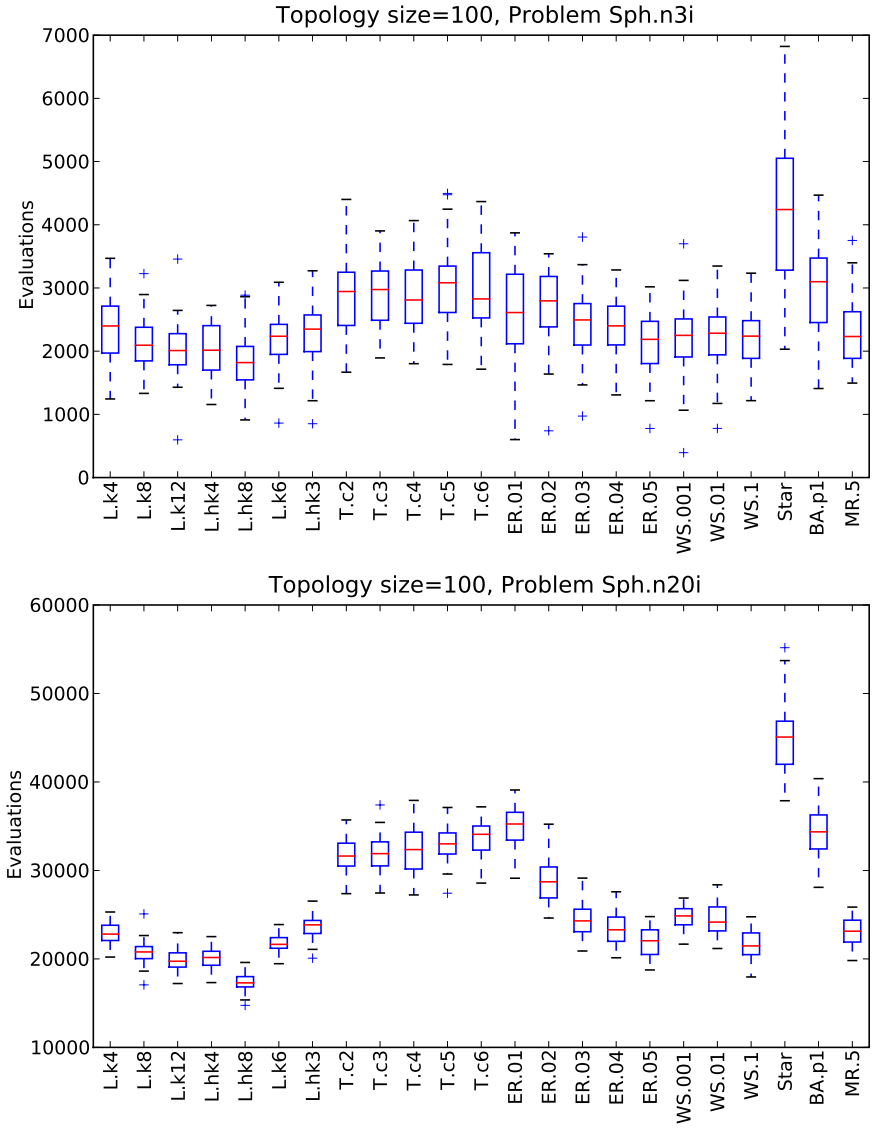


Figure 6.10: Evaluations to Success (ES), as a box and whisker plot, in comparison to topology for the Sph.n3i and Sph.n20i landscapes from the base experiment group of results.

As a topology group, the Tree topologies take longer than either lattices or random models, and the AES usually increases in proportion to the number of tree children. The Star topology is by definition an extreme tree topology, and in line with this initial observation regarding trees, across all problem landscapes the Star topology is consistently a poor performer. Tree and Star topology have a large proportion of sensitive critical paths and nodes. Because of this, valuable traits may have few opportunities to be shared. In some instances the Star is still able to discover adequate solutions, but its success rate (SR) and evaluations to success (ES) distribution seem to indicate that the topology hinders rather than supports the evolutionary search processes.

A distinct lack of success results for the Ros.n20i and FMSi problem landscapes supports their status as difficult problems, with only one successful result for FMSi. An

interesting feature of the success group comparison plot for both these problems (see Figure 6.11) is that some topology types are more likely to become genetically fixed while others reach the arbitrary generation limit. The obvious examples of this are the ER.01 and ER.02 where the high likelihood of isolated subcomponent graphs explicitly prevent genetic convergence. For such topologies it may be worth creating a new "fixed" measure that considers if each subcomponent is fixed instead of a single global measure. Regardless, this type of success result does support expectations based on the known properties of the underlying topology; possible subcomponents influence the EA search process and results.

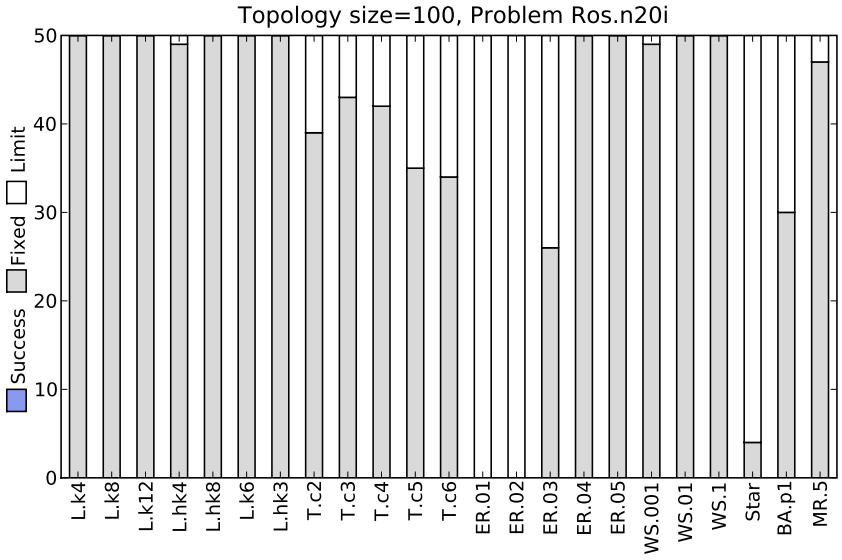


Figure 6.11: Ros.n20i "success", "fixed" and "limit" ratio comparison plot for each topology type from the base experiment results

WD4B.5 and WD4B.10 problem instances are both challenging and a population topology that supports diversity and recombination of the separable deceptive 4-bit blocks would be expected to do well. The lattice, random ER and rewired WS topologies do best with respect to AES and SR results (Figure 6.12). It is interesting to see that the Tree group tend to have a much larger spread of ES values, including quite a few outliers. As a group the WD4B domains provide a niche example, where the complex topology of WS graphs apparently support good SR performance and respectable evaluation distribution.

For the SUS.100, SUS.1000 and SUS.1000e binary problem group the variation in ES distribution between topology groups is small (Figure 6.13). The normally poor performing Tree group actually produces the minimal AES result (T.c5) in both SUS.1000 and SUS.1000e cases. A look at the Mann-Whitney U comparison values (presented as CDROM reports for all experiments) supports there being little distinction between any of the ES distributions among topology instances on these problems. The topology seems to have little or no influence on EA outcomes; the problem domain is not challenging in a way that might benefit from an EA search with processes localised for exploration (specialisation) or globalised for exploitation.

The general result from this initial collection of search landscapes and the range of

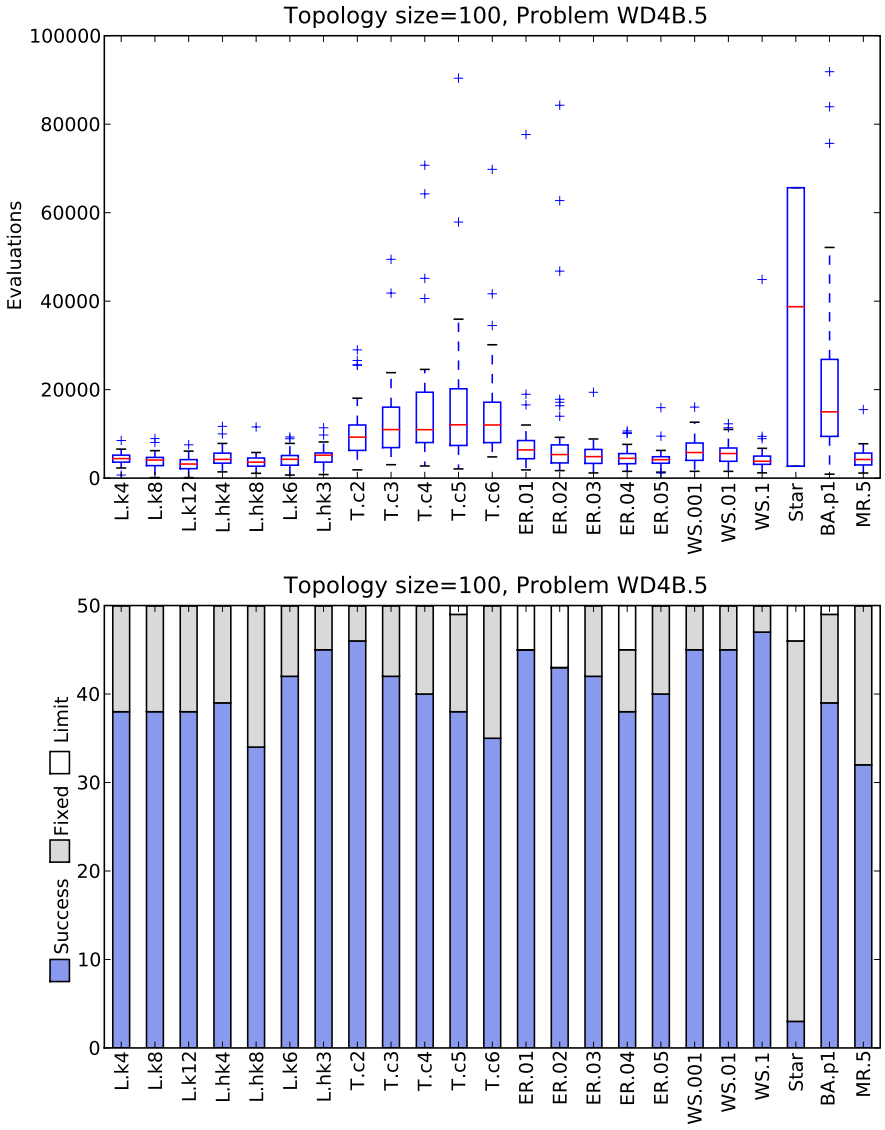


Figure 6.12: ES distribution plot and success ratio comparison plot for each topology type on the WD4B.5 landscape of the base experiment

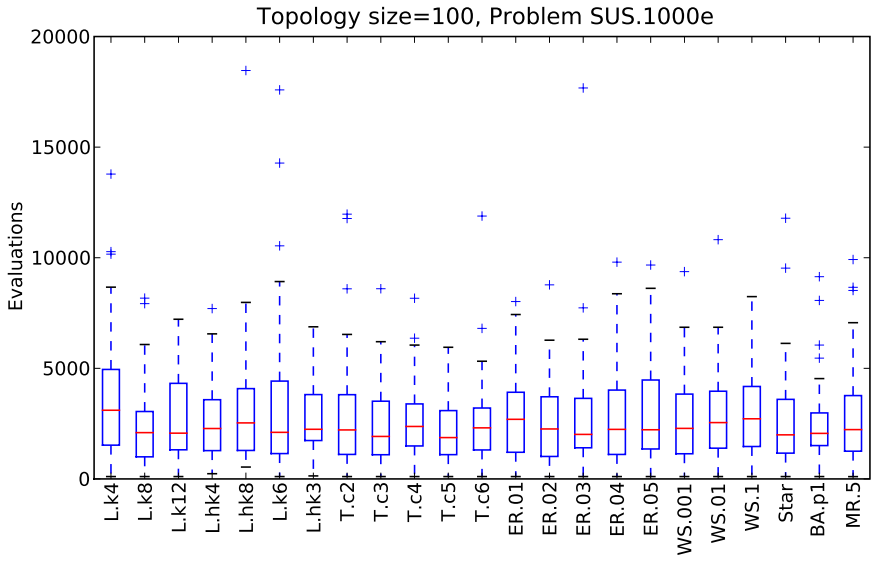


Figure 6.13: ES distribution and topology comparison plot for SUS.1000e from the base experiment results

EA population topologies is that topology does influence both the success rate, and the average number of evaluations to success. For some search domains a reduction in ES or AES results can also result in premature "fixed" population convergence, reducing the overall success ratio. Searches that take a longer time to succeed (influenced by topology) are more robust. For some domains, the distribution characteristics of ES results change with topology groups as a characteristic; some distributions are normal, others skewed, and some have extreme outlier points.

It is also observed that for some simple search domains, such as the SUS and MTTP.20i examples, the influence of topology on the EA search process is reduced to a negligible level. As a consequence there are domains where the overhead of a complex population topology is not useful within an EA configuration.

Full results of the base experiment are included as a HTML and PDF report on the CDROM. A contents listing and guide to the reports are provided in Appendix D.

Additional Problem Results

Although the selection of search landscapes have illustrated interesting topology sensitive results, an additional selection (Table 6.6) of three real value and three binary value landscapes is also considered. The real value landscapes are represented in Figure 6.14. Note that the MSG landscape presented is a random instance of the generator function, and parameters are represented in the landscape features. Additional descriptions of each landscape are included in Appendix B.

The additional real and binary valued landscapes selected are:

- Schwefel's (**Sch**). This is a classic multimodal minimisation problem [310]. It is a deceptive landscape with a single global minimum geometrically distant from the

best local minima within the bound search space. The parameters of the landscape are additively separable. See Appendix B.3.11.

- Max Set of Gaussians (**MSG**). Developed by Gallagher and Yuan [124], this landscape is an irregular and inseparable composition of peaks and distributions. See Appendix B.5.11.
- Massively Multimodal Deceptive Problem (**MMDP**). This deliberately deceptive landscape is similar to the Goldberg 3-bit and Whitney 4-bit (WD4B) deceptive functions. See Appendix B.5.2.
- P-Peak (**PPeaks**) Multimodal Problem Generator. Creates strongly epistatic landscapes, where a solution's fitness is based on the nearest matching peak in a randomly generated set of peaks. See Appendix B.5.3.
- L-SAT Random Satisfiability Problem (**3SAT**). For this landscape, a solution's fitness is the normalised number of true clauses in a Boolean CNF (conjunctive normal form) satisfiability expression. See Appendix B.5.4.

Real Value Landscapes	
Sch.n2i	$n = 2$, inverted, multimodal
Sch.n10i	$n = 10$, inverted, multimodal
MSG.n2	$n = 2$, multimodal ($m = 3$, $p = 1.0$, $r = 0.4$)
Binary Value Landscapes	
MMDP6.20	$n = 120$, $20\times$ 6-bit subcomponents
PPeaks.100	$n = 100$, $P = 100$, inverted, multimodal
3SAT	$n = 100$, $L = 430$ (clauses), $K = 3$ (length)

Table 6.6: Labels and summary details for additional real and binary landscapes

The AES and SR summary results are presented in Table 6.7. Clearly, the difficult MMDP6.20 (deceptive), 3SAT (highly epistatic) and 10 dimensional Sch.10i problem landscapes have not been successfully searched. For the MMDP6.20 and 3SAT cases the generational limit is reached every time (with no "fixed" run results) which indicates that either the search limit is set too low or that the problem domain is too difficult. It may be impractical to set this to a large enough value when other limits such as overall population size may also be restrictive.

Figure 6.15 show three plots, the first of which is the box and whisker evaluations to success (ES) plot for the PPeaks problem. We can see the familiar variation in success evaluation distribution correlated across different population topology which supports previous findings. The success group comparison plot for Sch.n2i also shows similar ratios as those discussed for the earlier Sph.2ni search domain: although the L.hk3 topology is not the minimal example of AES convergence, the high level of SR indicates a robust search process.

Unlike the previous WD4B.5 case where the Tree group of topologies displayed a large degree of variance and a number of outliners, for the Sch.n2i it is the ER topologies with

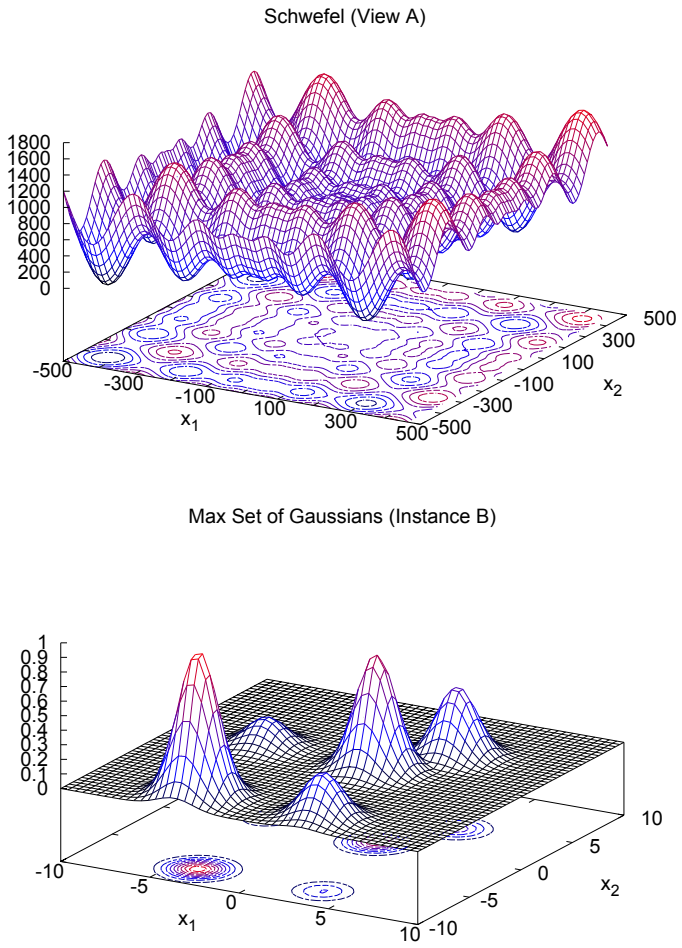


Figure 6.14: The Schwefel landscape and an example landscape generated by the Max Set of Gaussians (MSG) landscape generator

low edge probability (ER.01 and ER.02) that present the most numerous ES distribution outliers.

In summary, half of these additional problem landscapes are supportive of the initial base result findings and the remaining landscapes were not successfully explored. These unsuccessful results show that there are limits within the base settings that may prevent success for some domains. Although the configuration limits could be extended, that is not the focus of this investigation. The additional landscapes are not considered further and are included simply to broaden the range of application results and to suggest limits.

Real Genome Results

The results presented so far have all used binary genome representations, and so it is worth considering if the result of topology sensitivity is particular to binary genomes, or is similar for other genome types.

To investigate this, real genome species were applied to a selection of the real value landscapes considered in the base experiment. A detailed presentation and discussion of the results is not given here, however the simple and concise outcome is that the selection

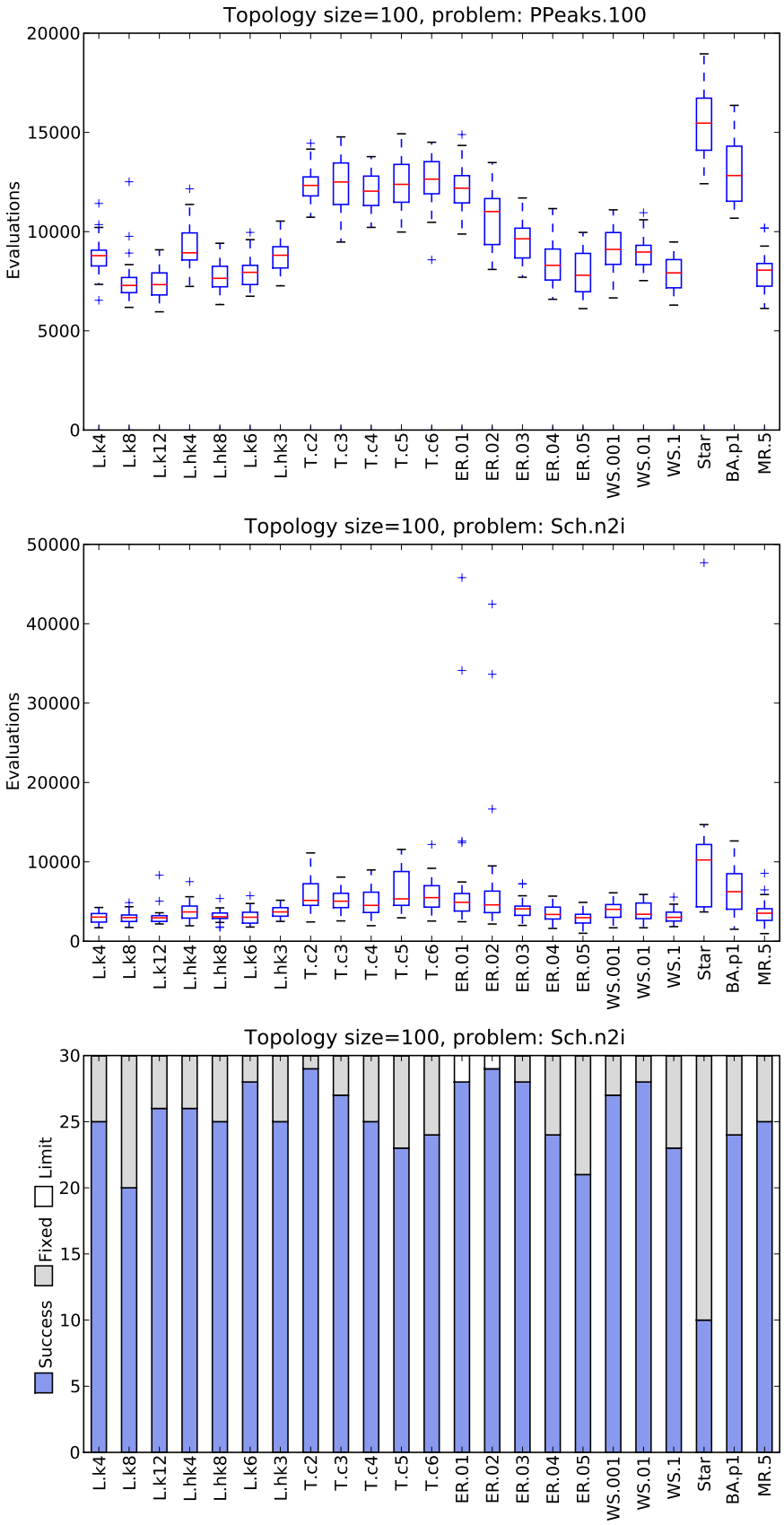


Figure 6.15: Additional results for the PPeaks.100 and Sch.n2i ES landscapes, showing ES distribution plots (for both) and success ratio comparison plot (for Sch.n2i).

	MMDP6.20		PPeaks.100		3SAT		Sch.n2i		Sch.n10i		MSG.n2	
L.k4	-.-	0%	8555.3	100%	-.-	0%	3009.7	83%	-.-	0%	1083.3	100%
L.k8	-.-	0%	7333.4	100%	-.-	0%	2966.3	67%	-.-	0%	1031.5	100%
L.k12	-.-	0%	**7159.7**	100%	-.-	0%	3105.1	87%	-.-	0%	913.7	100%
L.hk4	-.-	0%	9040.8	100%	-.-	0%	3766.6	87%	-.-	0%	1051.3	100%
L.hk8	-.-	0%	7464.8	100%	-.-	0%	3206.0	83%	-.-	0%	**911.1**	100%
L.k6	-.-	0%	7739.3	100%	-.-	0%	3031.2	93%	-.-	0%	1061.5	100%
L.hk3	-.-	0%	8475.5	100%	-.-	0%	3695.8	83%	-.-	0%	1001.6	100%
T.c2	-.-	0%	11963.4	100%	-.-	0%	5834.9	97%	-.-	0%	1350.3	100%
T.c3	-.-	0%	12110.6	100%	-.-	0%	5102.5	90%	-.-	0%	1450.4	100%
T.c4	-.-	0%	11678.5	100%	-.-	0%	5006.7	83%	-.-	0%	1442.9	100%
T.c5	-.-	0%	12095.8	100%	-.-	0%	6389.6	77%	-.-	0%	1467.0	100%
T.c6	-.-	0%	12200.6	100%	-.-	0%	5836.9	80%	-.-	0%	1613.9	100%
ER.01	-.-	0%	11771.5	100%	-.-	0%	7607.5	93%	-.-	0%	1287.6	100%
ER.02	-.-	0%	10332.8	100%	-.-	0%	7538.9	97%	-.-	0%	1238.3	100%
ER.03	-.-	0%	9226.8	100%	-.-	0%	4042.4	93%	-.-	0%	1068.1	100%
ER.04	-.-	0%	8269.5	100%	-.-	0%	3413.8	80%	-.-	0%	994.4	100%
ER.05	-.-	0%	7590.1	100%	-.-	0%	**2908.1**	70%	-.-	0%	1140.9	100%
WS.001	-.-	0%	8882.9	100%	-.-	0%	3836.4	90%	-.-	0%	1197.8	100%
WS.01	-.-	0%	8660.8	100%	-.-	0%	3620.9	93%	-.-	0%	1178.5	100%
WS.1	-.-	0%	7699.7	100%	-.-	0%	3132.5	77%	-.-	0%	1010.2	100%
Star	-.-	0%	**15001.2**	100%	-.-	0%	**11831.3**	33%	-.-	0%	**3129.3**	97%
BA.p1	-.-	0%	12617.2	100%	-.-	0%	6317.5	80%	-.-	0%	1513.7	100%
MR.5	-.-	0%	7696.5	100%	-.-	0%	3733.9	83%	-.-	0%	923.6	100%

Table 6.7: AES and SR results for additional real and binary value landscapes as an extension of the base configuration results

of different topologies does appear to have the same type of influence as seen in binary genome results, including the correlation of stronger influence observed in complex and deceptive domains. The detailed summary report is presented as CDROM documents.

This brief consideration of a different genome representation is not exhaustive and the intent is simply to create a more complete picture of the relationship between evolutionary search processes, search domains and the interaction (if any) of solution representation (genome) with population topology. As discussed in the introduction of the base experiment, binary genomes were specifically selected because they are applicable to both binary and mapped real value landscape domains. Also, only simple operators are needed for binary genomes, while the range and complexity of operators that can be used with real value genomes (in particular for both crossover and mutation) adds additional configuration variables (many sensitive to the search domain) and complexity which is outside of the scope of the investigations presented here.

It would be interesting to investigate if, for a real value domain using effective and specialised real value operators, the influence of topology was increased or reduced, and whether it is more effective to adjust population topology or real value operator selection and parameters as an EA optimisation strategy.

6.3.3 Topology Scale

Size and Influence

The base configuration selects populations with a topology size of, or near to, $n = 100$. Given that the primary reproduction process is specified as recombination (crossover) with only low levels of mutation, the initial random population needs to contain enough genetic diversity as raw material for the EA search process. For most search landscapes $n = 100$ was a suitable and successful size. However, some domains proved difficult or reached limits of genetic fixation (premature convergence) or arbitrary generational limits.

To consider the raw influence that population size has, the base configuration was repeated with scaled up population sizes of $n = 400$ and $n = 900$. The size values are based on consideration of simple 2D lattices, where $10 \times 10 = 100$ in the base configuration, and a linear increase in dimensional size to $20 \times 20 = 400$ and $30 \times 30 = 900$. When the exact scale size is not possible for a topology type (due to structural constraints[5].), the nearest suitable size is selected.

With the additional supply of diverse raw material in larger populations, it is expected that an EA is more likely to be successful. However the additional number of individuals that are processed may result in a larger number of average evaluations to success. It is also more likely that a healthy level of diversity in the population will be preserved for a longer generational time (influenced by the topology) which is supportive of a robust EA search process.

Scale-up Results

Two new batches of results were collected for populations of $n = 400$ and $n = 900$ respectively.[6] The number of runs performed per configuration was decreased from 50 to 30. All other configuration details are the same as for the base experiment.

Summary tables for real and binary value landscape results are presented for the $n = 400$ batch in Table 6.8 and Table 6.9, and for the $n = 900$ batch in Table 6.10 and Table 6.11. These results can be compared to those presented earlier for the $n = 100$ topologies (Table 6.4 and Table 6.5), although a more direct side-by-side comparison is made in the next section.

For the $n = 100$ results the Sph.n20 proved difficult with only a few success results for slightly over half the topology types. In the $n = 400$ results all but the Star topology have some success results, with the lattice and WS small-world models having the greatest SR levels ($70 \sim 100\%$). This trend continued in the $n = 900$ result where the lattice and WS models attained 100% SR.

Similarly the Ros.n2i domain was challenging for the $n = 100$ topologies with most SR results in the $70 \sim 85\%$ range, while for the $n = 400$ results all but three topology

[5] For example, the `L.hk3` "honeycomb" lattice needs to be a particular width and height to avoid introducing vertices of degree other than $k = 3$

[6] The `esec` package allows for the existing base configuration file to be imported directly and the topology sizes re-specified.

	Sph.n3i		Sph.n20i		Sph.n3		Sph.n20		Ros.n2i		Ros.n20i		FMSi	
L.k4	7590.5	100%	80526.1	100%	21754.7	100%	147059.8	100%	5733.5	100%	-.-	0%	117202.0	10%
L.k8	6971.9	100%	71467.6	100%	20168.7	100%	129933.1	87%	6700.9	100%	-.-	0%	-.-	0%
L.k12	6722.2	100%	67147.0	100%	17974.4	100%	124402.6	80%	5676.2	100%	-.-	0%	-.-	0%
L.hk4	6140.9	100%	69862.1	100%	18502.2	100%	128763.4	93%	5870.5	100%	-.-	0%	106301.5	7%
L.hk8	5778.8	100%	59402.3	100%	**16387.1**	100%	**109858.6**	73%	6214.5	97%	-.-	0%	-.-	0%
L.k6	7385.8	100%	76178.7	100%	20751.2	100%	139352.5	100%	5953.9	100%	-.-	0%	87014.0	3%
L.hk3	8358.7	100%	89598.9	100%	23459.3	100%	165485.0	100%	8348.8	100%	-.-	0%	96276.7	10%
T.c2	9031.1	100%	111833.1	100%	26827.9	100%	217492.5	63%	5965.9	100%	-.-	0%	170244.0	3%
T.c3	9296.6	100%	112130.2	100%	27260.3	100%	218641.6	43%	7601.2	100%	-.-	0%	149246.0	3%
T.c4	9478.6	100%	113340.8	100%	27656.3	100%	216085.2	33%	6971.1	100%	-.-	0%	188148.0	3%
T.c5	9771.9	100%	118417.8	100%	27688.1	100%	213224.9	47%	9350.5	100%	-.-	0%	-.-	0%
T.c6	10134.5	100%	117740.2	100%	29064.6	100%	219695.0	10%	5891.2	100%	-.-	0%	-.-	0%
ER.01	7226.2	100%	73965.5	100%	19554.8	100%	138287.9	50%	7100.4	100%	-.-	0%	-.-	0%
ER.02	6853.7	100%	63281.4	100%	18022.3	100%	118768.6	37%	6895.0	100%	-.-	0%	49779.0	3%
ER.03	6183.4	100%	61297.8	100%	17192.2	100%	112820.0	33%	6500.3	100%	-.-	0%	55922.0	3%
ER.04	6428.0	100%	60548.7	100%	17019.7	100%	111446.3	43%	**5441.9**	97%	-.-	0%	-.-	0%
ER.05	**5775.9**	100%	**58433.2**	100%	16589.2	100%	115688.6	27%	6007.4	100%	-.-	0%	**45711.0**	3%
WS.001	7700.5	100%	93888.3	100%	23023.7	100%	176283.8	100%	7427.0	100%	-.-	0%	189144.0	3%
WS.01	7479.7	100%	87072.7	100%	22157.1	100%	160693.5	97%	7169.2	100%	-.-	0%	**232662.0**	3%
WS.1	6544.5	100%	71286.7	100%	19760.7	100%	130139.8	70%	7930.1	100%	-.-	0%	70624.5	7%
Star	**14429.3**	100%	**175701.3**	100%	**49048.1**	80%	-.-	0%	**67892.6**	97%	-.-	0%	-.-	0%
BA.p1	9707.3	100%	125620.8	100%	29214.0	100%	**258793.1**	47%	9435.9	100%	-.-	0%	-.-	0%
MR.5	7233.2	100%	75834.5	100%	20689.3	100%	142839.3	27%	8371.7	100%	-.-	0%	67314.7	10%

Table 6.8: Topology size= 400. Average Evaluations to Success (AES) and Success Rate (SR) results for real value landscapes.

	MTTP.20i		MTTP.100i		WD4B.5		WD4B.10		SUS.100		SUS.1000		SUS.1000e	
L.k4	4741.0	100%	154662.8	83%	12343.9	100%	36526.0	97%	5592.4	100%	8030.5	100%	4799.8	100%
L.k8	4041.6	100%	154359.4	73%	10326.9	100%	29721.7	87%	5471.7	100%	8820.8	100%	4305.9	100%
L.k12	4236.0	100%	110326.2	87%	8941.7	100%	25313.4	90%	5714.2	100%	7721.1	100%	5219.3	100%
L.hk4	4109.8	100%	163073.9	67%	11397.5	100%	35271.3	90%	4104.0	100%	7083.1	100%	3913.9	100%
L.hk8	**3719.0**	100%	125758.1	67%	10390.5	100%	27450.4	87%	**3327.0**	100%	**5813.8**	100%	**3754.6**	100%
L.k6	4516.6	100%	132850.5	77%	11576.6	100%	33400.2	100%	4047.8	100%	7512.9	100%	3900.0	100%
L.hk3	5014.5	100%	198471.9	90%	15276.7	100%	46322.3	100%	5346.6	100%	7672.7	100%	4462.6	100%
T.c2	5278.3	100%	260395.2	43%	26255.2	100%	82087.4	97%	4474.2	100%	6572.7	100%	4101.9	100%
T.c3	5373.5	100%	270253.5	63%	24924.2	100%	88632.6	80%	4376.6	100%	7393.1	100%	4590.1	100%
T.c4	5610.1	100%	270006.8	50%	21461.3	100%	88576.2	80%	3819.5	100%	7068.0	100%	4294.3	100%
T.c5	5889.3	100%	223570.3	40%	28764.6	100%	80800.2	87%	3952.4	100%	6313.4	100%	3987.3	100%
T.c6	5219.0	100%	250404.5	27%	26224.4	100%	80648.5	77%	4376.6	100%	6809.9	100%	4142.1	100%
ER.01	5037.6	100%	145565.1	83%	10795.1	100%	31873.3	90%	5047.4	100%	7465.6	100%	4492.1	100%
ER.02	4192.4	100%	97735.5	77%	9477.3	100%	22604.4	80%	5349.2	100%	7262.8	100%	4947.9	100%
ER.03	4106.9	100%	93955.8	87%	8461.3	100%	**20229.0**	87%	4566.4	100%	7239.9	100%	**5947.9**	100%
ER.04	3856.0	100%	134233.9	83%	8447.3	100%	27418.4	73%	3561.6	100%	8890.0	100%	3907.9	100%
ER.05	4000.5	100%	**89221.9**	80%	**7903.1**	97%	20499.5	80%	**6955.3**	100%	**9692.6**	100%	4904.2	100%
WS.001	4565.5	100%	249691.1	77%	17119.0	100%	79310.5	100%	4571.9	100%	7052.8	100%	4098.2	100%
WS.01	4608.4	100%	219767.5	70%	14768.5	100%	44036.3	97%	4391.8	100%	7379.5	100%	4415.5	100%
WS.1	4254.1	100%	133306.1	80%	9902.2	100%	28300.9	97%	6297.0	100%	7438.6	100%	4825.1	100%
Star	**31170.5**	90%	-.-	0%	**212605.4**	40%	-.-	0%	5135.8	100%	6797.2	100%	4619.8	100%
BA.p1	6211.1	100%	**299827.2**	33%	35376.3	100%	**173443.3**	93%	3735.1	100%	6351.6	100%	3898.0	100%
MR.5	4903.4	100%	138385.3	97%	10343.1	100%	31230.1	70%	4469.2	100%	8019.0	100%	4519.3	100%

Table 6.9: Topology size= 400. Average Evaluations to Success (AES) and Success Rate (SR) results for binary value landscapes.

	Sph.n3i		Sph.n20i		Sph.n3		Sph.n20		Ros.n2i		Ros.n20i		FMSi	
L.k4	14635.0	100%	176404.9	100%	45607.5	100%	313744.1	100%	8570.8	100%	-.-	0%	218907.0	7%
L.k8	12913.6	100%	153781.3	100%	41615.4	100%	279040.2	100%	8581.6	100%	-.-	0%	194789.2	17%
L.k12	12877.5	100%	141779.6	100%	37469.8	100%	257383.7	100%	9171.3	100%	-.-	0%	163843.0	3%
L.hk4	11850.8	100%	150228.0	100%	38740.2	100%	273596.1	100%	8666.4	100%	-.-	0%	278079.6	30%
L.hk8	**10693.3**	100%	129353.9	100%	34534.8	100%	230285.0	100%	**7298.9**	100%	-.-	0%	133832.5	7%
L.k6	14676.2	100%	166300.4	100%	41912.1	100%	294610.0	100%	9180.9	100%	-.-	0%	227335.0	10%
L.hk3	15016.3	100%	193142.3	100%	49011.5	100%	345092.7	100%	9066.6	100%	-.-	0%	314050.7	20%
T.c2	17807.6	100%	238287.0	100%	57915.7	100%	453325.8	97%	8929.4	100%	-.-	0%	322016.0	3%
T.c3	17632.4	100%	240827.3	100%	56917.9	100%	453460.4	90%	10692.3	100%	-.-	0%	281966.8	20%
T.c4	18195.6	100%	240025.6	100%	56382.3	100%	459202.7	87%	12141.3	100%	-.-	0%	323323.5	7%
T.c5	18145.9	100%	247294.2	100%	60183.4	100%	465387.8	83%	11493.0	100%	-.-	0%	300008.0	7%
T.c6	17459.8	100%	247055.1	100%	58189.4	100%	473504.6	53%	10368.7	100%	-.-	0%	185686.0	10%
ER.01	12815.4	100%	126841.2	100%	35396.6	100%	233653.9	97%	9508.6	100%	-.-	0%	87251.5	13%
ER.02	12308.7	100%	118456.4	100%	35205.2	100%	219658.0	87%	8688.5	100%	-.-	0%	92384.4	17%
ER.03	12423.4	100%	117838.2	100%	34178.3	100%	217876.3	83%	9766.8	100%	-.-	0%	124903.0	20%
ER.04	12025.6	100%	116725.0	100%	34566.5	100%	213785.1	90%	10442.6	100%	-.-	0%	102946.0	17%
ER.05	11482.7	100%	**113469.5**	100%	**33241.3**	100%	**211777.0**	87%	10977.4	100%	-.-	0%	**79279.8**	20%
WS.001	14760.5	100%	205445.2	100%	46934.5	100%	401593.1	100%	8877.5	100%	-.-	0%	**338370.3**	20%
WS.01	14759.3	100%	186426.4	100%	44578.2	100%	333356.5	100%	9205.2	100%	-.-	0%	199474.8	17%
WS.1	13622.5	100%	147213.4	100%	41891.1	100%	262282.1	100%	8813.6	100%	-.-	0%	135530.3	23%
Star	**33070.9**	100%	**383730.0**	100%	**147593.1**	100%	-.-	0%	**101284.8**	100%	-.-	0%	-.-	0%
BA.p1	19007.0	100%	271328.5	100%	61009.3	100%	**528297.9**	97%	9785.9	100%	-.-	0%	332389.0	3%
MR.5	14035.9	100%	161160.4	100%	44973.4	100%	300780.8	87%	9736.4	100%	-.-	0%	179029.3	13%

Table 6.10: Topology size= 900. Average Evaluations to Success (AES) and Success Rate (SR) results for real value landscapes.

	MTTP.20i		MTTP.100i		WD4B.5		WD4B.10		SUS.100		SUS.1000		SUS.1000e	
L.k4	9110.0	100%	320483.3	100%	20276.2	100%	69654.5	100%	5958.7	100%	10188.0	100%	5394.4	100%
L.k8	8793.9	100%	222091.0	100%	19854.6	100%	59544.9	100%	5670.0	100%	10114.3	100%	6436.1	100%
L.k12	8398.0	100%	188895.5	100%	18606.0	100%	50367.6	100%	5944.2	100%	11594.6	100%	6187.6	100%
L.hk4	7624.2	100%	341391.6	90%	25056.6	100%	72277.5	100%	4618.9	100%	7384.8	100%	5180.0	100%
L.hk8	**6713.4**	100%	212621.6	93%	18115.0	100%	54381.3	100%	5189.1	100%	8065.2	100%	**4874.6**	100%
L.k6	8470.7	100%	252058.5	100%	19855.9	100%	66919.4	100%	5363.6	100%	9111.0	100%	5630.3	100%
L.hk3	9481.2	100%	335271.8	100%	25714.0	100%	85124.5	100%	5025.5	100%	9045.2	100%	6071.0	100%
T.c2	10207.9	100%	557455.5	87%	40387.7	100%	128625.6	100%	5833.4	100%	8227.6	100%	5981.2	100%
T.c3	11229.4	100%	498489.1	77%	38194.8	100%	129881.6	100%	5004.3	100%	9880.7	100%	5431.4	100%
T.c4	10534.2	100%	497949.1	70%	43153.7	100%	134554.5	100%	5948.5	100%	**6545.9**	100%	5731.8	100%
T.c5	10279.6	100%	541368.4	83%	38119.2	100%	140636.7	100%	5722.9	100%	9316.2	100%	6311.5	100%
T.c6	11313.8	100%	533513.9	73%	43621.4	100%	161760.7	100%	4687.3	100%	8841.4	100%	5943.6	100%
ER.01	7366.8	100%	143626.6	100%	15213.3	100%	40659.0	100%	5153.0	100%	11109.9	100%	5867.1	100%
ER.02	7620.6	100%	131784.1	100%	14289.7	100%	36523.8	100%	6817.4	100%	**11611.4**	100%	5520.0	100%
ER.03	7573.2	100%	**115331.5**	100%	13075.0	100%	36807.2	100%	5244.2	100%	11288.7	100%	7031.2	100%
ER.04	6918.7	100%	116867.2	100%	14041.4	100%	**34513.1**	100%	6639.4	100%	8878.1	100%	6731.5	100%
ER.05	7662.1	100%	125088.8	100%	**13064.8**	100%	35827.4	100%	5311.0	100%	10402.4	100%	**7047.2**	100%
WS.001	8421.9	100%	480686.1	90%	26380.6	100%	119026.9	100%	**4486.7**	100%	9683.3	100%	6928.5	100%
WS.01	8085.5	100%	336106.3	97%	24393.0	100%	84996.7	100%	4583.3	100%	10551.0	100%	6063.9	100%
WS.1	7909.9	100%	225863.9	100%	17320.6	100%	52298.7	100%	5602.3	100%	9932.8	100%	6004.4	100%
Star	**28023.9**	100%	**808441.0**	3%	**391897.3**	87%	-.-	0%	**8473.8**	100%	10713.8	100%	6096.5	100%
BA.p1	11842.5	100%	700886.1	47%	57863.8	100%	**256520.5**	100%	5394.7	100%	8572.4	100%	5048.4	100%
MR.5	9311.8	100%	237833.9	100%	21236.0	100%	58416.0	97%	4752.3	100%	10566.7	100%	5329.9	100%

Table 6.11: Topology size= 900. Average Evaluations to Success (AES) and Success Rate (SR) results for binary value landscapes.

types have 100% SR levels, with the other three at 97% (only one unsuccessful result). At $n = 900$ the SR level is 100% for all topologies.

The FMSi problem domain had only one successful result in the $n = 100$ results, while in the $n = 400$ results 14 topologies have at least one success result, and at $n = 900$ all but the Star have some degree of success. Clearly the FMSi domain requires or benefits from a large population. The variability of AES and SR results support the analysis that this is a highly epistatic domain.

For all three batches no success result was recorded for the Ros.n20i. Although independent trials, not presented, indicated that it is possible to increase the evaluation limit and the population size in order to support success results it is outside the focus of the investigation.[7]

With respect to overall AES performance for real value landscapes, the lattice and ER groups achieve the best results. The result also suggests that the rapid convergence of ER instances is at the expense of SR levels which are, in several cases, below that of the solid lattice performances.

Within the $n = 100$ binary landscape results the MTTP.20i and all three SUS instances had consistently high SR levels ($96 \sim 100\%$), and for the larger topology results the SR levels improved to 100% for all of these domains. For such simple binary landscapes the variation in AES results between topology types appears negligible; there is little to indicate at a scale level of $n = 900$ that the topology type has any influence on search outcome.

If the domain is simple, and/or the initial population variation is large enough, the influence of the topology on search outcomes is reduced.

The MTTP.100i problem had consistently low SR results ($2 \sim 20\%$) for the $n = 100$ topologies. At a scale of $n = 400$ this improved with most SR levels in the $67 \sim 90\%$ range, and at $n = 900$ twelve topology types achieved 100% SR except the Tree topology group which only achieved $73 \sim 87\%$ results. The BA and Star models also struggled on this domain at all scale levels.

The WD4B.5 and WD4B.10 domains are deliberately deceptive. At $n = 100$ all topology types had success results for WD4B.5 but the SR values were below 100%. Within the small variation it appears that at $n = 100$, as a group, the WS models did well on SR level. At $n = 400$ and $n = 900$ the WD4B.5 domain has a 100% SR for almost all topology types.

The WD4B.10 instance is clearly difficult at $n = 100$ with four topologies unable to have any success, and the SR range for others limited to 26%. (Two of the WS models achieved the best SR value.) At $n = 400$ this domain is still challenging with only a few topologies achieving 100% SR (L.k6, L.hk3 and WS.001). The ER models have again achieve a low AES, apparently at the expense of SR which are low among the better performing topology groups.

The Star topology remains a poor performing configuration across both the $n = 400$

[7] Some of the best EA approaches for this domain, such as ES [326] and G3 [81], utilise real value genomes and adaptive step-size variation (mutation) operators that scale (directly or indirectly) and refine the search.

and $n = 900$ results, but with some increased levels of success likely due to the larger initial population sizes and diversity rather than effective search processes. When successful, Star AES levels are almost always the largest or near the largest except for the extremely simple binary domains at large scale where topology has little influence

Scale Comparison Results

In order to present a clearer comparative view, the scale results are combined and tabled with the existing $n = 100$ results. As the total number of topologies, including scale variations, is large they are divided into topology groups:

- *Lattices* (L.k4, L.k8, L.k12, L.hk4, L.hk8, L.k6, L.hk3),
- *Trees* (T.c2, T.c2, T.c3, T.c4, T.c5),
- *ER* (ER.01, ER.02, ER.03, ER.04, ER.05),
- *WS* (WS.001, WS.01 and WS.1), and
- *Others* (Star, BA.p1 and MR.5).

Each topology scale level is denoted with a postscript label of ":a", ":b" and ":c" for the $n = 100$, $n = 400$ and $n = 900$ sizes respectively.

Result summary tables for each topology group are presented in real value and binary value problem groups to make performance comparisons between groups and among problem domains easier. The Ros.n20i results have been excluded as there are no success results, however the fixed and limit results are considered in later sections. Values are not shown for problem domains where the success rate (SR) values are 100% in all or most topology configurations. Minimum and maximum column values are bolded and presented in red.

Table 6.12 and Table 6.13 are the real value problem summaries for the Lattice and Tree topology groups, and Table 6.14 and Table 6.15 present the binary value problem summaries for the same topology groups.

Within the lattice group results (Table 6.12) the hollow and sparse L.hk8:a ($n = 100$) topology had the minimal AES values for all Sph problems, however for the Sph.n20 result the small population size is not as effective with respect to SR as the larger topology sizes. For the binary problems (Table 6.14), the same L.hk8:a topology still had a minimal AES performance, but with significantly poor SR values on non-trivial problems (such as the MTTP and WD4B domains).

The large and sparse L.hk3:c ($n = 900$) had the maximum AES values for the same group of real value Sph problems, as well as the difficult FMSi. For the binary problems, it also took the maximum AES time for MTTP.20i and the WD4B domains. For the simpler SUS domains the maximum AES values presented are similar to those seen in other large and dense lattices.

Maximum and minimum AES values trends are mixed between the different Tree topologies (Table 6.13 and Table 6.15). Certainly for some simple domains small, narrow and deep T.c2 topologies are quick (low AES) and effective (high SR). A large $n = 900$ and moderate tree T.c3 seem to be a good balance for high SR values and reasonable

	Sph.n3i	Sph.n20i	Sph.n3	Sph.n20		Ros.n2i		FMSi	
L.k4:a	2357.3	22893.7	6073.5	42186.0	2%	4544.3	74%	-.-	0%
L.k4:b	7590.5	80526.1	21754.7	147059.8	100%	5733.5	100%	117202.0	10%
L.k4:c	14635.0	176404.9	45607.5	313744.1	100%	8570.8	100%	218907.0	7%
L.k8:a	2099.8	20695.6	5782.0	38397.0	2%	3413.4	80%	-.-	0%
L.k8:b	6971.9	71467.6	20168.7	129933.1	87%	6700.9	100%	-.-	0%
L.k8:c	12913.6	153781.3	41615.4	279040.2	100%	8581.6	100%	194789.2	17%
L.k12:a	2017.9	19892.6	5611.9	37356.0	4%	**3302.8**	64%	-.-	0%
L.k12:b	6722.2	67147.0	17974.4	124402.6	80%	5676.2	100%	-.-	0%
L.k12:c	12877.5	141779.6	37469.8	257383.7	100%	9171.3	100%	163843.0	3%
L.hk4:a	2013.1	19986.1	5125.6	39296.0	2%	4075.8	74%	-.-	0%
L.hk4:b	6140.9	69862.1	18502.2	128763.4	93%	5870.5	100%	106301.5	7%
L.hk4:c	11850.8	150228.0	38740.2	273596.1	100%	8666.4	100%	278079.6	30%
L.hk8:a	**1817.4**	**17321.0**	**4676.2**	**32743.0**	2%	3497.7	62%	-.-	0%
L.hk8:b	5778.8	59402.3	16387.1	109858.6	73%	6214.5	97%	-.-	0%
L.hk8:c	10693.3	129353.9	34534.8	230285.0	100%	7298.9	100%	133832.5	7%
L.k6:a	2207.9	21776.2	5854.8	41229.5	4%	4126.5	78%	-.-	0%
L.k6:b	7385.8	76178.7	20751.2	139352.5	100%	5953.9	100%	**87014.0**	3%
L.k6:c	14676.2	166300.4	41912.1	294610.0	100%	**9180.9**	100%	227335.0	10%
L.hk3:a	2267.9	23661.5	6360.1	44204.0	2%	3636.2	74%	-.-	0%
L.hk3:b	8358.7	89598.9	23459.3	165485.0	100%	8348.8	100%	96276.7	10%
L.hk3:c	**15016.3**	**193142.3**	**49011.5**	**345092.7**	100%	9066.6	100%	**314050.7**	20%

Table 6.12: Lattice group scale comparison of AES and SR results on real value landscapes. SR % values are not shown when all results for a domain are 100% . The Ros.20i values have been excluded since there were no success results. When an SR value is 0%, the AES value is shown as "-.-".

	Sph.n3i	Sph.n20i	Sph.n3	Sph.n20		Ros.n2i		FMSi	
T.c2:a	2886.5	**31664.9**	8056.0	-.-	0%	**4558.4**	90%	-.-	0%
T.c2:b	9031.1	111833.1	26827.9	217492.5	63%	5965.9	100%	170244.0	3%
T.c2:c	17807.6	238287.0	57915.7	453325.8	97%	8929.4	100%	322016.0	3%
T.c3:a	2908.2	31814.9	**7823.5**	-.-	0%	4740.5	84%	-.-	0%
T.c3:b	9296.6	112130.2	27260.3	218641.6	43%	7601.2	100%	**149246.0**	3%
T.c3:c	17632.4	240827.3	56917.9	453460.4	90%	10692.3	100%	281966.8	20%
T.c4:a	**2837.0**	32452.5	8122.2	-.-	0%	7573.7	86%	-.-	0%
T.c4:b	9478.6	113340.8	27656.3	216085.2	33%	6971.1	100%	188148.0	3%
T.c4:c	**18195.6**	240025.6	56382.3	459202.7	87%	**12141.3**	100%	**323323.5**	7%
T.c5:a	3042.3	33025.9	8285.8	**63983.5**	4%	6150.6	84%	-.-	0%
T.c5:b	9771.9	118417.8	27688.1	213224.9	47%	9350.5	100%	-.-	0%
T.c5:c	18145.9	**247294.2**	**60183.4**	465387.8	83%	11493.0	100%	300008.0	7%
T.c6:a	2981.9	33643.7	8708.4	64697.0	4%	5909.6	82%	-.-	0%
T.c6:b	10134.5	117740.2	29064.6	219695.0	10%	5891.2	100%	-.-	0%
T.c6:c	17459.8	247055.1	58189.4	**473504.6**	53%	10368.7	100%	185686.0	10%

Table 6.13: Tree group scale comparison of AES and SR results on real value landscapes. Where all SR values are 100% the values are not shown. The Ros.20i results excluded. When SR is 0% the AES is shown as "-.-".

	MTTP.20i	MTTP.100i		WD4B.5		WD4B.10		SUS.100	SUS.1000	SUS.1000e
L.k4:a	1597.0	58083.7	18%	4385.9	76%	12051.8	12%	3994.5	**4243.6**	3621.5
L.k4:b	4741.0	154662.8	83%	12343.9	100%	36526.0	97%	5592.4	8030.5	4799.8
L.k4:c	9110.0	320483.3	100%	20276.2	100%	69654.5	100%	**5958.7**	10188.0	5394.4
L.k8:a	1363.2*	57469.6	14%	4033.4	76%	10829.5	4%	4105.8	4400.7	**2468.1**
L.k8:b	4041.6	154359.4	73%	10326.9	100%	29721.7	87%	5471.7	8820.8	4305.9
L.k8:c	8793.9	222091.0	100%	19854.6	100%	59544.9	100%	5670.0	10114.3	**6436.1**
L.k12:a	1460.1	43468.4	20%	**3197.7**	76%	**5621.0**	2%	3781.3	5118.0	2669.9
L.k12:b	4236.0	110326.2	87%	8941.7	100%	25313.4	90%	5714.2	7721.1	5219.3
L.k12:c	8398.0	188895.5	100%	18606.0	100%	50367.6	100%	5944.2	**11594.6**	6187.6
L.hk4:a	1496.7*	40520.3	6%	4791.3	78%	12146.0	2%	4055.6*	4741.0	2682.4
L.hk4:b	4109.8	163073.9	67%	11397.5	100%	35271.3	90%	4104.0	7083.1	3913.9
L.hk4:c	7624.2	**341391.6**	90%	25056.6	100%	72277.5	100%	4618.9	7384.8	5180.0
L.hk8:a	**1289.1***	**29066.7**	6%	3787.2	68%	-.-	0%	4130.0*	5885.9*	3178.1
L.hk8:b	3719.0	125758.1	67%	10390.5	100%	27450.4	87%	**3327.0**	5813.8	3754.6
L.hk8:c	6713.4	212621.6	93%	18115.0	100%	54381.3	100%	5189.1	8065.2	4874.6
L.k6:a	1516.2	29696.7	6%	4209.9	84%	9236.0	10%	4964.5	5815.7	3440.0
L.k6:b	4516.6	132850.5	77%	11576.6	100%	33400.2	100%	4047.8	7512.9	3900.0
L.k6:c	8470.7	252058.5	100%	19855.9	100%	66919.4	100%	5363.6	9111.0	5630.3
L.hk3:a	1613.1	66200.5	16%	4892.0	90%	11623.8	12%	3616.5	4439.7	2727.6
L.hk3:b	5014.5	198471.9	90%	15276.7	100%	46322.3	100%	5346.6	7672.7	4462.6
L.hk3:c	**9481.2**	335271.8	100%	**25714.0**	100%	**85124.5**	100%	5025.5	9045.2	6071.0

Table 6.14: Lattice group scale comparison of AES and SR results on binary value landscapes. Where all, or almost all, SR values are 100% the values are not shown. AES values marked with "*" indicate that the SR value was only one or two success values below 100%. When SR is 0% the AES is shown as "-.-".

	MTTP.20i	MTTP.100i		WD4B.5		WD4B.10		SUS.100	SUS.1000	SUS.1000e
T.c2:a	2125.7	76261.0	4%	**10397.7**	92%	35862.9	14%	4984.7	4001.2	2814.9
T.c2:b	5278.3	260395.2	43%	26255.2	100%	82087.4	97%	4474.2	6572.7	4101.9
T.c2:c	10207.9	**557455.5**	87%	40387.7	100%	128625.6	100%	5833.4	8227.6	5981.2
T.c3:a	**2034.5**	89831.0	8%	12835.8	84%	**23262.1**	14%	4706.6	4953.3	2442.8
T.c3:b	5373.5	270253.5	63%	24924.2	100%	88632.6	80%	4376.6	7393.1	4590.1
T.c3:c	11229.4	498489.1	77%	38194.8	100%	129881.6	100%	5004.3	**9880.7**	5431.4
T.c4:a	2196.3	65693.0	4%	16000.0	80%	25534.0	6%	3974.1	4878.8	2583.4
T.c4:b	5610.1	270006.8	50%	21461.3	100%	88576.2	80%	**3819.5**	7068.0	4294.3
T.c4:c	10534.2	497949.1	70%	43153.7	100%	134554.5	100%	**5948.5**	6545.9	5731.8
T.c5:a	2283.4	**64191.0**	2%	16310.3	76%	29626.3	6%	4063.6	**3817.7**	**2061.9**
T.c5:b	5889.3	223570.3	40%	28764.6	100%	80800.2	87%	3952.4	6313.4	3987.3
T.c5:c	10279.6	541368.4	83%	38119.2	100%	140636.7	100%	5722.9	9316.2	**6311.5**
T.c6:a	2039.5	75401.0	8%	15434.6	70%	30294.5	4%	4135.3	3994.5	2506.5
T.c6:b	5219.0	250404.5	27%	26224.4	100%	80648.5	77%	4376.6	6809.9	4142.1
T.c6:c	**11313.8**	533513.9	73%	**43621.4**	100%	**161760.7**	100%	4687.3	8841.4	5943.6

Table 6.15: Tree group scale comparison of AES and SR results on binary value landscapes. Where all SR values are 100% the values are not shown.

AES time. The shallow and broad T.c6 topology do, in general, take the longest time to converge but with good success rate level at larger topology sizes for difficult or deceptive problems (such as the MTTP.100i and WD4B domains).

For the ER topology group best (minimum) AES results for real value domains (Table 6.16) were among the small and dense ER.05:a ($n = 100$) instances, while for binary domains (Table 6.19) the minimal AES result varied, but still showed preference for small and dense ER instances (ER.03:a and ER.05:a). The success rate improved as the graph scale, n, increased.

Across both the real and binary value landscapes (Table 6.17 and Table 6.20) the WS small-world topologies consistently showed minimal AES performance for the the small graphs ("a:" $n = 100$), and mostly for the larger rewiring probability of $p = 0.1$ (WS.1:a) with the exceptions being among the simpler binary domains or the difficult real valued FMSi domain (where the WS.1:a instance was not successful). Similarly, the largest AES values were among the large $n = 900$ and relatively untouched lattice of the WS.001:c. Success rate for difficult domains again improved with scale increase (as seen in the Sph.n20, Ros.n2i, FMSi, MTTP.100i and WD4B domains).

The collection of "Other" topologies makes group level comparison less clear (Table 6.18 and Table 6.21), but between the Star, BA and MR models, the ER based MR.4:a consistently performed well with minimal AES values across real and binary problem domains, but with the familiar poor SR value on difficult domains. The exceptions being among the simple SUS binary domains where all topology groups have been shown to have less influence among other stochastic factors. As noted several times, the Star topology appears to specifically hinder evolutionary progress taking the majority of largest AES and poorest SR classifications.

Figure 6.16 shows a comparison of lattice topologies applied to the Sph.n3 and WD4B.5 domains, both of which are of sufficient difficulty to show the influence of topology configuration and size. In the Sph.n3 results not only is there a clear scaled increase related to topology scale, but also a wide separation between distributions. Similarly, within the WD4B.5 domain larger topologies take more time to reach success. However in the WD4B.5 case the results also tend to have a larger and overlapping ES distribution spread for larger populations. This is something that is suggested in other results, but not as clearly evident as within the Sph.n3 results.

The MTTP.100i domain is a complex domain that presents a reasonable challenge for all topology groups and across scale, with the L.hk8 lattice and the ER.03 and ER.05 providing successful and efficient at different scales. Figure 6.17 demonstrates two examples, from the lattice and the ER group of results, that topology size increases the ratio or profile of the SR, fixed and limit results changes. Overall, the ratio of success results increases with scale in all cases. In the lattice group, the ratio of "fixed" results also tends to be reduced as scale increases. Within the ER.01 and ER.02 instances of small $n = 100$ size there are no "fixed" results. As discussed before, this is almost certainly due to the occurrence of isolated subgraph components that prevent population wide convergence. The same success rate ratios can be observed with the MTTP.100i domain with other

	Sph.n3i	Sph.n20i	Sph.n3	Sph.n20		Ros.n2i		FMSi	
ER.01:a	2658.5	34894.1	8356.9	83281.3	22%	8162.5	100%	-.-	0%
ER.01:b	7226.2	73965.5	19554.8	138287.9	50%	7100.4	100%	-.-	0%
ER.01:c	**12815.4**	**126841.2**	**35396.6**	**233653.9**	97%	9508.6	100%	87251.5	13%
ER.02:a	2698.2	28733.9	7461.7	85141.0	2%	10618.9	98%	-.-	0%
ER.02:b	6853.7	63281.4	18022.3	118768.6	37%	6895.0	100%	49779.0	3%
ER.02:c	12308.7	118456.4	35205.2	219658.0	87%	8688.5	100%	92384.4	17%
ER.03:a	2444.6	24321.8	6585.0*	-.-	0%	7922.9	86%	-.-	0%
ER.03:b	6183.4	61297.8	17192.2	112820.0	33%	6500.3	100%	55922.0	3%
ER.03:c	12423.4	117838.2	34178.3	217876.3	83%	9766.8	100%	**124903.0**	20%
ER.04:a	2364.8	23415.4	5884.1	42298.0	2%	4134.5	78%	**24642.0**	2%
ER.04:b	6428.0	60548.7	17019.7	111446.3	43%	5441.9	97%	-.-	0%
ER.04:c	12025.6	116725.0	34566.5	213785.1	90%	10442.6	100%	102946.0	17%
ER.05:a	**2120.9**	**21985.9**	**5759.3**	**39036.0**	2%	**3542.0**	74%	-.-	0%
ER.05:b	5775.9	58433.2	16589.2	115688.6	27%	6007.4	100%	45711.0	3%
ER.05:c	11482.7	113469.5	33241.3	211777.0	87%	**10977.4**	100%	79279.8	20%

Table 6.16: ER group scale comparison of AES and SR results on real value landscapes. Where all, or almost all, SR values are 100% the values are not shown. The AES value marked with "*" indicates that the SR value was 98%. Ros.20i results excluded. When SR is 0% the AES is shown as "-.-".

	Sph.n3i	Sph.n20i	Sph.n3	Sph.n20		Ros.n2i		FMSi	
WS.001:a	2208.3	24776.7	6468.7	48582.1	16%	4524.3	88%	-.-	0%
WS.001:b	7700.5	93888.3	23023.7	176283.8	100%	7427.0	100%	189144.0	3%
WS.001:c	**14760.5**	**205445.2**	**46934.5**	**401593.1**	100%	8877.5	100%	**338370.3**	20%
WS.01:a	2224.2	24290.2	6453.2	46083.0	8%	3675.1	84%	-.-	0%
WS.01:b	7479.7	87072.7	22157.1	160693.5	97%	7169.2	100%	232662.0	3%
WS.01:c	14759.3	186426.4	44578.2	333356.5	100%	**9205.2**	100%	199474.8	17%
WS.1:a	**2194.2**	**21633.6**	**5902.2**	**40558.0**	6%	**2809.3**	64%	-.-	0%
WS.1:b	6544.5	71286.7	19760.7	130139.8	70%	7930.1	100%	**70624.5**	7%
WS.1:c	13622.5	147213.4	41891.1	262282.1	100%	8813.6	100%	135530.3	23%

Table 6.17: WS group scale comparison of AES and SR results on real value landscapes. Where all SR values are 100% the values are not shown. Ros.20i results excluded. When SR is 0% the AES is shown as "-.-".

	Sph.n3i	Sph.n20i	Sph.n3		Sph.n20		Ros.n2i		FMSi	
Star:a	4241.9	44887.4	13859.2	82%	-.-	0%	21102.0	60%	-.-	0%
Star:b	14429.3	175701.3	49048.1	80%	-.-	0%	67892.6	97%	-.-	0%
Star:c	**33070.9**	**383730.0**	**147593.1**	100%	-.-	0%	**101284.8**	100%	-.-	0%
BA.p1:a	3014.1	34603.4	8608.4	98%	-.-	0%	7944.8	84%	-.-	0%
BA.p1:b	9707.3	125620.8	29214.0	100%	258793.1	47%	9435.9	100%	-.-	0%
BA.p1:c	19007.0	271328.5	61009.3	100%	**528297.9**	97%	9785.9	100%	**332389.0**	3%
MR.5:a	**2305.1**	**23037.9**	**6196.4**	100%	**42253.0**	2%	**3826.6**	66%	-.-	0%
MR.5:b	7233.2	75834.5	20689.3	100%	142839.3	27%	8371.7	100%	**67314.7**	10%
MR.5:c	14035.9	161160.4	44973.4	100%	300780.8	87%	9736.4	100%	179029.3	13%

Table 6.18: Other group scale comparison of AES and SR results on real value landscapes. Where all SR values are 100% the values are not shown. Ros.20i results excluded. When SR is 0% the AES is shown as "-.-".

	MTTP.20i	MTTP.100i		WD4B.5		WD4B.10		SUS.100	SUS.1000	SUS.1000e
ER.01:a	2021.6	78143.5	12%	8379.2	90%	25823.6	10%	3776.8	5902.4	2891.8
ER.01:b	5037.6	**145565.1**	83%	10795.1	100%	31873.3	90%	5047.4	7465.6	4492.1
ER.01:c	7366.8	143626.6	100%	**15213.3**	100%	**40659.0**	100%	5153.0	11109.9	5867.1
ER.02:a	1844.1	72180.5	8%	10160.6	86%	18370.3	6%	5012.5	4951.8	**2661.8**
ER.02:b	4192.4	97735.5	77%	9477.3	100%	22604.4	80%	5349.2	7262.8	4947.9
ER.02:c	7620.6	131784.1	100%	14289.7	100%	36523.8	100%	6817.4	**11611.4**	5520.0
ER.03:a	**1635.6**	70109.3	8%	5294.0	84%	**9896.0**	6%	3939.8	**4819.9**	2828.3
ER.03:b	4106.9	93955.8	87%	8461.3	100%	20229.0	87%	4566.4	7239.9	5947.9
ER.03:c	7573.2	115331.5	100%	13075.0	100%	36807.2	100%	5244.2	11288.7	7031.2
ER.04:a	1655.7*	61657.0	20%	4680.2	76%	14678.3	6%	3811.9	5955.6	2866.0
ER.04:b	3856.0	134233.9	83%	8447.3	100%	27418.4	73%	**3561.6**	8890.0	3907.9
ER.04:c	6918.7	116867.2	100%	14041.4	100%	34513.1	100%	6639.4	8878.1	6731.5
ER.05:a	1637.8	**47182.7**	20%	**4321.7**	80%	-.-	0%	4756.8	6403.1*	2972.9
ER.05:b	4000.5	89221.9	80%	7903.1	97%	20499.5	80%	**6955.3**	9692.6	4904.2
ER.05:c	**7662.1**	125088.8	100%	13064.8	100%	35827.4	100%	5311.0	10402.4	**7047.2**

Table 6.19: ER group scale comparison of AES and SR results on binary value landscapes. Where all, or almost all, SR values are 100% the values are not shown. The AES values marked with "*" indicate that the SR value was 98%.

	MTTP.20i	MTTP.100i		WD4B.5		WD4B.10		SUS.100	SUS.1000	SUS.1000e
WS.001:a	1668.2*	72542.2	10%	6126.8	90%	23660.1	26%	4661.2	**4616.7**	**2685.4**
WS.001:b	4565.5	249691.1	77%	17119.0	100%	79310.5	100%	4571.9	7052.8	4098.2
WS.001:c	**8421.9**	**480686.1**	90%	**26380.6**	100%	**119026.9**	100%	4486.7	9683.3	**6928.5**
WS.01:a	1630.5	47938.6	10%	5785.4	90%	17960.2	26%	**3665.7**	4968.7	2870.5
WS.01:b	4608.4	219767.5	70%	14768.5	100%	44036.3	97%	4391.8	7379.5	4415.5
WS.01:c	8085.5	336106.3	97%	24393.0	100%	84996.7	100%	4583.3	**10551.0**	6063.9
WS.1:a	**1621.4**	**47674.6**	20%	**4987.1**	94%	**12056.5**	4%	4738.3*	5241.4	3083.3
WS.1:b	4254.1	133306.1	80%	9902.2	100%	28300.9	97%	**6297.0**	7438.6	4825.1
WS.1:c	7909.9	225863.9	100%	17320.6	100%	52298.7	100%	5602.3	9932.8	6004.4

Table 6.20: WS group scale comparison of AES and SR results on binary value landscapes. Where all, or almost all, SR values are 100% the values are not shown. AES values marked with "*" indicate that the SR value was only one or two success values below 100%.

	MTTP.20i		MTTP.100i		WD4B.5		WD4B.10		SUS.100	SUS.1000
Star:a	5156.0	70%	-.-	0%	35692.0	6%	-.-	0%	4074.7*	4878.8
Star:b	**31170.5**	90%	-.-	0%	212605.4	40%	-.-	0%	5135.8	6797.2
Star:c	28023.9	100%	**808441.0**	3%	**391897.3**	87%	-.-	0%	**8473.8**	**10713.8**
BA.p1:a	2333.9	100%	**61929.0**	4%	22530.3	78%	31410.3	6%	**3675.5**	**4823.3**
BA.p1:b	6211.1	100%	299827.2	33%	35376.3	100%	173443.3	93%	3735.1	6351.6
BA.p1:c	11842.5	100%	700886.1	47%	57863.8	100%	**256520.5**	100%	5394.7	8572.4
MR.5:a	**1644.6**	98%	62343.1	14%	**4484.4**	64%	-.-	0%	3824.6	6414.8
MR.5:b	4903.4	100%	138385.3	97%	10343.1	100%	**31230.1**	70%	4469.2	8019.0
MR.5:c	9311.8	100%	237833.9	100%	21236.0	100%	58416.0	97%	4752.3	10566.7

Table 6.21: Other group scale comparison of AES and SR results on binary value landscapes. Where all, or almost all, SR values are 100% the values are not shown. The AES value marked with "*" indicates that the SR value is 98%. SUS.1000e results excluded. When SR is 0% the AES is shown as "-.-".

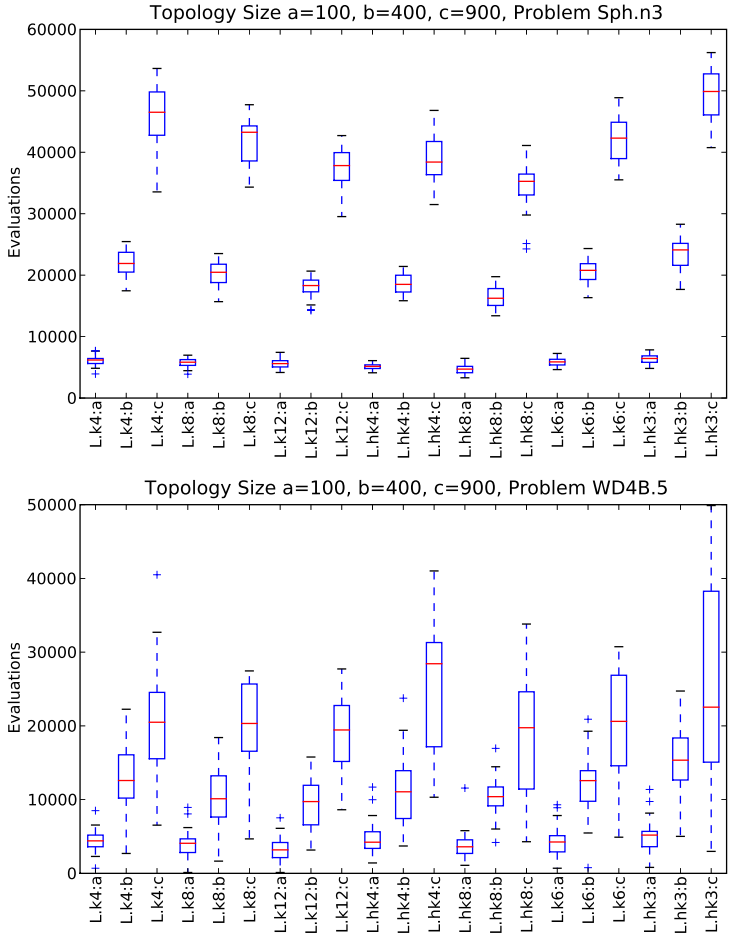


Figure 6.16: Lattice group comparison of ES distributions on Sph.n3 and WD4B.5 domains. Note the wide scale-based ES distribution separation for Sph.n3, and the wider and overlapping ES distributions of WD4B.5.

topology groups such as the Trees (not shown here, but included in the complete CDROM appendices of result summary reports).

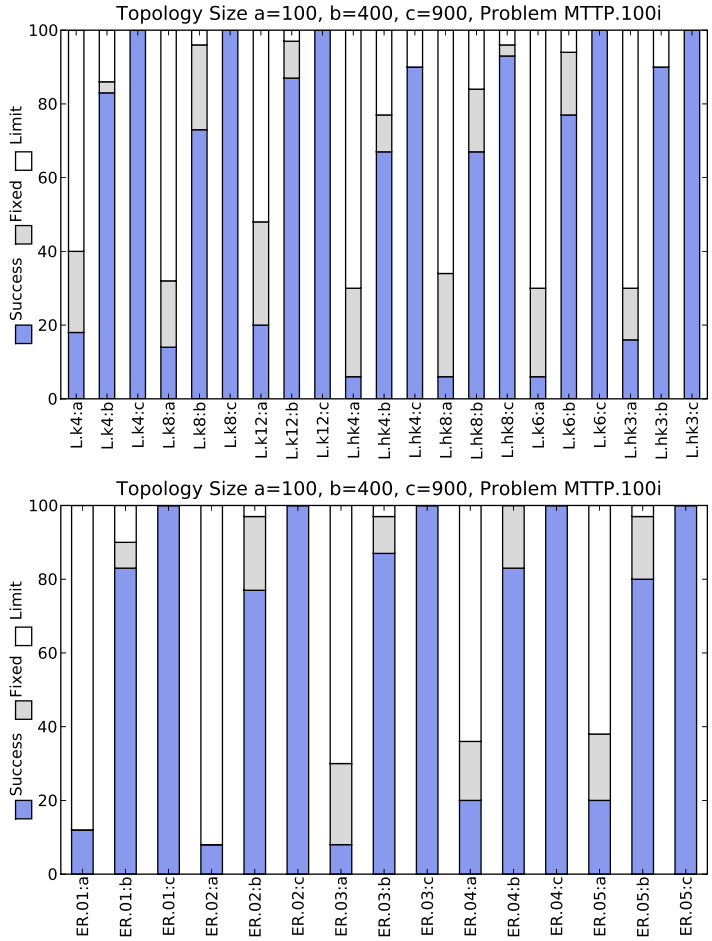


Figure 6.17: Lattice and ER group comparison of SR ratio on the MTTP.100i domain. Note that the ratio of fixed results tends to decrease with increased topology scale.

The small-world WS topology group presents results consistent with ES distribution and SR ratio results, and also indicate the influence of rewiring degree within the lattice based model. In Figure 6.18 the simple SUS.100 domain shows only a small distribution response to scale and topology changes. Figure 6.19 shows how the WS models are strongly influenced by overall topology scale when applied to the 20 dimension Sph.n20i domain (which is not deceptive but does exaggerate features noted on simpler and lower dimensional domains). This is consistent with the observations of Figure 6.16 earlier. An additional noticeable feature is that the position of the median and distribution range decreases with increased levels of rewiring. The higher the level of rewiring, the lower the mean path length. The ES distribution results support the idea that a population with lower mean path length characteristics converges quicker than isolated and sparse graphs.

Figure 6.20 shows the matching ES distribution and SR ratio results for the WS topology group applied to the MTTP.100i domain (which has already been selected for discussion and figure presentation several times because of its interesting SR characteristics). Note how in this matching set of plots the ES distributions spread and median values in-

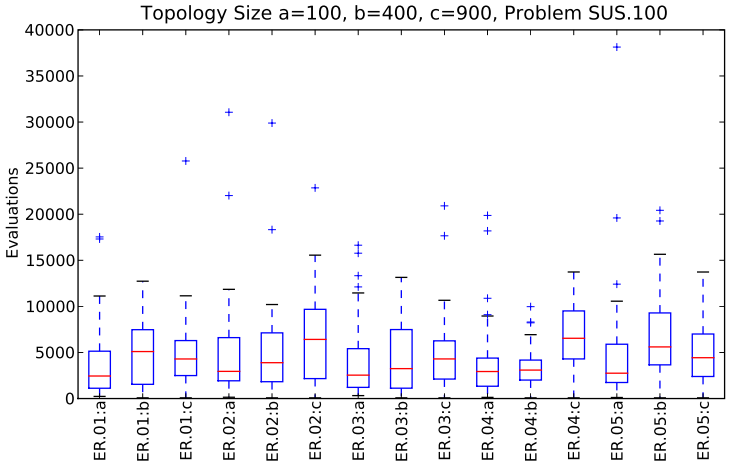


Figure 6.18: ER group scale comparison of ES distribution for the SUS.100 domain. For this simple domain the distributions overlap, and the influence of scale and topology is not as pronounced as that observed in more complex or deceptive domains.

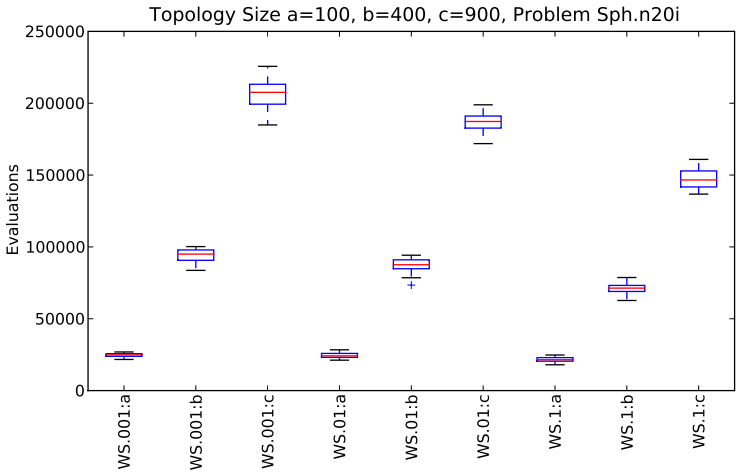


Figure 6.19: WS group scale comparison of ES distribution for Sph.n20i domain. In this search domain the WS models are strongly influenced by scale. A big separation between distributions is clearly evident. Note the mean and distribution decrease as rewiring levels increase – a result of reduced mean path length.

crease as overall topology scale increases, and decrease as the level of rewiring is increased. The SR ratio results correspondingly increase in success and decrease in fixed results. It is a clear representation of the trade off between the number of evaluations required to achieve success, resources allocated (population size and the topologies mean path length) and the efficacy result of the SR values.

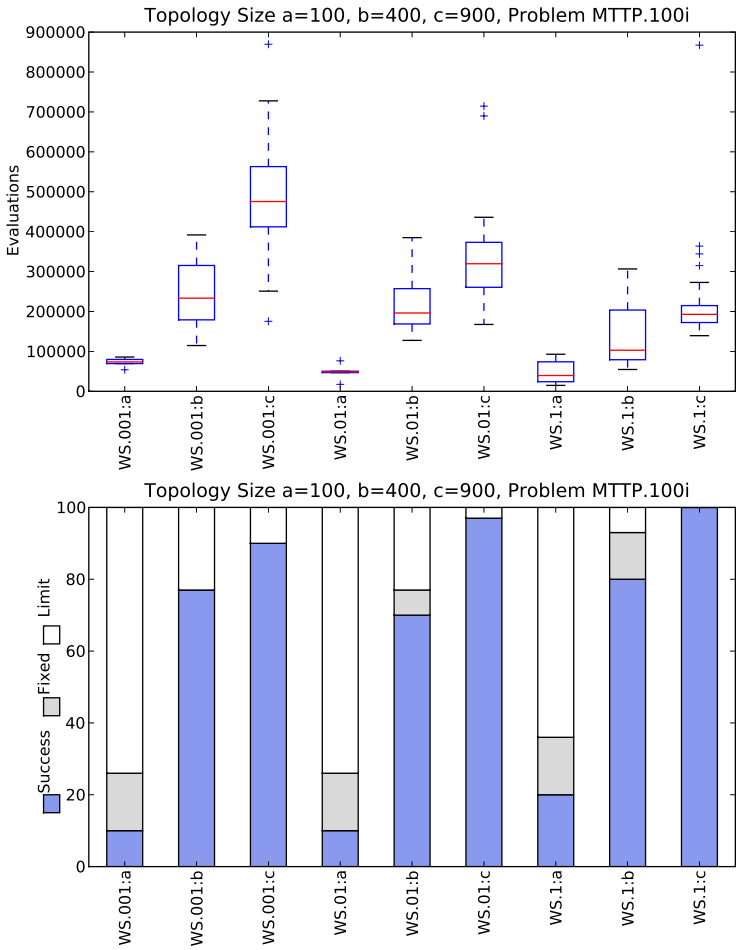


Figure 6.20: WS group scale comparison of ES distribution and SR ratio for MTTP.100i. Both the influence of overall topology scale and the level of rewiring used in the WS model have an influence on the ES distributions. As scale increases the number of success results increases, and as rewiring levels increase the SR levels increase and fixed results decrease.

Full Graph Comparison Results

The results so far have not included full graph population topologies. As part of investigations into the influence of population size and scaling factors, three full graph (panmictic) population topologies were applied to the same range of real and binary value landscapes. Combined result summary tables with the full graph results are presented in Table 6.22 and Table 6.23. A small selection of topologies (T.k8, T.c4, ER.05 and WS.01) is included in the comparison as representative examples of each topology group. Scale results from the three sizes collected earlier (labelled again with “a”, “b” and “c”) are presented with the full graph results.

	Sph.n3i	Sph.n20i	Sph.n3	Sph.n20		Ros.n2i		FMSi	
L.k8:a	2099.8	20695.6	5782.0	**38397.0**	2%	3413.4	80%	-.-	0%
L.k8:b	6971.9	71467.6	20168.7	129933.1	87%	6700.9	100%	-.-	0%
L.k8:c	12913.6	153781.3	41615.4	279040.2	100%	8581.6	100%	194789.2	17%
T.c4:a	2837.0	32452.5	8122.2	-.-	0%	7573.7	86%	-.-	0%
T.c4:b	9478.6	113340.8	27656.3	216085.2	33%	6971.1	100%	188148.0	3%
T.c4:c	**18195.6**	**240025.6**	**56382.3**	**459202.7**	87%	**12141.3**	100%	**323323.5**	7%
ER.05:a	2120.9	21985.9	5759.3	39036.0	2%	3542.0	74%	-.-	0%
ER.05:b	5775.9	58433.2	16589.2	115688.6	27%	6007.4	100%	**45711.0**	3%
ER.05:c	11482.7	113469.5	33241.3	211777.0	87%	10977.4	100%	79279.8	20%
WS.01:a	2224.2	24290.2	6453.2	46083.0	8%	3675.1	84%	-.-	0%
WS.01:b	7479.7	87072.7	22157.1	160693.5	97%	7169.2	100%	232662.0	3%
WS.01:c	14759.3	186426.4	44578.2	333356.5	100%	9205.2	100%	199474.8	17%
Full:a	**1839.7**	**18826.6**	**5154.0**	-.-	0%	**2744.6**	68%	-.-	0%
Full:b	5915.7	57151.0	15753.5	99013.5	4%	6077.1	98%	57443.0	4%
Full:c	11789.0	-.-	32608.4	-.-	0%	10979.5	100%	79449.3	8%

Table 6.22: Full graph AES and SR comparison results for real value landscapes. Where all SR values are 100% or 0% the values are not shown. Ros.n20i results are excluded (no success). When SR is 0% the AES is shown as "-.-".

	MTTP.20i	MTTP.100i		WD4B.5		WD4B.10		SUS.100	SUS.1000
L.k8:a	1363.2*	57469.6	14%	4033.4	76%	10829.5	4%	4105.8	**4400.7**
L.k8:b	4041.6	154359.4	73%	10326.9	100%	29721.7	87%	5471.7	8820.8
L.k8:c	8793.9	222091.0	100%	19854.6	100%	59544.9	100%	5670.0	10114.3
T.c4:a	2196.3	65693.0	4%	16000.0	80%	25534.0	6%	3974.1	4878.8
T.c4:b	5610.1	270006.8	50%	21461.3	100%	88576.2	80%	3819.5	7068.0
T.c4:c	**10534.2**	**497949.1**	70%	**43153.7**	100%	**134554.5**	100%	5948.5	6545.9
ER.05:a	1637.8	47182.7	20%	4321.7	80%	-.-	0%	4756.8	6403.1*
ER.05:b	4000.5	89221.9	80%	7903.1	97%	20499.5	80%	**6955.3**	9692.6
ER.05:c	7662.1	125088.8	100%	13064.8	100%	35827.4	100%	5311.0	10402.4
WS.01:a	1630.5	47938.6	10%	5785.4	90%	17960.2	26%	**3665.7**	4968.7
WS.01:b	4608.4	219767.5	70%	14768.5	100%	44036.3	97%	4391.8	7379.5
WS.01:c	8085.5	336106.3	97%	24393.0	100%	84996.7	100%	4583.3	10551.0
Full:a	**1303.1**	**30838.6**	18%	**3819.6**	62%	**6868.0**	2%	4016.3	7630.7
Full:b	3949.2	60985.1	68%	8343.4	100%	21399.6	72%	4525.2	10216.5
Full:c	7012.2	89460.0	44%	13878.6	100%	35734.9	98%	5083.7	**10734.2**

Table 6.23: Full graph AES and SR comparison results for binary value landscapes. Where all, or almost all, SR values are 100% the values are not shown. The AES value marked with "*" indicates that the SR value is 98%. The SUS.1000e results are excluded.

Figure 6.21 shows full graph comparison results applied to the simple unimodal Sph.n3i domain. The full graph ES distribution profile and scaling influence is similar to other topology instances, and particularly the lattice L.k8 and ER.05 results.

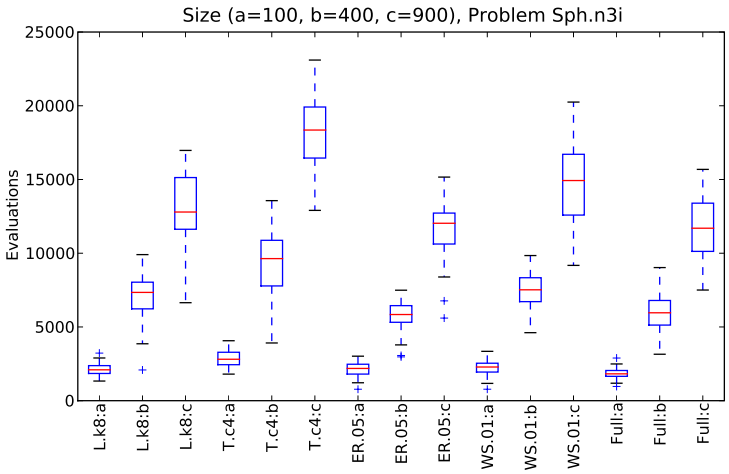


Figure 6.21: Full graph comparison of ES distribution results for Sph.n3i

The MTTP.100i domain is again selected as it provides interesting success ratio results for full graph comparison across scale and topology types. In Figure 6.22 the full graph success rate increases with topology size, and the number of fixed results decreases. This is consistent with other topologies. On comparative performance for this problem domain, full graphs have strong SR values, and low AES and compact ES distributions.

Figure 6.23 shows another ES distribution and success ratio comparison for full graphs applied to the WD4B.10 problem. Here the full graph AES and ES distribution profiles are again low and compact, but the success ratio results are not as strong as other topologies. For example the small-world WS.05 graphs have a larger AES and wider distribution of ES values, though a stronger success rate at all scale levels.

Considering both the earlier MTTP.100i full graph results and the WD4B.10 results, it is possible that for some search domains simple topology (such as a full graph) is not only easier, it is preferred. In other situations a simple topology (with properties such as rapid convergence tendencies) is clearly inadequate, and a more complicated topology (such as a lattice or small-world model) is more appropriate. It may even be possible that a specific or complex topology creates a specialised niche role for some search domains.

As a final comparison of full graph performance and the influence of scale, again consider the simple SUS.1000 search domain. Figure 6.24 shows the ES distribution, and in this set of results it is clear that topology and scale have a much reduced impact; AES values do increase with topology size in most cases, however the spread of the ES distributions is not normalised with many outlier data points, and most ES distributions have a significant amount of overlap with other topology ES distributions.

6.3.4 Circular and Bound Lattices

To compare the influence that a bound lattice has in direct comparison to a circular lattice, the base experiment lattices were reconfigured in bound (non-circular) form. Results were

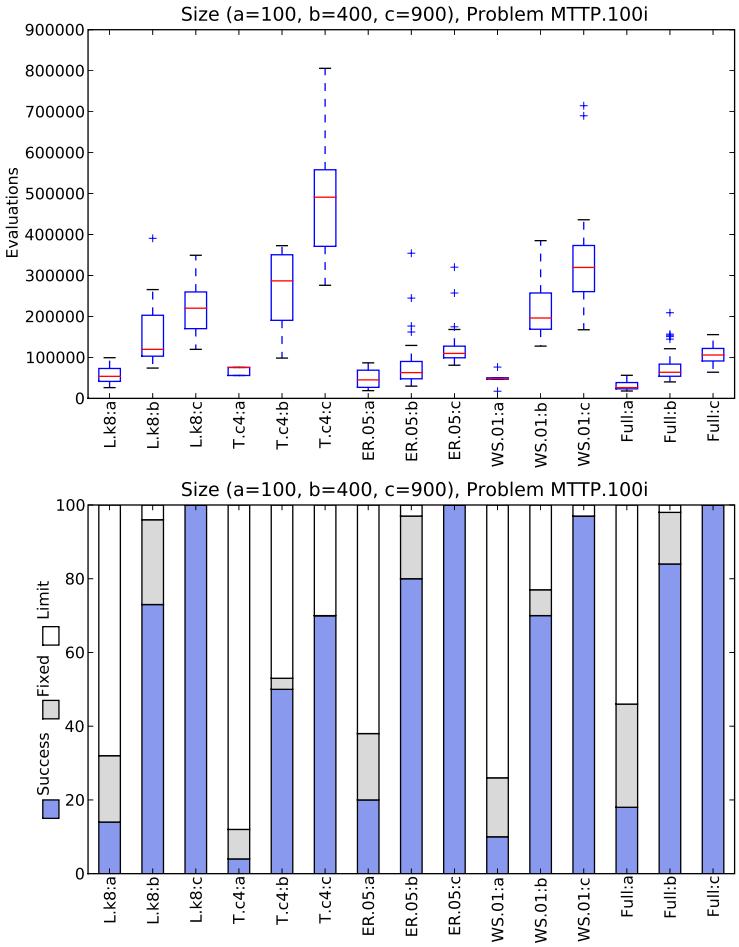


Figure 6.22: Full graph comparison of ES distribution and success ratio for MTTP.100i

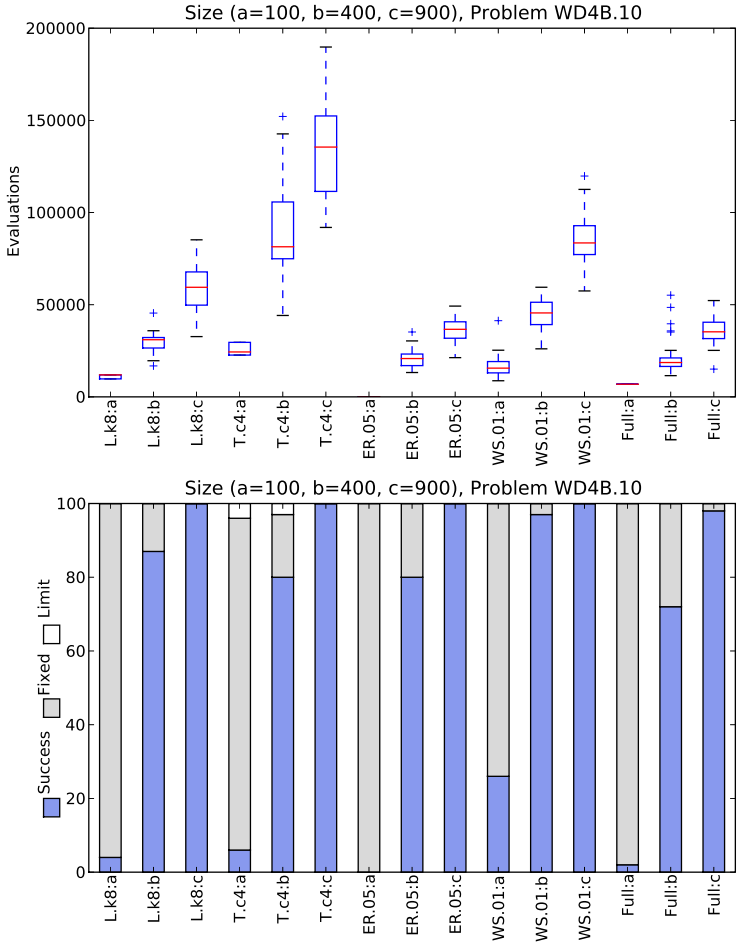


Figure 6.23: Full graph comparison of ES and success ratio for the WD4B.10 domain

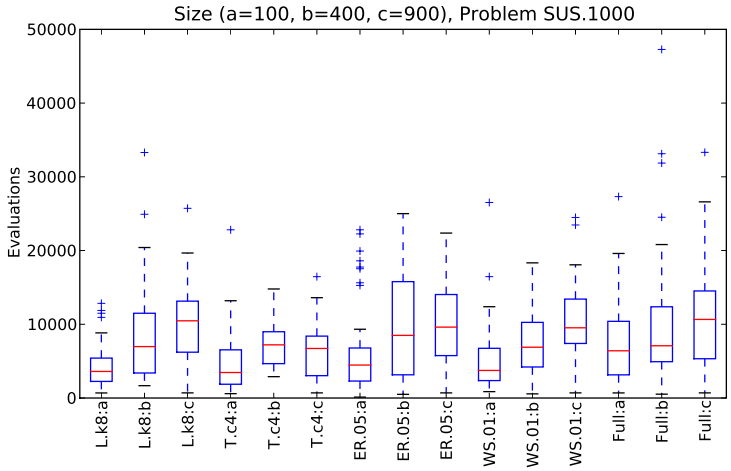


Figure 6.24: SUS.1000 ES distribution compared across topology scales, including three full graph instances and other topology examples

collected on the standard base selection of real and binary problem landscapes. The bound lattice forms are simply denoted with a "b" postfix. Summary results are presented in Table 6.24 and Table 6.25 for real and binary landscape.

Overall, non-circular bound lattices, with their longer mean path length L profile, have an extended range of ES distribution values. Figure 6.25 shows a comparison of the mean path length histograms for circular and bound L.k4 lattices. The change in L has been considered in Chapter 4 and in detail within the topology survey presented in Appendix C. For simple search domains the influence is minimal (Sph.n3i, and SUS domains), but is pronounced for more difficult (Sph.n20i) or deceptive (WD4B) domains. The idea is supported by the ES distribution plots shown in Figure 6.26.

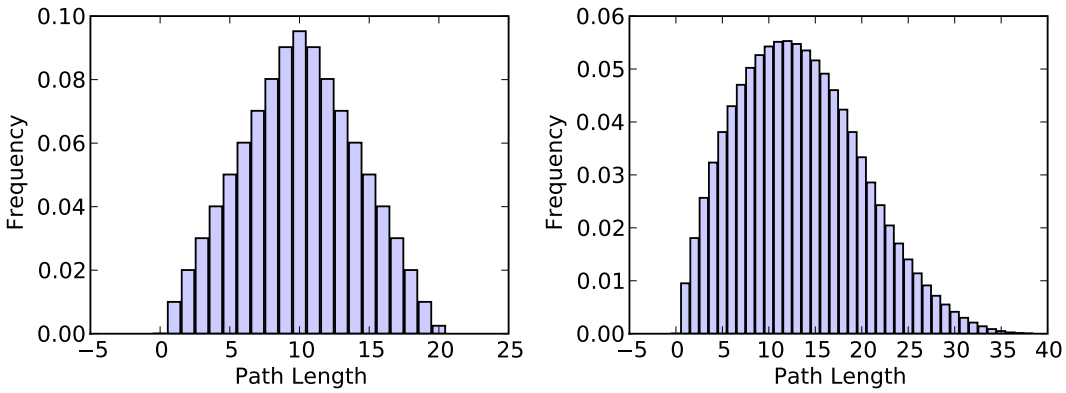


Figure 6.25: Mean path length histogram comparison for (left) circular and (right) bound L.k4 lattice graphs of size $n = 400$. Note that in the bound case a wider range of values and long-tail profile which influences topology based processes.

One suggestion for the benefit of bound lattices is that the increase in mean path length helps to delay premature convergence and maintain healthy diversity. The Ros.n20i domain, which previously has been noted for the lack of success results by any configuration, is able to show how the ratio of fixed and limit result cases changes from circular to bound lattices. Figure 6.27 clearly shows that convergence (always premature) is delayed in bound lattice configurations, sustaining diversity for a longer duration than an equivalent circular lattice.

In the deceptive Ros.n2i and WD4B domains (WD4B.5 and WD4B.10) there is evidence that not only is the fixed ratio reduced, but that the success ratio (shown in Figure 6.28) also increases from circular to bound lattices. Again, the increase in mean path length increases the overall range of the ES distribution. In other words, the search will take longer, but the greater level of lattice separation assists the search in avoiding deceptive and premature convergence.

As a counter example to this, the MTTP.100i domain results show that the SR ratio actually decreases from the circular to the bound cases in most topology configurations. The sample size of data is, however, very small.

	Sph.n3i		Sph.n20i		Sph.n3		Sph.n20		Ros.n2i		Ros.n20i		FMSi
L.k4	2357.3	100%	22893.7	100%	6073.5	100%	42186.0	2%	4544.3	74%	-.-	0%	-.- 0%
L.k4b	2398.7	100%	24313.8	100%	6417.0	100%	44290.0	2%	**2983.1**	76%	-.-	0%	-.- 0%
L.k8	2099.8	100%	20695.6	100%	5782.0	100%	38397.0	2%	3413.4	80%	-.-	0%	-.- 0%
L.k8b	2144.3	100%	22567.1	100%	5985.5	100%	42068.0	2%	3602.8	88%	-.-	0%	-.- 0%
L.k12	2017.9	100%	19892.6	100%	5611.9	100%	37356.0	4%	3302.8	64%	-.-	0%	-.- 0%
L.k12b	2143.8	100%	21272.6	100%	5587.0	100%	43513.0	2%	3301.3	78%	-.-	0%	**9119.0** 2%
L.hk4	2013.1	100%	19986.1	100%	5125.6	100%	39296.0	2%	4075.8	74%	-.-	0%	-.- 0%
L.hk4b	**2552.3**	100%	**26419.0**	100%	**6914.1**	100%	**48817.5**	4%	**5363.8**	88%	-.-	0%	-.- 0%
L.hk8	**1817.4**	100%	**17321.0**	100%	**4676.2**	100%	**32743.0**	2%	3497.7	62%	-.-	0%	-.- 0%
L.hk8b	2221.0	100%	23196.7	100%	6307.4	100%	45201.0	4%	3063.3	80%	-.-	0%	**11360.0** 2%
L.k6	2207.9	100%	21776.2	100%	5854.8	100%	41229.5	4%	4126.5	78%	-.-	0%	-.- 0%
L.k6b	2249.8	100%	23380.9	100%	6340.1	100%	41443.0	2%	3612.6	80%	-.-	0%	-.- 0%
L.hk3	2267.9	100%	23661.5	100%	6360.1	100%	44204.0	2%	3636.2	74%	-.-	0%	-.- 0%
L.hk3b	2280.4	100%	25589.5	100%	6615.7	100%	46507.3	12%	3416.3	78%	-.-	0%	-.- 0%

Table 6.24: Circular vs Bound (b) lattice summary results for real value landscapes. When SR is 0% the AES is shown as "-.-".

	MTTP.20i		MTTP.100i		WD4B.5		WD4B.10		SUS.100		SUS.1000		SUS.1000e	
L.k4	1597.0	100%	58083.7	18%	4385.9	76%	12051.8	12%	3994.5	100%	**4243.6**	100%	3621.5	100%
L.k4b	1587.7	100%	44425.4	10%	5640.2	84%	12794.2	20%	3520.6	100%	4519.2	100%	**2314.3**	100%
L.k8	1363.2	98%	57469.6	14%	4033.4	76%	10829.5	4%	4105.8	100%	4400.7	100%	2468.1	100%
L.k8b	1566.8	100%	34050.7	12%	4859.9	82%	9942.3	8%	**2943.7**	100%	5279.4	100%	2489.4	100%
L.k12	1460.1	100%	43468.4	20%	**3197.7**	76%	**5621.0**	2%	3781.3	100%	5118.0	100%	2669.9	100%
L.k12b	1574.4	98%	38571.3	12%	4306.7	80%	8440.0	2%	3401.1	100%	5386.9	100%	**3646.8**	100%
L.hk4	1496.7	98%	40520.3	6%	4791.3	78%	12146.0	2%	4055.6	96%	4741.0	100%	2682.4	100%
L.hk4b	**1904.0**	100%	51060.0	6%	6292.9	88%	16393.4	20%	3760.8	100%	4624.3	100%	2593.3	100%
L.hk8	**1289.1**	98%	**29066.7**	6%	3787.2	68%	-.-	0%	4130.0	98%	**5885.9**	98%	3178.1	100%
L.hk8b	1632.5	100%	38818.6	10%	6929.3	80%	12401.2	10%	4472.0	100%	4338.5	100%	3090.8	100%
L.k6	1516.2	100%	29696.7	6%	4209.9	84%	9236.0	10%	**4964.5**	100%	5815.7	100%	3440.0	100%
L.k6b	1712.9	100%	41741.0	4%	5295.9	88%	11889.3	12%	4187.1	100%	5217.1	100%	3438.4	100%
L.hk3	1613.1	100%	**66200.5**	16%	4892.0	90%	11623.8	12%	3616.5	100%	4439.7	100%	2727.6	100%
L.hk3b	1738.1	100%	32581.0	4%	**7059.0**	86%	**18837.1**	16%	4648.7	100%	4967.7	100%	3296.6	100%

Table 6.25: Circular vs Bound (b) lattice summary results for binary value landscapes. When SR is 0% the AES is shown as "-.-".

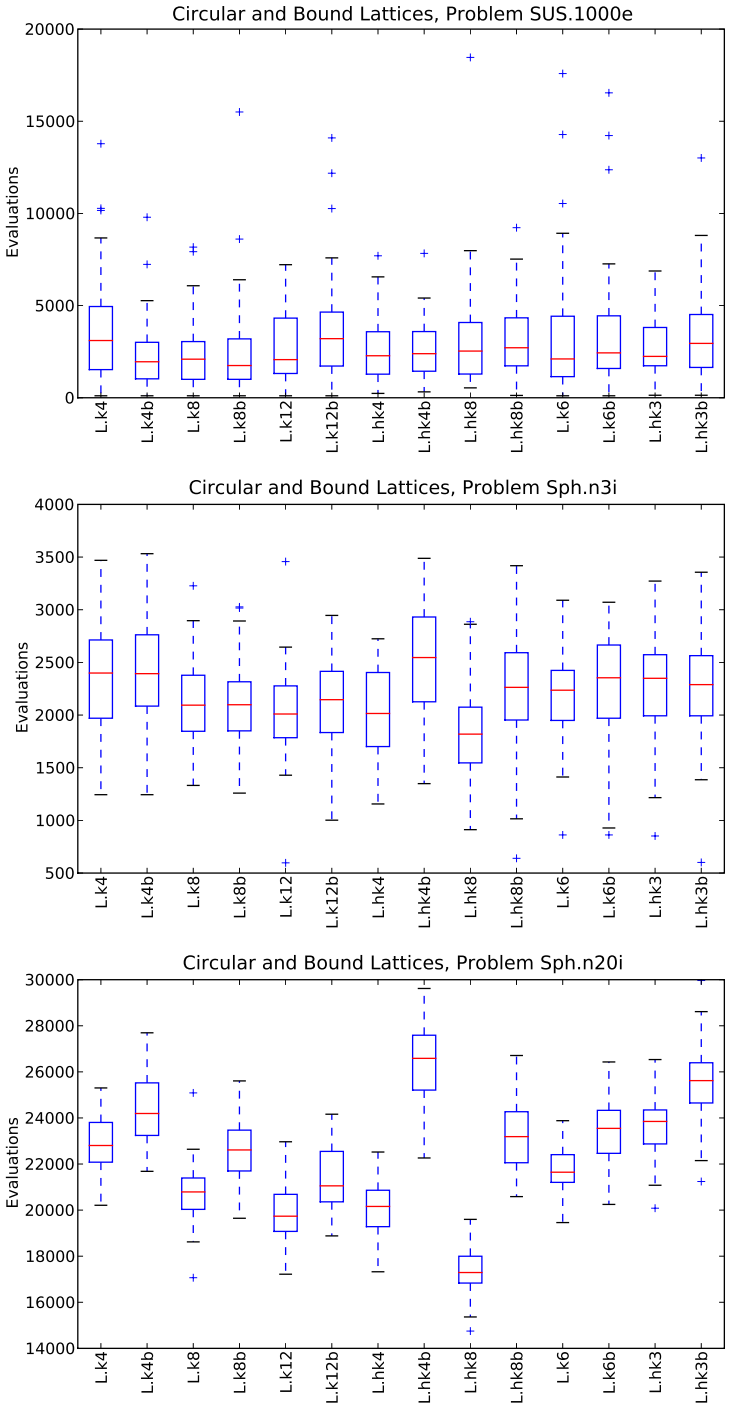


Figure 6.26: Bound lattice influence on SUS.1000i, Sph.n3i and Sph.n20i domain. ES distribution is compared between circular and bound ("b") lattices where the strongest influence is shown in the more complex (but not deceptive) Sph.n20i domain.

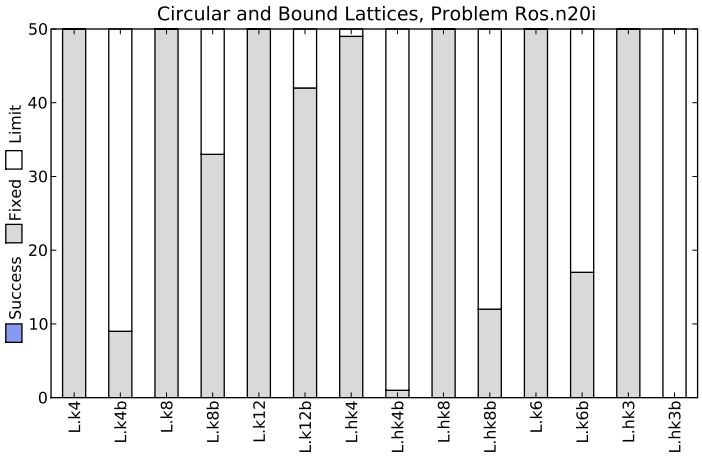


Figure 6.27: Comparison of fixed:limit ratio between circular and bound lattices for the difficult Ros.n20i domain. Bound lattices show a decrease in fixed results.

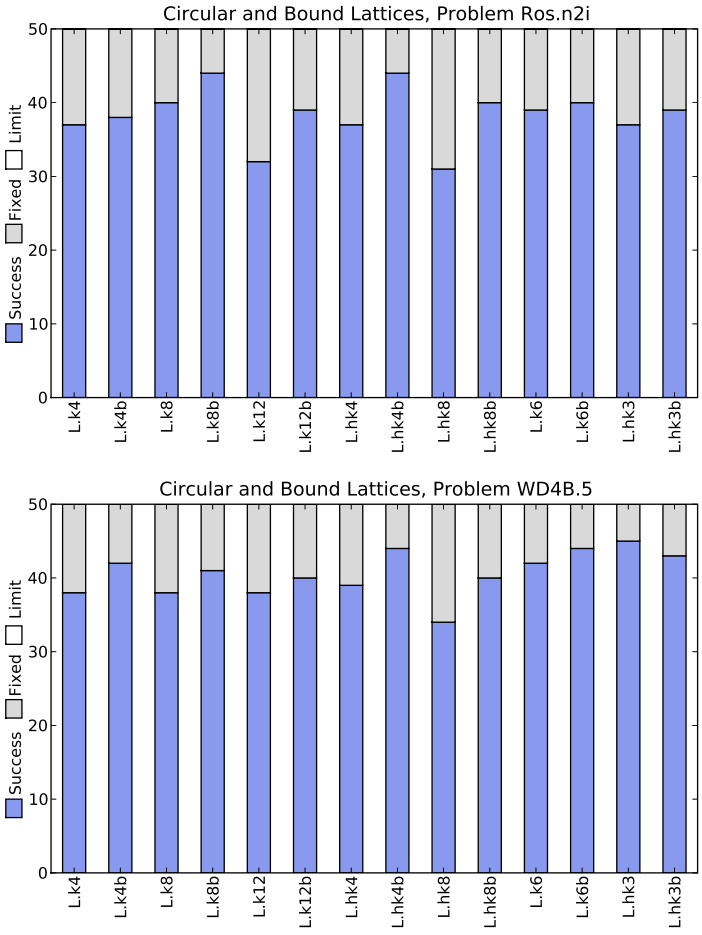


Figure 6.28: Examples of increased success ratio of bound lattice for the Ros.n2i and WD4B.5 domains

6.3.5 Influence of Order and Mate Selection

Update Order and Selection Pressure

Many published works on cellular and distributed population models have also considered the impact of update order on evolutionary process outcomes. Various method of structured population update (parent selection) order have been implemented within the `esec` package.[8] These include:

- *Uniform Random Choice* (URC) with replacement;
- *Uniform Random Sample* (URS) without replacement;
- *Fixed Line Sweep* (FLS);
- *Fixed Line Sweep Reversed* (FLSR);
- *Fixed Random Sweep* (FRS);
- *Fitness Ordered* (FIT) best to worst; and,
- *Fitness Ordered Reversed* (FITR) worst to best.

The notion of a "fixed" order is one that is defined once at the beginning of a simulation and used repeatedly without change throughout the duration of the evolutionary process. For example, a random sequence of topology location can be generated and then used as a "fixed" order, and compared to a fixed line sweep or random sweep order to distinguish between the influence of a dynamic order or a random order – two quite different questions.

A conceptual model suggests that a fixed line sweep update order, or similar orderly process, would allow valuable trait material to be selected and propagated across a population within a single generational period – in essence supporting rapid mixing and convergence of good solution components. Published work suggested that for some problem domains a fixed line sweep order is beneficial, and for others of neutral or harmful influence [318, 307, 5, 129]. A series of structured topology results are presented to resolve this possibility.

If the line sweep order does encourage a rapid convergence and transfer of useful material it is also possible to construct other update order sequences that also try to encourage useful traits to be reproductively transferred and mixed, and not simply in a singular manner. It would be possible for a number of different "flow" terrains incorporating "source" and "sink" regions of the topology that are created simply by an ordered update sequence. In line with this thinking, three novel update orders – SpiralIn, SpiralOut and ZigZag – have been implemented in the `esec` package and are considered.

A specific update order is not enough to ensure the propagation of valuable traits; without strong mate selection valuable traits may be ignored, and similarly without strong competitive selection valuable new recombinations may be ignored. The influence of strong mate selection and replacement competition is also considered with line sequence update orders.

[8] Specifically, see the `Structured` population class in the `esec.system.population` module. It is the population which applies the order to a population graph.

Line Sequences

Figure 6.29 show representations of the FLS and ZigZag order update patterns. A reverse FLS (denoted FLSR) can also be used, but for a typical lattice it is simply a rotation of the update order. However, the FLS order can be applied to graphs that are "grown" in a particular order such as trees or BA scale free graphs. In these cases old-to-new (FLS) or new-to-old (FLSR) order may have a structural influence. A fixed FLS order makes little distinctive sense when applied to random graph models.

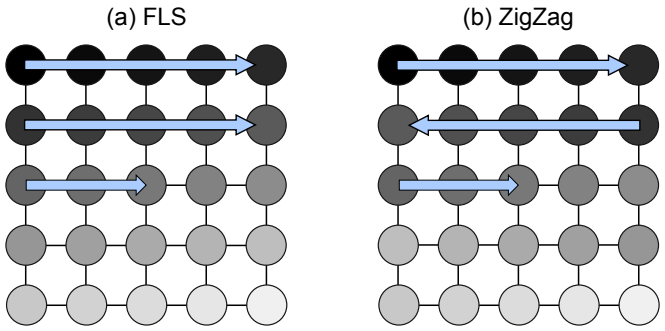


Figure 6.29: Representations of (a) Fixed Line Sweep (FLS) and (b) ZigZag lattice update sequences using arrows and a gradient of grey fill colours to emphasis order

Assuming order is able to propagate useful traits, the intention of the "ZigZag" order is not to interrupt the completion of one row by beginning again at the other side of the topology. Instead, the zig-zag order maintains "contact" with the current strong traits and so has the maximum opportunity to move strong traits with the update order.

Using the $n = 100$ based configuration of problem domains and topologies, a subset of non-random or structured topologies is selected: lattices, trees, WS and BA models. The base experiment results used a uniform random sample (URS) update order by default. New batches are created using FLS and FLSR update orders. The detailed summary report is provided as CDROM appendices. Table 6.26 and Table 6.27 present the summary AES and SR results for a combined comparison report that includes the base results ("a") and compares this with the FLS ("b") and FLRS ("c") results.

Not all of the topology groups used in the base configuration are appropriate for this line of inquiry, and so only the lattice, tree, WS (based on a lattice) and BA topologies are used. In the electronic report for this investigation (see Appendix F), the summary results are divided into results for each problem domain, showing how Lattice, Tree and the WS/BA model groups performed with random, FLS and FLSR update order.

The essential configuration details for the FLS and FLSR batch become the importing of the base configuration, and the setting of topology update order,[9] and the exclusion of ER, Star and MR.05 topologies.

As a representative example, for most lattice topologies, Figure 6.30 shows that there is a decrease in the AES for both FLS and FLSR with respect to the default random order results (URS). As expected the variation between FLS and FLSR is nominal as they are

[9] `cfg.system.topology.order = 'FLS'` or `'FLSR'`

	Sph.n3i		Sph.n20i		Sph.n3		Sph.n20		Ros.n2i		Ros.n20i		FMSi
L.k4:a	**2357.3**	100%	**22893.7**	100%	6073.5	100%	42186.0	2%	**4544.3**	74%	-.-	0%	-.-	0%
L.k4:b	2215.0	100%	22485.2	100%	**6118.2**	100%	**43401.0**	4%	4343.6	82%	-.-	0%	-.-	0%
L.k4:c	2098.1	100%	22064.7	100%	6100.5	100%	41604.5	4%	3975.8	78%	-.-	0%	-.-	0%
L.k6:a	2207.9	100%	21776.2	100%	5854.8	100%	41229.5	4%	4126.5	78%	-.-	0%	-.-	0%
L.k6:b	2203.2	100%	22069.2	100%	5938.3	100%	-.-	0%	**2185.2**	70%	-.-	0%	-.-	0%
L.k6:c	2273.4	100%	21568.9	100%	5985.3	100%	41106.4	10%	3144.9	72%	-.-	0%	-.-	0%
L.k8:a	2099.8	100%	20695.6	100%	5782.0	100%	38397.0	2%	3413.4	80%	-.-	0%	-.-	0%
L.k8:b	2050.6	100%	21088.5	100%	5689.1	100%	40660.0	2%	2862.8	68%	-.-	0%	-.-	0%
L.k8:c	2031.2	100%	21118.9	100%	5510.6	100%	40741.0	2%	2963.6	76%	-.-	0%	-.-	0%
L.k12:a	2017.9	100%	19892.6	100%	5611.9	100%	**37356.0**	4%	3302.8	64%	-.-	0%	-.-	0%
L.k12:b	**2011.7**	100%	19824.5	100%	5356.9	100%	-.-	0%	2882.6	74%	-.-	0%	-.-	0%
L.k12:c	2016.1	100%	**19807.7**	100%	**5215.2**	100%	38096.0	2%	3721.7	76%	-.-	0%	-.-	0%

Table 6.26: Random, FLS and FLSR update order summary results for real value landscapes. When SR is 0% the AES is shown as "-.-".

	MTTP.20i		MTTP.100i		WD4B.5		WD4B.10		SUS.100		SUS.1000		SUS.1000e	
L.k4:a	1597.0	100%	58083.7	18%	4385.9	76%	**12051.8**	12%	3994.5	100%	**4243.6**	100%	**3621.5**	100%
L.k4:b	1610.1	100%	45543.8	10%	5222.3	78%	7874.0	4%	4281.9	100%	5297.0	100%	**2271.1**	100%
L.k4:c	1607.9	100%	43317.8	10%	**5450.0**	90%	9944.2	10%	3848.6	100%	4925.9	100%	3062.4	100%
L.k6:a	1516.2	100%	29696.7	6%	4209.9	84%	9236.0	10%	**4964.5**	100%	5815.7	100%	3440.0	100%
L.k6:b	1601.3	98%	47775.3	6%	4709.1	76%	6079.0	4%	3893.6	100%	6169.6	100%	2473.4	100%
L.k6:c	**1675.1**	100%	43255.3	8%	4376.2	84%	7419.5	8%	4171.8	100%	**7422.4**	96%	3203.1	100%
L.k8:a	**1363.2**	98%	57469.6	14%	4033.4	76%	10829.5	4%	4105.8	100%	4400.7	100%	2468.1	100%
L.k8:b	1562.8	100%	43716.6	14%	4689.8	66%	7026.0	2%	3441.3	100%	5846.7	100%	3037.5	100%
L.k8:c	1525.2	100%	**60263.0**	26%	4003.5	84%	-.-	0%	**3005.1**	100%	4573.5	100%	3011.2	100%
L.k12:a	1460.1	100%	43468.4	20%	**3197.7**	76%	5621.0	2%	3781.3	100%	5118.0	100%	2669.9	100%
L.k12:b	1469.8	98%	49529.4	18%	3863.7	74%	**5376.7**	6%	3532.5	98%	5881.1	98%	2888.3	100%
L.k12:c	1492.8	100%	**21992.3**	6%	3809.1	74%	9445.5	4%	4291.4	100%	5937.1	100%	3280.0	100%

Table 6.27: Random, FLS and FLSR update order summary results for binary value landscapes. When SR is 0% the AES is shown as "-.-".

equivalent. Two of the hollow lattices (L.hk4 and L.hk8) actually showed a strong reverse trend where the fixed line orders have poorer (higher) ES distributions than the random comparison. This is not the case for the "honeycomb" hollow L.hk3 lattice. Perhaps the uniform nature of the lattice accounts for this.

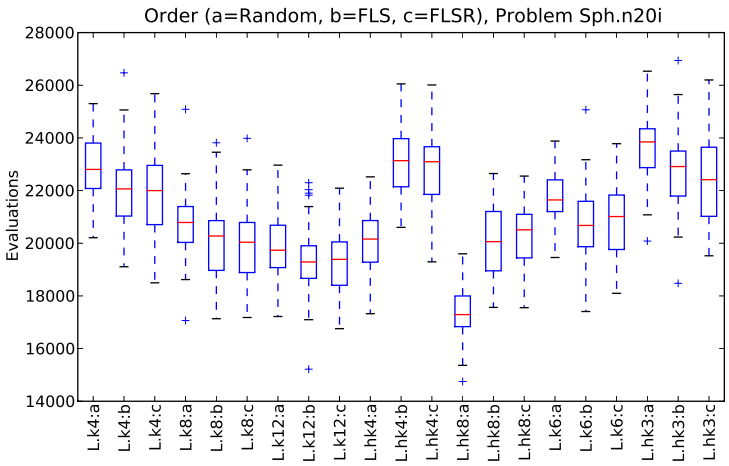


Figure 6.30: Random (URS), FLS and FLSR lattice update order influence on the Sph.n20i domain

The influence of FLS and FLSR order on the tree models seems negligible for the Sph.n20i domain. WS models have a slight AES decrease with both FLS and FLSR, while the BA model seems to have a slight increase in AES for both FLS and FLSR. Overall though, the basic influence of FLS and FLSR order is not strong, and for many domains clearly negligible.

Investigation of the "ZigZig" update order is restricted to lattice topologies where the order conceptually makes the most sense. The lattice order is presented from lowest to highest degree, as general results are correlated. For comparison the random and FLS orders are presented along with the ZigZag results.

Results for most problem domains show little or weak order influence in comparison to other stochastic variation. The strongest results are again from the Sph.n20i and Sph.n3i domains. Figure 6.31 shows that increasing lattice degree has a direct correlation to AES value, but importantly that FLS and ZigZag update order do influence the AES value and the ES distribution to a minor degree. ZigZag order is not, however, obviously more influential than the FLS order.

The role of selection, then, must be considered further. Using the same set of selected topologies and the same update order, the mate selection of the species configuration is specified as "best" from the default setting of "binary_tournament".[10] In this way each "parent" will, in turn, select the very best neighbour to be involved in reproduction. By default, the best of the two offspring from one point crossover is retained, and the survivor offspring competes with the parent for deterministic replacement survival.

When this stronger level of selection is applied to the initial FLS order results, the comparison shows a significant improvement in AES and ES distribution results (Figure 6.33).

[10] `cfg.species.recombine.mate.selection = 'best'`

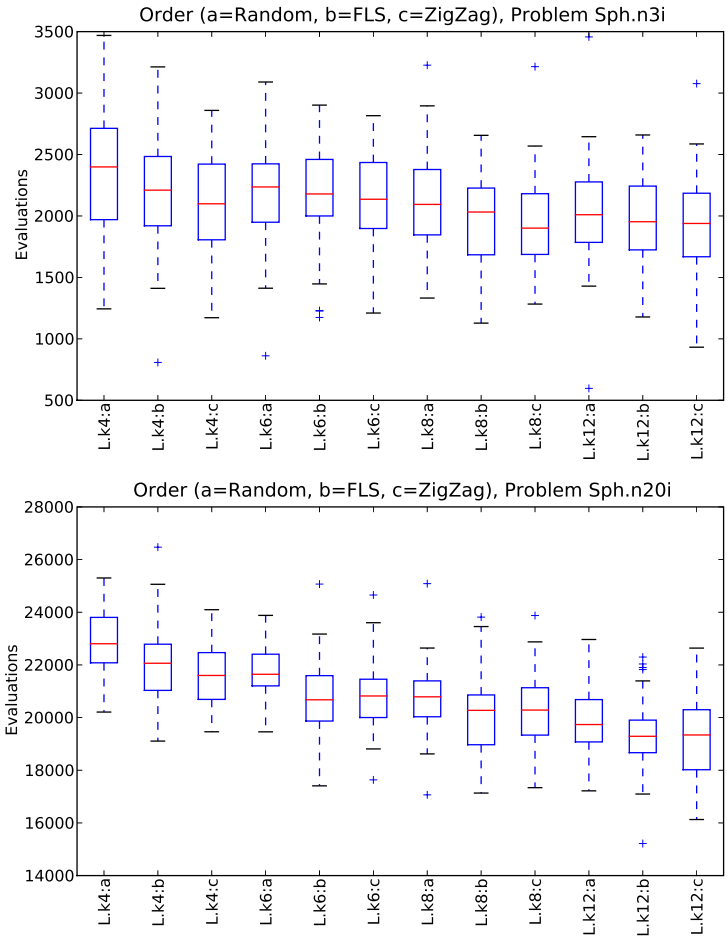


Figure 6.31: Random (URS), FLS and ZigZig lattice update order influence on the Sph.n3i and Sph.n20i domains. FLS and ZigZag order reduce the AES and ES distribution of values.

In simple and non-deceptive domains this is clearly beneficial. However it also creates premature convergence issues for many of the problem domains (not just the difficult or deceptive). For example, Figure 6.33 shows how for the WD4B.5 domain the use of "best" mate selection significantly increases the ratio of "fixed" outcomes. As the "best" method of mate selection is only applied to the FLS order, result comparison with the random order is not made.

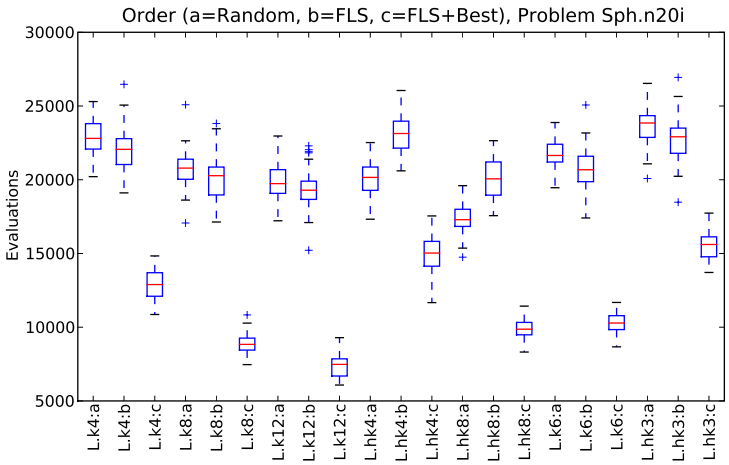


Figure 6.32: Random (URS), FLS and FLS+Best mate selection result on lattice topologies and applied to the Sph.n20i problem domain. The stronger level of mate selection pressure clearly reduces the value of AES results.

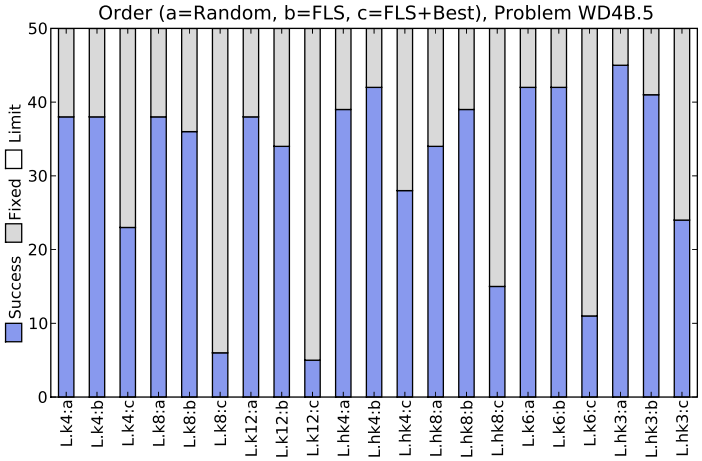


Figure 6.33: Random (URS), FLS and FLS+Best mate selection result on lattice topologies and applied to the WD4B.5 problem domain. Note that the additional selection pressure consistently leads to more unsuccessful "fixed" result instance.

Summary results have also been compiled and presented, as electronic appendices, for the comparison of FLS+Best with FLSR+Best results. The main objective of this comparison was to see if the ordered structures of trees and the BA growth model showed any performance differences between the FLS and FLSR update orders. There are some minor indications that the FLS order does better (lower AES values) than FLRS, however

the results are inconclusive. The results do support that best mate selection increases the ratio of premature convergence.

Spiral Sequences

Returning to the spiral alternative for update order, results for both a SpiralIn and SpiralOut update order (Figure 6.34), with and without best mate selection, have also been collected. Full summaries are presented as CDROM appendices, and the summary tables are presented in Table 6.28 and Table 6.29 for real-valued and binary-valued results.

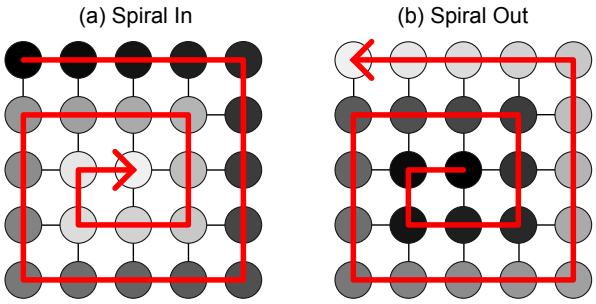


Figure 6.34: Representations of (a) Spiral In and (b) Spiral Out lattice update sequences using a line arrow and a gradient of grey fill colours to emphasis order

The comparison has been limited in this case to regular lattices and excludes the irregular hollow variations. Without strong mate selection, the results show some improvement in AES values with both spiral in and out update orders, yet there is no clear indication that either "in" or "out" is the preferred option for general search domains. As an example see Figure 6.35.

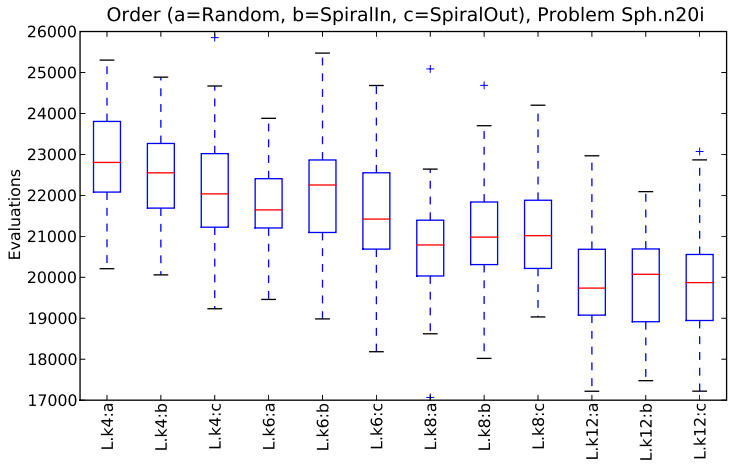


Figure 6.35: Random, SpiralIn and SpiralOut update order on lattice applied to Sph.n20i. The result presented here is using default mate selection (not "best"). In some lattice groups the spiral update orders have a positive influence, and spiral "out" appears to be slightly better than spiral "in".

The use of flow update order patterns within a population remains an interesting open question. It is quite obvious from these results that there may be some positive benefit

	Sph.n3i		Sph.n20i		Sph.n3		Sph.n20		Ros.n2i		Ros.n20i		FMSi
L.k4:a	**2357.3**	100%	**22893.7**	100%	**6073.5**	100%	**42186.0**	2%	**4544.3**	74%	-.-	0%	0%
L.k4:b	1439.1	100%	14220.8	100%	3916.0	98%	-.-	0%	2479.7	58%	-.-	0%	0%
L.k4:c	1441.5	100%	14266.8	100%	4071.6	96%	-.-	0%	2128.6	56%	-.-	0%	0%
L.k6:a	2207.9	100%	21776.2	100%	5854.8	100%	41229.5	4%	4126.5	78%	-.-	0%	0%
L.k6:b	1231.4	100%	12972.6	100%	3406.9	92%	-.-	0%	2953.6	56%	-.-	0%	0%
L.k6:c	1303.8	100%	12697.2	100%	3331.0	94%	-.-	0%	1900.5	62%	-.-	0%	0%
L.k8:a	2099.8	100%	20695.6	100%	5782.0	100%	38397.0	2%	3413.4	80%	-.-	0%	0%
L.k8:b	1147.1	100%	11891.7	100%	3166.3	96%	-.-	0%	1656.1	46%	-.-	0%	0%
L.k8:c	1161.2	100%	11743.5	100%	3227.2	94%	-.-	0%	1729.5	44%	-.-	0%	0%
L.k12:a	2017.9	100%	19892.6	100%	5611.9	100%	**37356.0**	4%	3302.8	64%	-.-	0%	0%
L.k12:b	915.4	100%	**9000.7**	100%	**2352.3**	88%	-.-	0%	**1461.3**	42%	-.-	0%	0%
L.k12:c	**899.6**	100%	9046.5	100%	2453.0	84%	-.-	0%	2403.3	36%	-.-	0%	0%

Table 6.28: Random, SpiralIn and SpiralOut update order, with best mate selection, summary results for real value landscapes. When SR is 0% the AES is shown as "-.-".

	MTTP.20i		MTTP.100i		WD4B.5		WD4B.10		SUS.100		SUS.1000		SUS.1000e	
L.k4:a	**1597.0**	100%	58083.7	18%	4385.9	76%	12051.8	12%	3994.5	100%	4243.6	100%	**3621.5**	100%
L.k4:b	1208.1	100%	43670.0	16%	3861.1	54%	9054.0	2%	4452.7	100%	4878.1	100%	2690.6	100%
L.k4:c	1191.4	98%	48470.0	8%	2947.4	44%	**16067.0**	2%	3175.5	100%	4068.5	100%	2914.2	100%
L.k6:a	1516.2	100%	29696.7	6%	4209.9	84%	9236.0	10%	4964.5	100%	**5815.7**	100%	3440.0	100%
L.k6:b	1149.4	100%	25251.5	4%	2875.0	42%	-.-	0%	**5130.8**	98%	3804.6	98%	2699.7	100%
L.k6:c	1238.2	94%	**24348.8**	10%	**4445.2**	44%	-.-	0%	3727.9	96%	5290.4	94%	2484.2	100%
L.k8:a	1363.2	98%	57469.6	14%	4033.4	76%	10829.5	4%	4105.8	100%	4400.7	100%	2468.1	100%
L.k8:b	1211.8	98%	**66236.8**	10%	2521.2	36%	-.-	0%	**2959.3**	98%	4930.3	94%	**2376.5**	100%
L.k8:c	1035.7	90%	61856.0	4%	2441.6	30%	-.-	0%	4379.8	90%	4459.5	92%	3294.4	98%
L.k12:a	1460.1	100%	43468.4	20%	3197.7	76%	**5621.0**	2%	3781.3	100%	5118.0	100%	2669.9	100%
L.k12:b	**932.8**	80%	50043.0	6%	2298.8	12%	-.-	0%	3312.1	92%	**3173.9**	88%	2761.5	94%
L.k12:c	972.7	82%	36740.0	4%	**1483.6**	20%	-.-	0%	3705.0	84%	3214.4	90%	2388.1	98%

Table 6.29: Random, SpiralIn and SpiralOut update order, with best mate selection, summary results for binary value landscapes. When SR is 0% the AES is shown as "-.-".

from the use of particular update orders, however for this base configuration without strong mate selection pressure, the influence of order is reduced, and certainly not as strong as other topological influence such as connection architecture and size. To consider update order more thoroughly is outside the scope of the work presented here.

Fitness Sequence

As a final variation of investigation into the possible influence of update order with population structures, an alternative fitness based scheme is proposed. Fitness based operators are central to evolution algorithms, as they influence the selection and replacement of individuals at (potentially) many different process stages.

Assuming that an update order can potentially propagate useful traits (albeit also increasing levels of premature convergence), it may be useful to order updates based directly on a ranked global fitness measure. Unlike the lattice specific line and spiral orders, a fitness based order can be applied to any structured topology (and an EA using a gap-based replacement model).

A detailed summary report for the FIT and FITR update order results for all topologies and all problem domains is included as CDROM appendices. The report breaks results into topology groups for comparison of (a) random, (b) FIT and (c) FITR applied to each problem. The report also includes landscape based summaries and alternative topology group summaries, and is one of the largest reports created for the thesis.

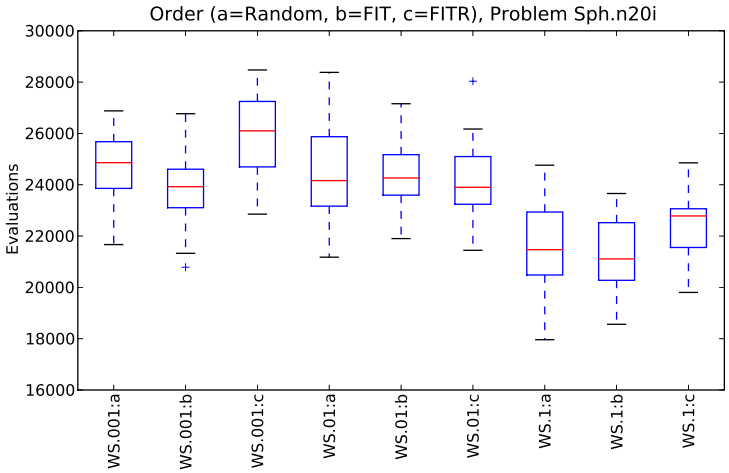


Figure 6.36: Random, FIT and FITR update order on WS topology applied to Sph.n20i

The results are interesting, but again the influence of order is less than other contributions such as overall topology architecture and the sensitivity of scale. In several examples, mainly for simple problem domains, FIT fitness based order improves both the minimum ES values, and reduces the AES and overall ES distribution level (Figure 6.36). Conversely the reverse fitness order FITR shows an increase in AES and ES distribution values above the base random order. This may indicate that a reverse order propagates less useful traits, or at least disrupts other useful traits being promoted. It could be that reverse order (or a partial form used with elitism) could be used as a mechanism for maintaining or encouraging the preservation of diversity (and improving success ratio results).

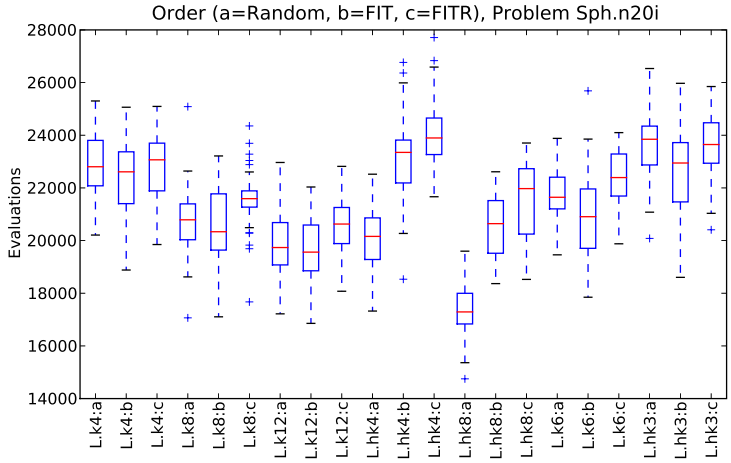


Figure 6.37: Random, FIT and FITR update order on lattice topology applied to Sph.n20i.

However this is speculation and additional experiments would be needed to validate these ideas.

Within the lattice results (Figure 6.37) most lattices with the FIT and FITR order follow the trends already described, with the notable exceptions of the hollow lattices (L.hk8 and L.hk4) which seem to be strongly disrupted by both the forward and reverse fitness based orders. This may have something to do with the non-homogenous number of connections each location has.

6.3.6 Juveniles with Delayed Competition

As discussed earlier ecological observations and principles suggest that both fitness assessment of individual and competitive replacement are delayed events within an organisms lifespan. A species population can contain a structure based on the age of individuals, and different models of interacting groups based on non-genetic factors such as culture and opportunity.

One ecological model inspired from biology that can easily be applied is that of delayed replacement. Essentially, it is the idea that new offspring (juveniles) rarely compete immediately or directly with adults of the same species. Similarly, a juveniles' competitive survival may relate to a different set of species than those that an adult member of the same species must interact with.

Delayed replacement is implemented in `esec` such that new offspring are allocated to the same topology "location" as the designated "parent" individual. The offspring remains with other offspring until a step time (update) of delay has elapsed, at which time the offspring must compete with other offspring and parents for its survival.

Support for this model is the that a parent individual has a reduced immediate competitive survival pressure, and may be able to contribute (reproduce) its unique traits and variation on more occasions. Secondly, in a model where multiple children are created each reproductive update, all the offspring can be retained and allowed to compete, enabling offspring competition before disrupting the adult breeding population.

Interestingly, this model increases the effective size of the population. It would be

reasonable then, in a thorough analysis, to measure or gauge the extent of "carrying capacity" that the delayed or juvenile model creates, and whether an equivalent simple adult model of the same capacity would display similar performance features.

Using the base experiment, a range of exponentially increasing delay values was selected $(4, 8, 16, 32, 64)$ and applied to a subset of the base topologies and full graphs: L.k4, L.k12, ER.05 and Full. Of the base problem domains, the Sph.n3i and SUS.1000e were excluded due to their limited differential contributions in previous results. As a practical limit the maximum number of evaluations is set to $100,000$, which in some cases introduce arbitrary "limit" results which need to be considered with ratio observations. Summary result tables are shown in Table 6.30 and Table 6.31.

Figure 6.38 clearly shows that the effect of delayed replacement increases the ES distribution in proportion to the delayed value. As already discussed, this may be due to the delay capacity acting as a population size increase, as scale results from earlier investigation also showed that AES values increase linearly with population size increase. Interestingly the full topologies are not influenced nearly as much as the sparse lattice, ER and MR models. If this is a general relationship to topology density, it explains why the L.k12 respond to a lesser degree than the sparse L.k4 topology.

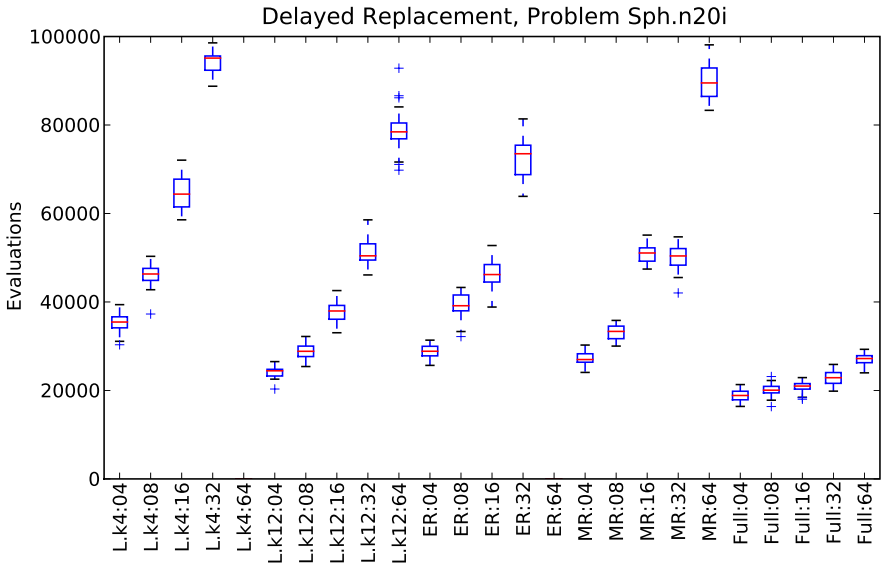


Figure 6.38: Delayed replacement comparison for the Sph.n20i domain.

The ES distribution results shown in Figure 6.39 continue to support the same topology related observations. The full graphs are least affected, and the dense L.k12 is less influenced than the L.k4 graph.

Results for the WD4B.10 domain again show (Figure 6.40) the exponential increase in ES distribution mean (AES) with respect to the delayed replacement parameter. The general spread of ES distribution is greater across all topologies in response to the deceptive nature of the domain. The influence of this deceptive domain is also seen in the success ratio results. By looking at the AES results matched to the success rates, a topology that results in greater search time is also more likely to succeed. This relationship can be seen clearly in all the lattice models, and to a lesser degree in ER and MR models. A full graph

	Sph.n20i		Sph.n3		Sph.n20		Ros.n2i		Ros.n20i		FMSi	
L.k4:04	35167.8	100%	8984.8	100%	60436.0	7%	3731.5	87%	-.-	0%	-.-	0%
L.k4:08	45937.8	100%	11669.5	100%	81678.5	37%	3887.0	80%	-.-	0%	39020.0	3%
L.k4:16	64758.0	100%	16311.9	100%	-.-	0%	5412.5	83%	-.-	0%	**33826.0**	3%
L.k4:32	**93916.5**	53%	24684.2	100%	-.-	0%	8458.3	93%	-.-	0%	-.-	0%
L.k4:64	-.-	0%	**38059.4**	100%	-.-	0%	8391.8	100%	-.-	0%	**76231.0**	3%
L.k12:04	24107.2	100%	6663.2	100%	43195.0	10%	3614.3	93%	-.-	0%	-.-	0%
L.k12:08	28679.4	100%	7396.2	100%	50410.0	10%	3384.6	83%	-.-	0%	-.-	0%
L.k12:16	37810.1	100%	9562.7	100%	66773.3	13%	5353.0	87%	-.-	0%	-.-	0%
L.k12:32	51441.0	100%	13742.9	100%	89529.3	27%	5222.8	97%	-.-	0%	-.-	0%
L.k12:64	78915.0	100%	19595.1	100%	-.-	0%	5806.8	97%	-.-	0%	43255.0	3%
ER:04	28786.7	100%	7538.9	100%	51494.0	7%	3441.0	83%	-.-	0%	-.-	0%
ER:08	39285.2	100%	10153.3	100%	60993.5	7%	4694.2	83%	-.-	0%	-.-	0%
ER:16	46244.5	100%	13026.5	100%	**91466.0**	10%	6111.7	87%	-.-	0%	-.-	0%
ER:32	72496.2	100%	18192.6	100%	-.-	0%	6658.9	90%	-.-	0%	45225.0	3%
ER:64	-.-	0%	29578.1	100%	-.-	0%	**9914.3**	93%	-.-	0%	-.-	0%
MR:04	27186.7	100%	7450.7	100%	-.-	0%	7388.8	83%	-.-	0%	-.-	0%
MR:08	33128.4	100%	8956.8	100%	-.-	0%	4943.6	83%	-.-	0%	-.-	0%
MR:16	50843.4	100%	9967.6	100%	-.-	0%	5131.2	90%	-.-	0%	-.-	0%
MR:32	50208.6	100%	16963.8	100%	-.-	0%	6686.5	87%	-.-	0%	-.-	0%
MR:64	89692.5	97%	23773.0	100%	-.-	0%	7658.4	93%	-.-	0%	69785.0	3%
Full:04	**18914.0**	100%	5362.6	100%	-.-	0%	**2197.6**	50%	-.-	0%	-.-	0%
Full:08	20068.2	100%	**5272.3**	100%	**37015.0**	3%	2832.7	53%	-.-	0%	-.-	0%
Full:16	20887.2	100%	5702.8	100%	-.-	0%	2908.9	67%	-.-	0%	-.-	0%
Full:32	22824.5	100%	6300.5	100%	-.-	0%	3172.0	73%	-.-	0%	-.-	0%
Full:64	27048.0	100%	7316.4	100%	-.-	0%	2519.1	70%	-.-	0%	-.-	0%

Table 6.30: Delayed juvenile replacement summary results for real value landscapes. When SR is 0% the AES is shown as "-.-".

	MTTP.20i		MTTP.100i		WD4B.5		WD4B.10		SUS.100		SUS.1000	
L.k4:04	2306.3	100%	75755.4	17%	6019.9	93%	13637.1	33%	4022.3	100%	5332.7	100%
L.k4:08	2839.6	100%	51481.0	23%	7014.2	100%	16459.6	47%	**6100.0**	100%	5046.4	100%
L.k4:16	3869.2	100%	56129.4	23%	7483.0	100%	18833.7	53%	4232.1	100%	6589.1	100%
L.k4:32	5645.2	100%	71772.6	33%	8470.7	100%	24384.3	77%	4233.4	100%	6534.5	100%
L.k4:64	**8463.9**	100%	88906.2	17%	**13022.4**	100%	**37309.5**	97%	5414.0	100%	**9369.7**	100%
L.k12:04	1767.3	100%	66517.0	3%	3649.6	67%	6162.5	7%	2770.2	100%	6755.2	100%
L.k12:08	1807.0	100%	56105.3	13%	4213.1	80%	8871.4	17%	3200.3	100%	6871.0	100%
L.k12:16	2284.8	100%	58808.3	20%	4519.2	100%	11237.1	37%	4722.0	100%	8454.2	100%
L.k12:32	3137.6	100%	51867.6	23%	5819.3	93%	12016.3	40%	3528.0	100%	6895.8	100%
L.k12:64	4544.3	100%	63986.5	27%	6878.0	100%	18074.4	57%	4421.8	100%	7829.1	100%
ER:04	2098.2	97%	69559.7	10%	5574.8	80%	11537.3	23%	4376.5	100%	8916.3	100%
ER:08	2460.4	100%	70669.6	17%	4858.9	90%	12521.3	10%	4869.2	100%	5416.9	100%
ER:16	3218.4	100%	44315.3	10%	6586.5	100%	15856.1	33%	4670.6	100%	7051.9	100%
ER:32	4864.0	100%	59937.3	37%	7859.8	97%	20306.2	30%	4845.0	100%	7605.8	100%
ER:64	6929.5	100%	**90532.0**	23%	9938.9	100%	27940.8	57%	5508.8	100%	7237.2	100%
MR:04	1989.6	100%	34464.0	17%	8733.8	87%	7396.3	10%	4476.8	100%	4991.3	100%
MR:08	2061.0	100%	**21556.0**	3%	6144.3	87%	10710.8	13%	4209.4	100%	5242.7	100%
MR:16	2580.5	100%	55900.0	7%	5927.5	93%	11832.7	20%	5080.4	100%	7217.4	100%
MR:32	3500.7	100%	57776.4	17%	6798.8	90%	20544.8	17%	5806.5	100%	6518.3	100%
MR:64	5474.4	100%	66173.6	17%	8530.1	100%	23983.4	33%	5537.6	100%	7346.2	100%
Full:04	1331.5	100%	59116.4	23%	3032.3	60%	**4722.5**	7%	3368.5	100%	6442.9	100%
Full:08	1319.0	100%	72960.7	10%	**2849.1**	67%	5192.0	3%	3257.3	100%	6617.2	100%
Full:16	**1318.1**	93%	61342.5	13%	3643.7	60%	10672.0	3%	2624.2	100%	**4081.3**	100%
Full:32	1540.7	100%	69052.7	10%	2965.6	70%	6205.0	3%	**2553.8**	100%	5668.1	97%
Full:64	1662.0	100%	47324.7	20%	3296.1	77%	-.-	0%	5569.1	93%	5296.0	90%

Table 6.31: Delayed juvenile replacement summary results for binary value landscapes

does not perform well on this domain. As already noted however, the delayed replacement model has little influence on the outcomes of searches using full graphs.

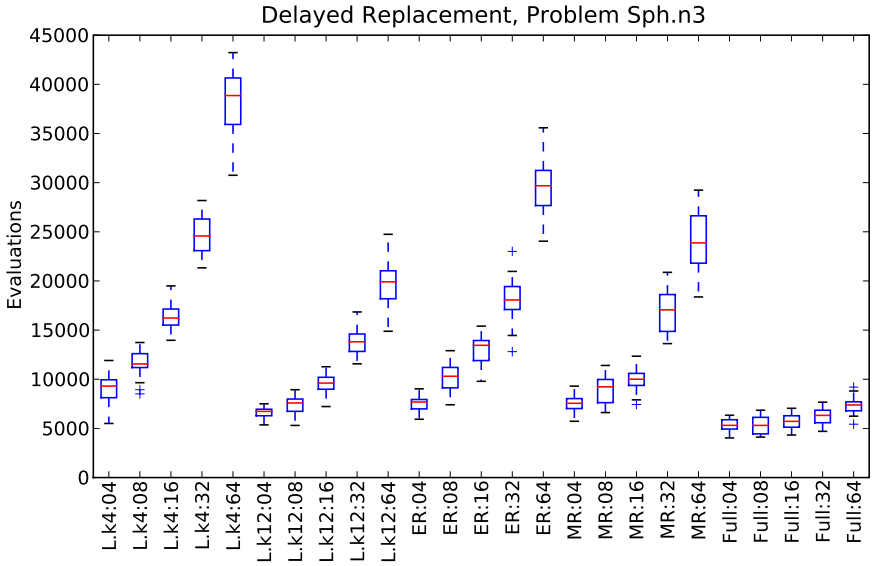


Figure 6.39: Delayed replacement comparison for the Sph.n3 domain.

For simple domains like the SUS.100, SUS.1000 the influence of delayed replacement is a slight increase in the overall spread of ES distribution as the delay rate increases.

The strong results of this delayed replacement model support the consideration of ecological models and principles to see if they can assist evolutionary search models. Clearly there are niche applications where the additional complexity of such a model is of value, and domains where the value is not realised cost effectively.

6.3.7 Rewired Lattices

In natural complex systems it has been observed that many graphs contain strong localised connections, and a sparse but effective number of long distance connections. The WS small-world model recreated such properties using a base lattice topology and varying degrees of rewiring. As previous results in this chapter have shown, the WS models perform well on a range of problem domains with respect to both evaluations to success and success rate results. However the range of rewiring and WS models has been limited so far.

Within the topology survey (Appendix C), the influence of rewiring on lattice properties is considered in detail. Figure 6.41 shows the influence of rewiring on global and local efficiency measures for three standard topologies. As the degree of rewiring is applied, the global efficiency (as measure of mean path length characteristics) increases in each case. Similarly the measure of local efficiency is eventually disrupted by the removal (and reallocation) of local connections. Some complex network models simply add new connections rather than removing and reallocating old ones, and thus preserving the local characteristics at the expense of additional connections.

Based on the results within the topology survey review, and on the success of the WS

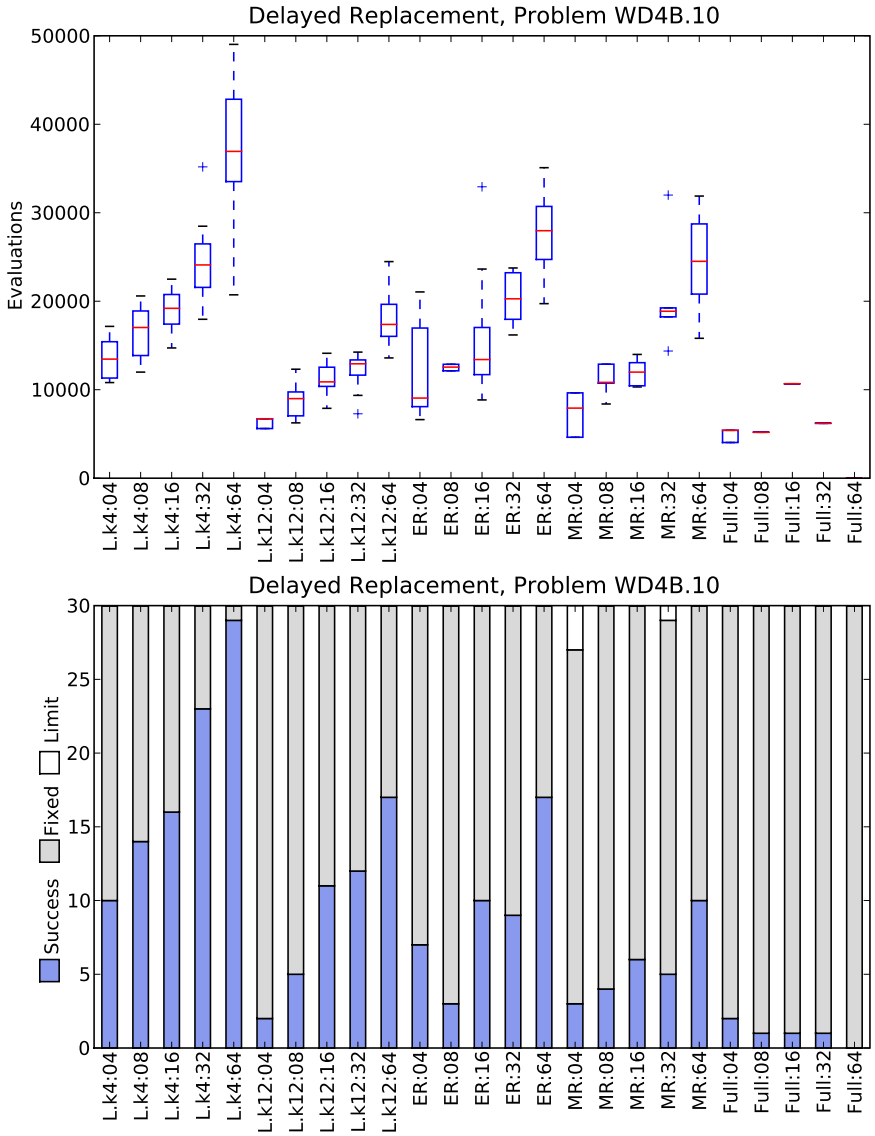
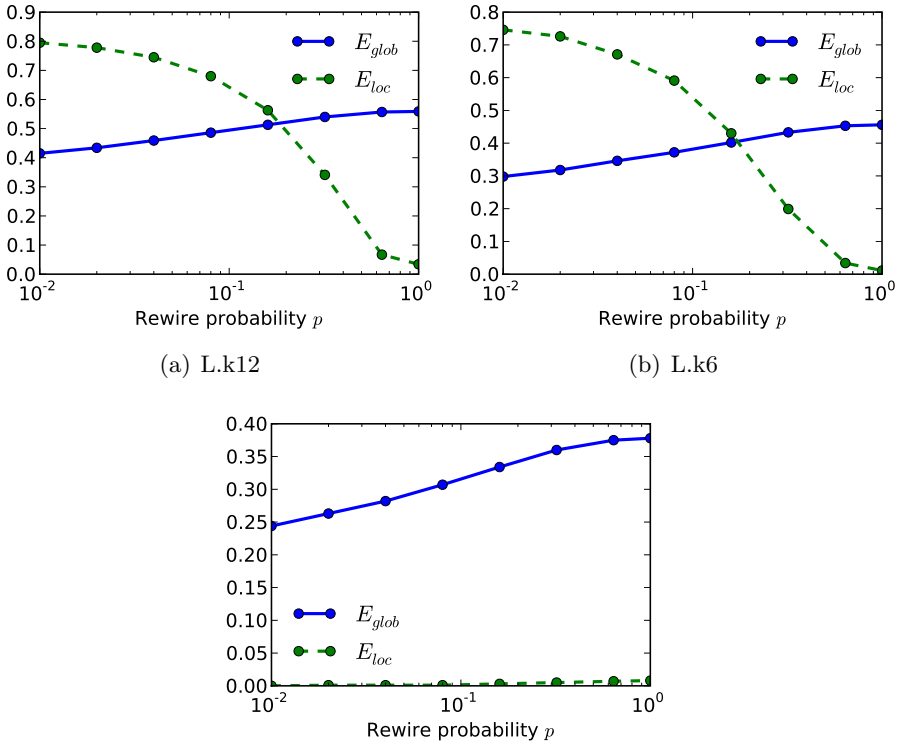


Figure 6.40: Delayed replacement comparison for the WD4B.10 domain.



(a) L.k12

(b) L.k6

(c) L.k4

Figure 6.41: Comparing the influence of rewiring on E_{glob} and E_{loc}

model in experiments consider thus far, and given the inspiration and motivation from both complex systems and ecology, the influence of rewiring on lattices and evolutionary processes should be considered further.

The standard base experiment lattices are used at a size of $n = 400$ (or as near as possible within constraints) using the same lattice dimensions (20×20) used in the scale investigations. The lattices are ordered by their effective mean degree $\langle k \rangle$ (at $n = 400$) which is different to the arbitrary order presented in previous investigation results: hk3($\langle k \rangle \approx 2.81$), k4($\langle k \rangle = 4$), hk8($\langle k \rangle \approx 5.63$), k6($\langle k \rangle = 6$), k8($\langle k \rangle = 8$), and k12($\langle k \rangle = 12$).

Each base lattice instance is rewired by a probability p from the following selection of values (as a subset of an exponential sequence): $(0.0, 0.01, 0.04, 0.16, 0.64)$. The standard problem landscapes are used, excluding domains that have previously presented little differential results. Specifically Sph.n3i, Ros.n2i, MTTP.20i, WD4B.5, SUS.100 and SUS.1000e are not included.

Table 6.32 presents the summary result table including both real and binary landscapes. The ordering of lattices by mean degree clearly shows that minimum and maximum AES results tend to occur at the extreme degrees.

Figure 6.42 shows how for a simple Sph.n3i (low dimensions, and non-deceptive) domain the overall ES mean trend (AES) is to decrease in line with base lattice mean degree; the denser the underlying graph the less evaluations are required to achieve a successful

	Sph.n3i		Sph.n20i		Sph.n20		Ros.n20i		FMSi		MTTP.100i		WD4B.10		SUS.1000	
hk3.00	**8538.2**	100%	89823.2	100%	**165162.8**	100%	-.-	0%	73600.7	14%	**169481.6**	74%	**45097.9**	100%	7028.1	100%
hk3.01	8368.1	100%	**90887.1**	100%	165098.4	96%	-.-	0%	86676.0	8%	167556.2	76%	44526.1	100%	7052.0	100%
hk3.04	8353.6	100%	87458.4	100%	159048.3	92%	-.-	0%	69893.0	6%	157013.9	76%	38645.5	100%	**6833.2**	100%
hk3.16	8387.5	100%	85549.1	100%	155784.7	84%	-.-	0%	88677.0	6%	143735.7	70%	37838.4	98%	7953.0	100%
hk3.64	8080.2	100%	84389.2	100%	155627.5	66%	-.-	0%	75224.0	6%	133279.5	66%	35950.8	92%	7813.8	100%
k4.00	7494.8	100%	80657.0	100%	147754.4	100%	-.-	0%	88932.5	8%	141936.4	74%	36985.9	98%	7116.6	100%
k4.01	7629.2	100%	79545.7	100%	145198.5	98%	-.-	0%	82244.8	10%	130673.7	82%	35392.5	98%	7427.8	100%
k4.04	7600.0	100%	76859.5	100%	141832.2	78%	-.-	0%	58129.7	6%	118703.9	76%	31349.9	96%	6869.1	100%
k4.16	7264.9	100%	73605.8	100%	136730.0	72%	-.-	0%	76259.0	4%	111622.7	76%	29699.2	90%	6977.0	100%
k4.64	7202.7	100%	71791.3	100%	132885.7	62%	-.-	0%	80939.0	2%	112554.4	82%	31265.7	96%	7802.0	100%
hk8.00	7234.2	100%	78113.0	100%	144269.7	94%	-.-	0%	74994.6	10%	135480.8	78%	34407.0	96%	9161.1	100%
hk8.01	7260.1	100%	78400.3	100%	142316.0	92%	-.-	0%	**96823.0**	4%	135084.3	76%	33975.9	94%	8540.5	100%
hk8.04	7128.3	100%	74911.4	100%	137900.0	86%	-.-	0%	**37665.5**	4%	125982.8	84%	30924.9	94%	8097.3	100%
hk8.16	6917.6	100%	71315.2	100%	130389.0	70%	-.-	0%	52338.0	6%	109152.7	82%	28641.9	78%	7384.8	100%
hk8.64	6663.3	100%	67994.2	100%	127231.8	64%	-.-	0%	49064.0	2%	112718.1	68%	25441.2	70%	7198.2	100%
k6.00	7317.8	100%	76317.3	100%	139212.6	100%	-.-	0%	63257.0	6%	138676.7	68%	32644.2	98%	7463.5	100%
k6.01	7218.8	100%	75581.2	100%	137499.1	94%	-.-	0%	82793.0	2%	127131.1	74%	32520.1	96%	7187.6	100%
k6.04	6925.7	100%	72387.8	100%	135536.4	86%	-.-	0%	68529.3	6%	111933.2	70%	28887.0	100%	7975.7	100%
k6.16	6594.0	100%	68380.8	100%	127279.8	72%	-.-	0%	55945.3	6%	115695.0	78%	25898.0	96%	7738.7	100%
k6.64	6805.0	100%	66398.1	100%	123882.1	62%	-.-	0%	50639.0	8%	99679.9	76%	23426.0	90%	7179.5	100%
k8.00	7009.8	100%	71679.3	100%	130951.9	84%	-.-	0%	-.-	0%	114874.1	58%	29085.4	88%	8122.3	100%
k8.01	6793.7	100%	70933.3	100%	133446.8	90%	-.-	0%	64375.6	18%	110332.9	72%	29001.7	98%	7909.5	100%
k8.04	6889.5	100%	68593.9	100%	128822.5	80%	-.-	0%	60661.0	10%	109348.9	68%	27638.2	86%	6953.2	100%
k8.16	6710.4	100%	65865.5	100%	121738.8	62%	-.-	0%	40025.0	8%	100700.4	80%	24709.1	90%	8076.4	100%
k8.64	6415.3	100%	64339.3	100%	117992.5	44%	-.-	0%	43776.3	6%	97131.9	90%	22323.0	78%	7747.8	100%
k12.00	6618.7	100%	66892.1	100%	123801.4	82%	-.-	0%	87207.0	2%	98806.9	82%	25258.5	90%	7842.8	100%
k12.01	6558.3	100%	66427.7	100%	122533.9	82%	-.-	0%	57824.0	10%	112775.4	82%	26013.7	94%	**9258.7**	100%
k12.04	**6312.4**	100%	65138.4	100%	120337.7	68%	-.-	0%	49930.0	4%	94699.8	72%	24614.9	80%	8845.7	100%
k12.16	6374.7	100%	62136.5	100%	115349.7	56%	-.-	0%	-.-	0%	**81857.5**	60%	23647.6	86%	8463.2	100%
k12.64	6376.4	100%	**61500.4**	100%	**113055.7**	42%	-.-	0%	40562.0	4%	96721.0	76%	**21225.2**	76%	7553.0	100%

Table 6.32: Rewired lattice summary results for real and binary landscapes When SR is 0% the AES is shown as "-.-".

outcome. The influence of rewiring in this set of results is not a strong artefact of the data presented.

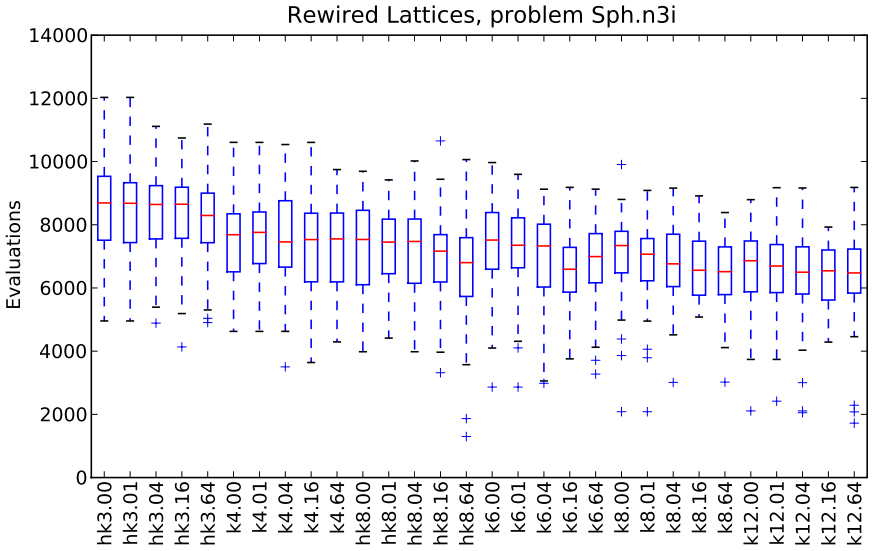


Figure 6.42: Rewired lattice performance comparison for the Sph.n3i domain

For the increased difficulty of the 20 dimensional Sph.n20i domain (non-deceptive) the underlying lattice degree trend remains (Figure 6.43), while the additional rewiring degree also reduces the AES mean and ES distribution values. Each topology type subgroup (for example the k4 group from k4.00 to k4.64) the influence of rewiring is clear and significant in AES distribution changes.

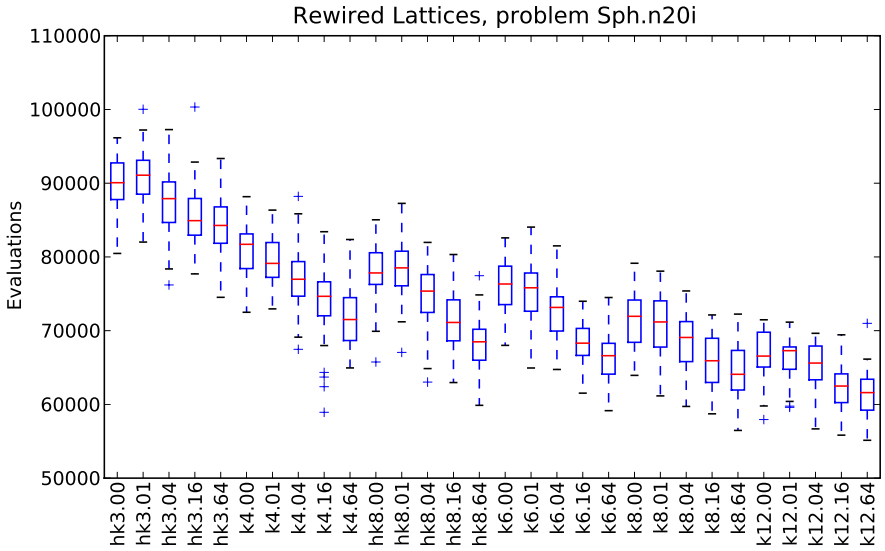


Figure 6.43: Rewired lattice performance comparison for the Sph.n20i domain

In the case of the deceptive Sph.n20 domain the overall success rates are reduced and the influence of rewiring on the success results can be observed. Figure 6.44 shows the success plot results and it is clear that not only does rewiring reduce the ES and AES values, but also critically impacts on the success rate. With increased levels of mixing

that result from the increased rewiring level, global efficiency increases and yet for this particular domain, it is a negative influence for the overall evolutionary search process. Rewiring encourages premature convergence of the population.

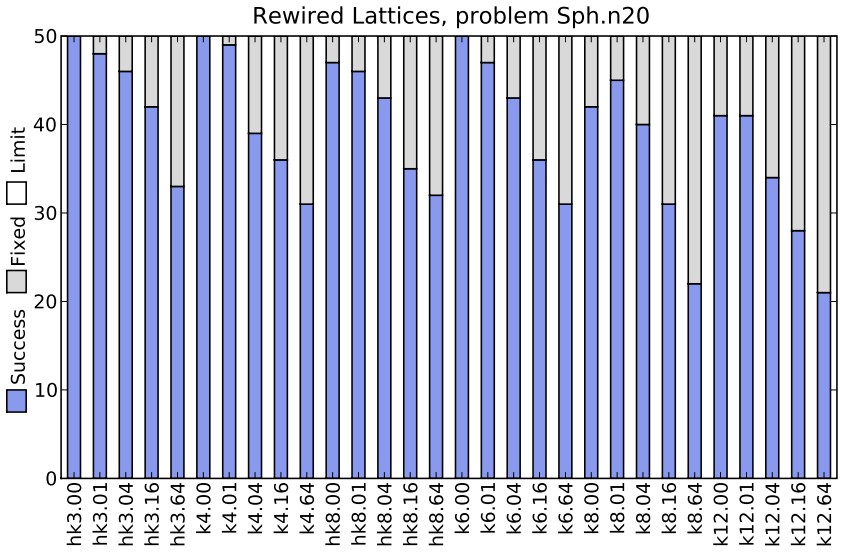


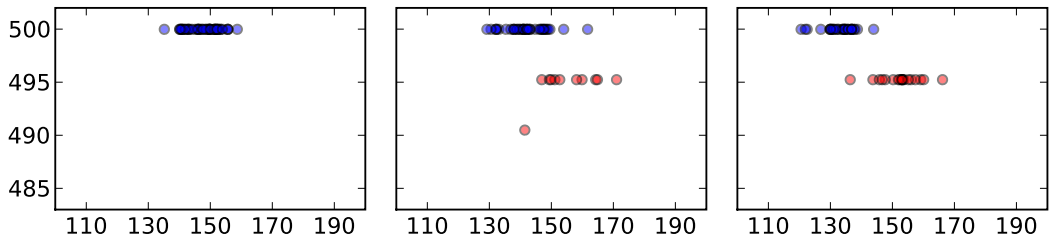
Figure 6.44: Rewired lattice success rate performance comparison for the Sph.n20 domain

As an enhancement to this observation of rewiring influence on result distribution, scatter plots of three different rewiring levels for the L.k4, L.k6 and L.k12 lattices for the Sph.n20 domain are shown in Figure 6.45. The deceptive nature of the search domain is shown by the horizontal banding of results. There is a clear decrease in success rate (blue circle) results as both the rewiring levels are increased (from left to right panels) and as the base lattice degree is increased (from L.k4 in (a) to L.k12 in(c)).

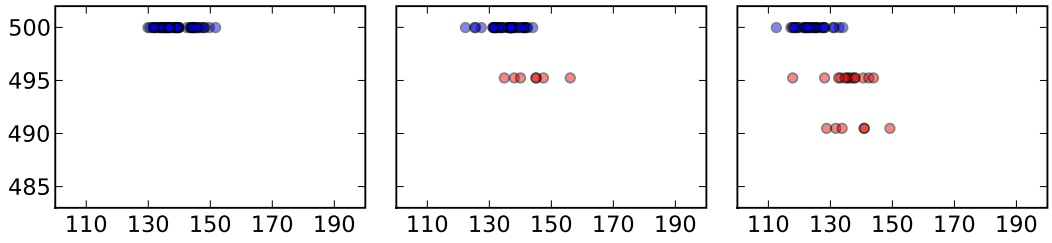
Figure 6.46 shows the rewired success ratio for the MTTP.100i domain, where influence of rewiring is less consistent across lattice degree groups, but still increases the observed ratio of fixed results. This is again shown in a series of scatter plots (Figure 6.47) where the distribution of success (blue), fixed (red) and limit (white) circles change in response to the topology created by rewiring.

The difficult and epistatic FSMi domain has few success results across all topology instances (Figure 6.48) and presents a large sample of fixed results. Within the scatter plot distribution of fixed results shown in Figure 6.49 we can see not only a clustering of suboptimal solutions separated at a distance from the ideal zero value, but also numerous "near" results that are also fixed. It seems that not only does the domain have deceptive values away from the optimal, but also near the best values as well.

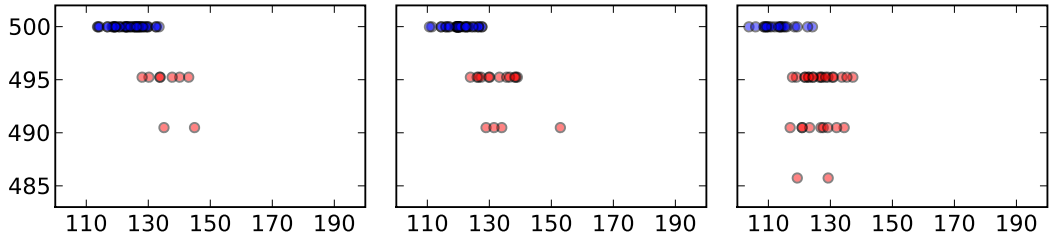
It can also be seen that successful results often occur early within the number of evaluations; the loss of diversity within an older population makes it less likely to find success results. Based on this, one targeted strategy for the domain would be a "multiple restart" approach of many short runs instead of one long run; this should collect more overall success results for an equivalent number of evaluations.



(a) L.k4



(b) L.k6



(c) L.k12

Figure 6.45: Rewired lattice success results scatter for the Sph.n20 domain. In each panel the horizontal axes represent the number of evaluations and the vertical axes the solution fitness. Each circle represent a single run result. Blue indicates a success, red a fixed result, and white a limit result (if any). The circles are transparent so that the density of points is also apparent. The three panels in each group represent rewiring levels of $p = 0.0$, $p = 0.04$ and $p = 0.64$ respectively.

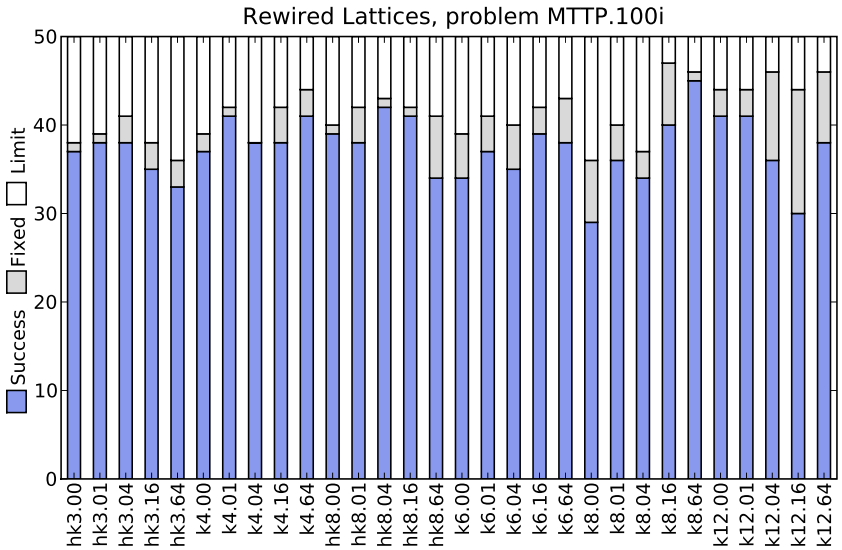


Figure 6.46: Rewired lattice success rate performance comparison for the MTTP.100i domain

6.4 Discussion

6.4.1 Outcomes

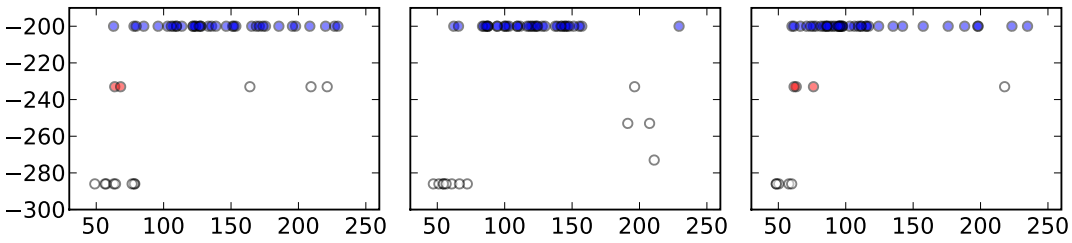
Perhaps the strongest overall outcome of the work presented in this chapter is simply that specific topological properties can and do influence evolutionary progress and outcomes. Results are sensitive to the application domain and the nature of the search space. Other factors such as selection pressure can dominate result outcomes. There are many examples where a varied topology property has a specific, correlated and often significant impact.

It was observed that as topology scale increases, the typical distribution of evaluations to success also increases. Also, as scale increases initial genetic diversity is increased, which can subsequently postpone premature convergence; this positively impacts success ratio results in many examples where a small population size struggles to find good solutions. The performance influence of scale on different topologies varies.

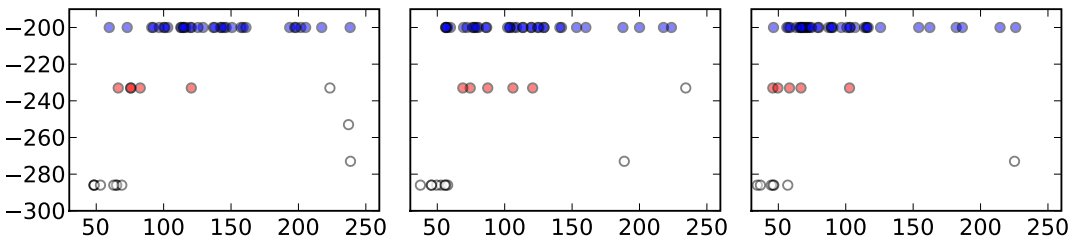
Circular and bound lattice performance was compared. The change in topological properties does translate to search performance influence, where the larger range of mean path length in non-circular lattices increases the mixing (takeover) time of a population, and so increases the evaluation time to success as well as increasing success rate for some domains.

Different lattice and structure based update orders were considered, as well as fitness based update order. Although results show that such processes can influence outcomes, their influence (on the base configuration) was minimal. A simple strong mate selection operator, by comparison, has a much stronger influence, with classic speed (AES) and quality (SR) tradeoffs.

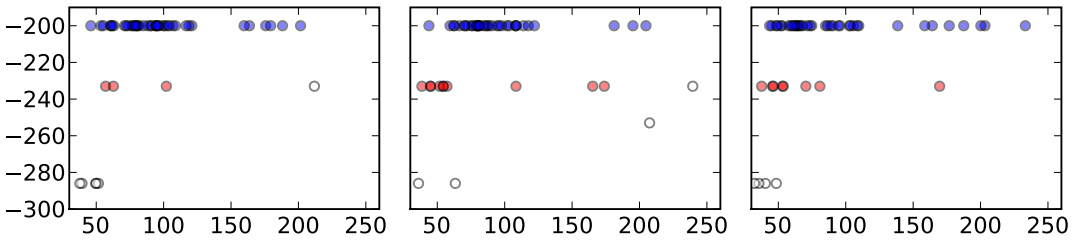
A delayed replacement model is a simple implementation of juvenile (age structured) population interactions. Strong results clearly showed that delayed replacement increased evaluation time to success and improved success rate on all topology types, but particularly for sparse structured lattices. Full graph models were the least influenced by delayed



(a) L.k4



(b) L.k6



(c) L.k12

Figure 6.47: Rewired lattice success result scatter for the MTTP.100i domain. In each panel the horizontal axes represent the number of evaluations and the vertical axes the solution fitness. Each circle represent a single run result. Blue indicates a success, red a fixed result, and white a limit result (if any). The circles are transparent so that the density of points is also apparent. The three panels in each group represent rewiring levels of $p = 0.0$, $p = 0.04$ and $p = 0.64$ respectively.

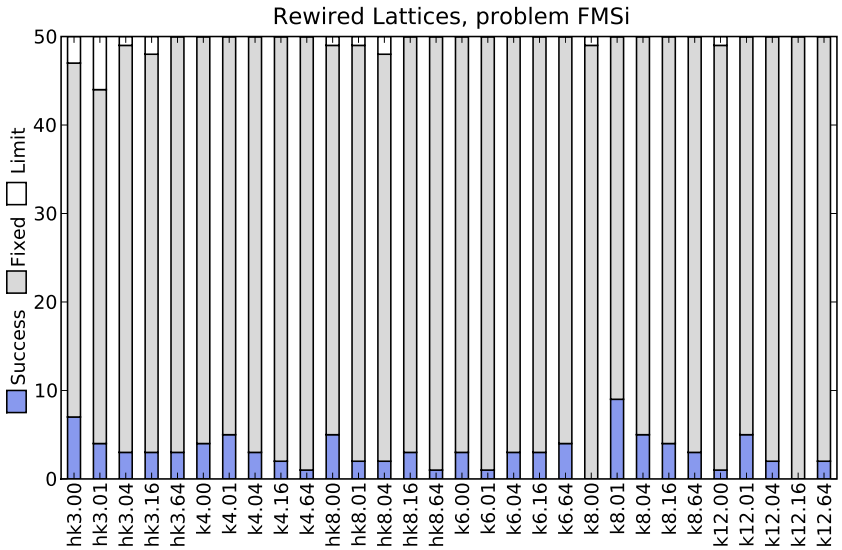


Figure 6.48: Rewired lattice success rate performance comparison for the FMSi domain

replacement. As suggested in earlier discussions, this result certainly encourages further consideration of ecological principles and models within evolutionary search.

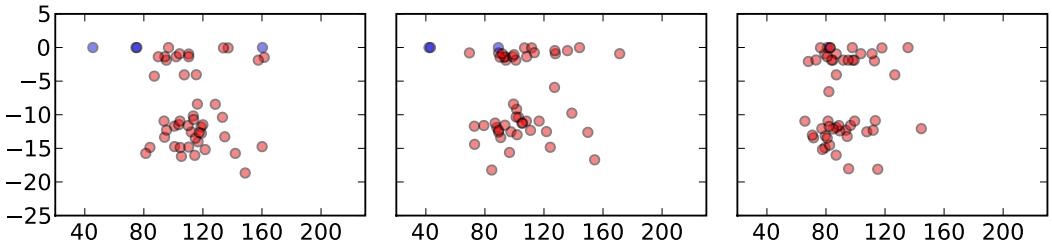
Lastly, the influence of rewired lattice topology on evolutionary performance was considered. A range of lattices, ordered by mean degree, showed a base-line trend that as the mean degree of the lattice increased the ES performance decreased, mixing increased and success rate decreased. With the addition of rewired topology, the ES trend is extended; as rewiring levels increase, ES performance decreases and success rates drop. The shape of the drop-off rate profile matches observed increases in global efficiency of lattices due to rewiring. This suggests that success rates, and a graphs' global efficiency, are strongly connected. Also, as rewiring levels increase, global mixing levels are increasing with the repercussion of greater levels of rapid premature convergence.

6.4.2 Future Opportunities

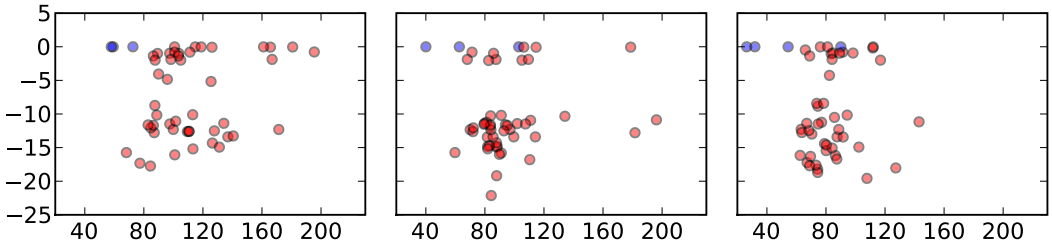
Numerous avenues exist for further research, many of which could easily be considered by extending the existing body of work presented in this chapter. For example, graph edges can be specified as directed and the influence on search considered with different overall topologies. For equivalent comparison two directed edges should be allocated for the equivalent bi-directional edges used to date.

Another interesting and strong ecology-based concept is the influence of occupancy pressure. If locations of the population topology can be unoccupied, then the influence of progressive colonisation and migration within topology can be considered, as well as the potential for interesting behaviour during different phases of system level occupation.

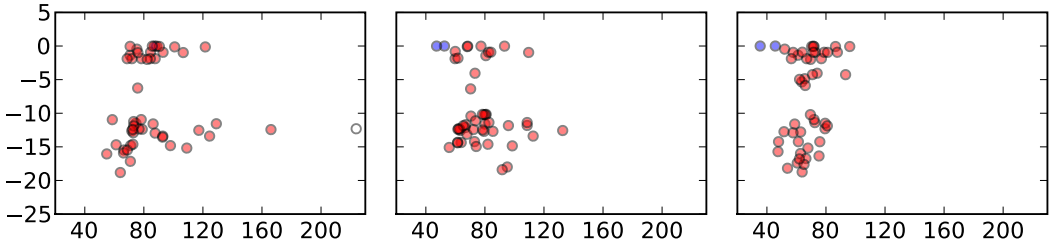
Disturbances such as extinction events, dynamic population topology and topology (environment) based gradient influence are all natural ecological extension that can be considered. Kirley and colleagues [199, 194, 195, 196] have considered several "Ecological Algorithm" approaches using population level ecology ideas (such as disturbances



(a) L.k4



(b) L.k6



(c) L.k12

Figure 6.49: Rewired lattice success result scatter for the FMSi domain. In each panel the horizontal axes represent the number of evaluations and the vertical axes the solution fitness. Each circle represent a single run result. Blue indicates a success, red a fixed result, and white a limit result (if any). The circles are transparent so that the density of points is also apparent. The three panels in each group represent rewiring levels of $p = 0.0$, $p = 0.04$ and $p = 0.64$ respectively.

and gradients), and their work can be replicated by the ESEC model and extended to consider additional variations. The body of work by Kirley and colleagues provides an excellent point of reference in regard to ecology based population topology ideas, as well as multi-species and meta-population models which related directly to the community and ecosystem organisations of the ESEC model.

As mentioned in the selection of problem landscapes, the base configuration for experiments specified the use of binary genomes. An experiment was conducted to confirm that the influence of topology was also present when using real valued genomes. The results were not presented in detail in this chapter, but are included as part of the CDROM appendices. However, the decision to use binary genomes does limit the generality of the results, and many of the investigations using real value landscapes could be repeated using specific real valued genomes with appropriate crossover and mutation operators. An extended investigation in this area could also try to determine the sensitivity of topology with respect to other parameter influences.

6.4.3 Complex Topology and Computational Cost

The addition of specific, and possibly complex, topology as part of an EA instance can add additional computational complexity. In the experiments considered here, topology structure is constant during each evolutionary run, and so the impact of the additional topology complexity is limited to initialisation, and a small lookup overhead to those operations that are based on topology. In the implementation of `esec`, graph related functionality is supported by an external compiled library which is well tested and optimised. As a result, the practical impact of additional topology related complexity is very small in comparison to other algorithm routines implemented in Python.

If topology were to be altered during an evolutionary run, as in adaptive parameter control experiments, this would impose an additional (but typically localised) computational cost. This type of impact could be reduced by reducing the frequency of topology changes to an episodic (epoch) model rather than a continuous one. Similarly, if topology is distributed to take advantage of parallel processing, the cost of topology change can also be distributed and parallelised.

6.4.4 Topology Selection Guidelines

From the results presented in this chapter, and results presented by other authors, there are several insights and recommendations regarding the use of different population structures that can be suggested to assist researchers and practitioners.

For all topology types, the issue of population size is influential to overall success and search time. If the population size is too small the search is more likely to be deceived, while a population that is too big takes more computational steps to achieve similar success rates. Finding adequate population size is a difficult problem for new domains, and perhaps best derived using parameter control approaches.

If we consider a simple problem domain that does not contain deceptive features, a simple panmictic population is all that is needed or justified. For example, if an EA using

a panmictic topology is applied to a new problem domain and consistently finds a good solution, there is no motivation for additional population complexity. Alternatively, if the EA search is regularly deceived, a different population topology can be selected to see if an alternative is useful. There may also be practical application advantages to using other population topologies, such as the advantage of distributed processing, but this is not considered in detail here.

Regular lattice topologies are well known to support specialisation and diversity through localised operators. An undesirable consequence is that EAs using lattice structures typically take longer to converge. This is noted by measures of increased trait takeover time in cellular EAs. This result correlates directly to the increased mean path length profile of lattice topology with respect to full graphs. One suggestion, then, is that when a simple population topology is regularly (or suspected of being) deceived by a new unknown domain, a lattice topology should be tried instead. If success rate performance improves, it can indicate that the domain is deceptive and niche support is a useful feature. As notes, lattice populations can take longer to succeed in comparison to a full graphs, however the increased success rate can justify the decision.

While circular lattices are a homogenous structure, within a bound lattice there is a greater range of path lengths and longer paths. As a result, circular lattices support a wider range of niche structures with differing levels of isolation. If a bound lattice provides better results than those of a circular lattice, it may indicate that the overall size of niches in a circular lattice is not optimal, or that a range of different niche sizes is a better support structure for successful evolution.

Mean edge degree (the density of edges in a graph) affects convergence time and success rates. Higher edge degree lowers mean path length and so supports rapid convergence. While more local links mean better local efficiencies, and as a result greater local fitness selection pressure (competition), it also essentially creates less capacity for specialisations. Also, if local clique subgraph topologies overlap it reduces the mean path length of the topology to a greater amount. From this it can be suggested that increasing the mean degree of a lattice is another parameter that can be adjusted for an EA, with low values encouraging niches and specialisation, and high values decreasing convergence time but also reducing niche support and make the population more susceptible to deception features of a new problem domain.

One strategy for deceptive domains is to combine the niche supporting features of a lattice population with other approaches, such as local search optimisation or distributed island EA models.

Another topology model explored in this chapter was the use of rewiring on lattices. When a regular lattice is randomly rewired, the mean path length values decrease while regular local niche structure remain until rewiring levels become too destructive. By specifying a lattice of this form, an EA search can take advantage of niche diversity support, while still enabling better overall convergence rates. It was shown in the work of this chapter that rewiring level creates another tuneable EA parameter. Low levels of rewiring improve convergence, especially for simple non-deceptive domains. High levels

of rewiring can destroy the usefulness of lattice niche support such that the a search population is deceived in a deceptive domains.

Although there are some clear recommendations regarding population topology that can be applied to new domains, EAs contain many other influential parameters. Population topology should be selected, tried and adapted using a controlled processes to find an appropriate structure that supports other algorithm operators.

6.5 Closing

The work presented in this chapter shows that topology can and will influence evolutionary search outcomes, moderated by the complexity and deceptiveness of the search domain an EA is being applied to, as well as the form of representation used for individuals and specific operator details.

Overall topology form and scale, as well as other topology based factors such as circular and bound lattice configuration, update order within structured populations, delayed replacement models and the influence of rewiring, all present representative examples of the topology based influence.

Chapter 7 continues the work of this chapter by exploring the capability of the ESEC model using community and ecosystem organisation models, and in particular considers the opportunities for further research.

Chapter 7

Open Research and the ESEC Model

7.1 Introduction

The objective of this chapter is to describe how the design of the ESEC model, and the `esec` package implementation, lend themselves to further investigations in open areas of EC research.

In Chapter 6 the ecosystem notion of simple and complex population topology for single species populations was explored, using an ecosystem framework for evolutionary computation presented in Chapter 5 and implemented as the `esec` Python package. The varying of a single species population organisation highlighted how specific topological properties can influence evolutionary adaption and search outcomes. Chapter 6 represents the main research investigations and results of the thesis.

This chapter presents and considers in detail the two additional organisational structures that the ESEC model includes: *community* and *ecosystem*. Supporting concepts are first considered in Section 7.2 and Section 7.3, and then `esec` package configuration details are outlined. Several reference configuration examples are presented for both community and ecosystem organisations, followed by a discussion of many interesting and open research questions.

7.2 Community

7.2.1 Subpopulations

As noted in Chapter 2, in natural systems coevolution is by far the dominant example of evolution; it is rare to find an environment that contains an isolated population of a single species. Niches, speciation and interaction between species create important dynamics for diverse and robust adaptation in natural systems.

The ESEC model extends a single species population organisation to that of a community organisation containing multiple interacting species. Single species populations are

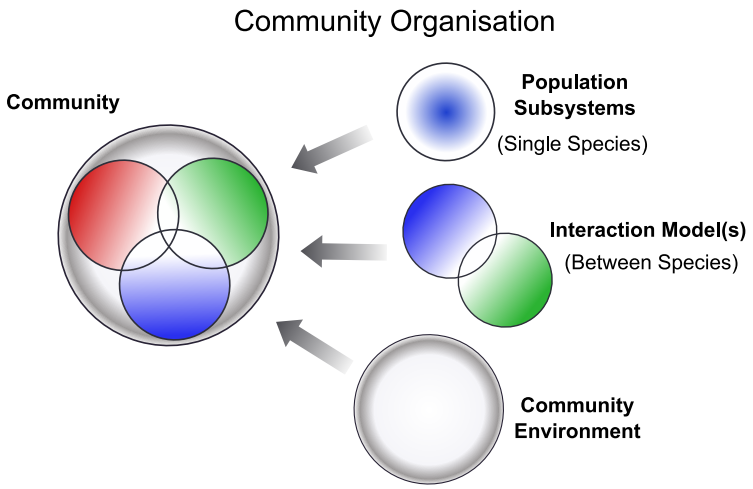


Figure 7.1: Community organisation composition within the ESEC framework. The role of the community is to define and contain a collection of species populations and interspecies interactions.

included (as presented in Chapter 6), with an additional topology of interaction between species individuals.

7.2.2 Multiple Species

The term "community", as a part of ecology vocabulary, represents a group (or "assemblage") of interdependent organisms that inhabit a common location and interact with each other.[1] When the characteristics of species evolve (adapt) together it is an example of coevolution. Thus, a coevolution model composed of multiple species fits within the "community" organisation of the ESEC framework.

Problem landscapes with multiple parameters can easily be applied to a cooperative coevolution community model by creating separate species and populations for each of the problem parameters. Evaluation of a potential solution for the landscape requires that a set of individuals from each species be selected. Such a simple approach is not generally applicable. Many problem domains contain variables and parameter interactions that are difficult to separate, making an external "hard-coded" definition of each species a limited approach.

Similarly, search methods for some problem landscapes benefit from an incremental development of solution complexity which is possible using a competitive coevolutionary community model. Different species interact competitively creating a dynamic fitness evaluation; the fitness of one individual is its ability to successfully defeat other species individuals.

A community model does not need to be limited to exclusive cooperative or competitive models. Biological examples show that interactions between species can have a complex mixture of helpful, harmful and neutral influences.

[1] In a more basic sense community may be defined simply by a common or shared attribute such as an individuals' location or goals.

7.2.3 Interaction Models

As listed and described in Section 2.3.5 of Chapter 2, there are three distinct models of interaction within a community of individuals:

- Symbiosis: Interaction between individuals of two or more species which benefits at least one. There are three kinds of symbiosis: commensalism, mutualism and parasitism.
- Predation: Predator-prey relationships where predator individuals remove prey individuals from the community.
- Competition: Two or more individuals, of the same (intraspecific) or different (interspecific) species, competing for resources.

Competition within a single species population is embodied within classic EC models as one or more forms of selection pressure based on fitness measures. For example, fitness based parent selection, fitness based fertility, and the replacement of old or low fitness individuals. Single species models have been considered in detail in Chapter 3 and Chapter 6.

Interspecific competition between different species opens a range of interaction processes and variables. As discussed in Chapter 2, the competitive exclusion principle states that two or more species with similar environment requirements will not coexist indefinitely, and as a consequence all but one species will be excluded.

In natural systems predator-prey relationships are typically between different species, although cannibalistic behaviour within a single species does occur. For example an adult individual might prey on juvenile members of the population, a parent individual might consume a mate after reproduction, and rivalry between siblings can result in fatalities.[2]

Predator-prey systems often exhibit interesting interaction dynamics such as seasonal cyclic population growth of prey, and subsequently predator populations. There are also examples of correlated adaptation of traits between predator and prey species in a so called "arms-race" of successional changes. For example, when a prey species acquires a useful trait (such as speed) that allows it to avoid predators, the predator species is then pressured to acquire a matching trait (such as speed or alternative stealth abilities).

Symbiosis provides the model of mutualism where multiple species cooperate to overcome the challenges of their environment. In nature the ultimate expression of mutually beneficial relationships is endosymbiosis where two species are so interdependent that they are linked and cannot exist without each other.

Symbiotic evolution has been successfully applied and reported in EC literature, particularly as cooperative coadaptation or cooperative coevolution. The fundamental concept is that the fitness of individuals from one species are defined in relation to one or more

[2]Spiders provide many supportive examples. In Australia the female Redback spider (*Latrodectus hasselti*) is known to consume males during and after mating. For many spider species, young spiders have been observed consuming their siblings immediately after hatching.

other species. The term "cooperation" suggests that all species have a positive contribution (mutualism) or influence. In general symbiosis does not require mutual benefit to all species involved.

7.2.4 Interaction Structure

A challenge for any EC model supporting interacting species is how to define which specific individuals are to be involved in collaborations, and how any measure of interaction-based performance (the raw fitness value) should be credited to contributing species. An exhaustive evaluation of all individuals of a species with all individuals of other species is combinatorially expensive. The concern of selecting an appropriate subset of individuals to evaluate together is well known as the "pairing problem".

In natural systems interaction between species is localised by the environment and mobility of individuals, and from a computational perspective a small yet adequate sample of pairings is certainly preferred. The selection of sample individuals can be arbitrary (by a structured population or through random selection), or in a more "directed" (and greedy) approach a representative individual (such as "best") can be selected for each species.

The structured (cellular) population approach to pairing is a celebrated EC cohabitation model [98]. Lattice topologies have a well established ability to support localised variation[3] within a population. Collaboration between two or more species is done in a localised way defined by cohabitation of individuals within a lattice. The number of combined evaluations is reduced and coadaptation can occur.

Populations of different species groups can have many different interaction structures. Consider the following interaction models and example scenarios:

- **1-to-1**: A symbiotic bacterial organism living in the gut of a large mammal has a limited "1-to-1" interaction with a single host animal from the species population.
- **1-to-many**: A single keystone predator with a great range of mobility will have a "1-to-many" influence on numerous prey species in its hunting area.
- **many-to-many**: Small prey compete with other similar species in complex "many-to-many" interaction models.

In all cases the connection between one species and another is a topology. Although simple rules can be used to define the interaction topology, such as a simple full graph or a basic lattice, other more complex graphs can also be used. Interaction topology can also be dynamic, changing periodically and adapted from one form to another, influencing the rate and outcome of coadaptation.

[3] Also known as "speciation" but in this context the notion of different "subspecies" within a single species population could be confusing.

7.3 Ecosystem

7.3.1 Concepts

The next aspect of the ESEC framework to consider is the ecosystem; an evolutionary system composed of interacting subsystems. The rules or "protocol" for interactions between subsystems define a connection topology, while subsystems individually have the capacity for any specification that is possible for a population or community, as well as support for nested ecosystem models.

An ecosystem model of organisation for evolutionary computation within the ESEC framework, is in many regards the simplest to define. It is essentially a high-level specification for composition. Figure 7.2 shows a conceptual example of the ecosystem organisational model, composed of simple and complex subsystem components.

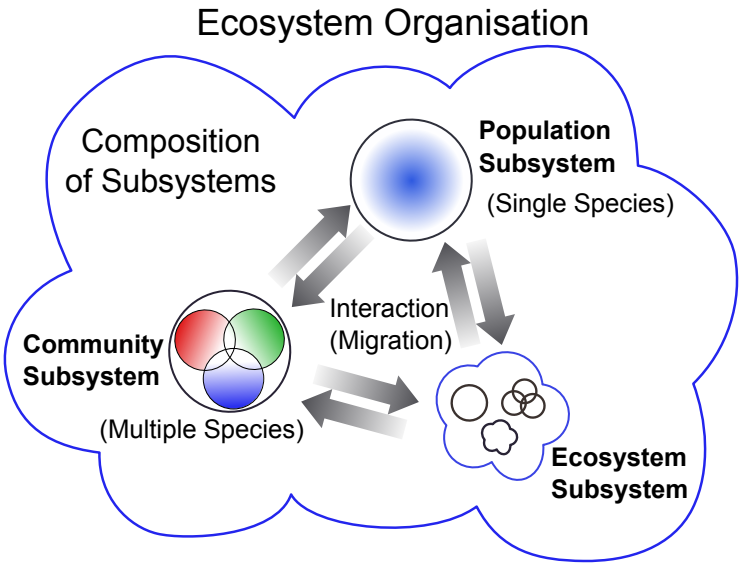


Figure 7.2: Ecosystem organisation model within the ESEC framework. The role of the ecosystem is to define and contain subsystems and interactions.

It is the role of the ecosystem organisation to contain subsystems, and to define how and when (at least initially) the subsystem components *interact* with each other. Subsystems may be unaware of other subsystems or the ecosystem in a direct sense, however they are implicitly "linked" to each other.

In keeping with the concepts of an ecological model, the ecosystem structure can also define the distribution and limits of resources, such that an economic notion of *quotas* for resources can be managed. For example, all subsystems can be allocated a quota of time that limits the amount of adaptation or evolution. Similarly, subsystems may be allocated specific resources which then limit the number of individuals that can be sustained within subsystem populations. As in natural systems, it is possible for such energy and resource limits to be adjusted dynamically, either deterministically or stochastically.

Using the notion of quota limits, an ecosystem can contain different subsystems that "evolve" at different rates. This could allow mitigation of the influence interacting species

have simply through "size" (resource) quotas. For example, the development of species in one subsystem population could be "slowed down" while other subsystems are allowed periods of accelerated adaption or reduced relative fitness competition by increasing the available population size. Similarly, reducing the size of a subsystem is a means of specialising evolution while limiting resource use.

The roll of the ecosystem organisation defined in this manner may be thought of as a "master" in classic master-slave models. However, implementation need not be limited to communication via the ecosystem as a master proxy. Direct links between subsystems can be created based on the specification of the ecosystem. In this way the "initialisation" of the ecosystem (or later, possibly dynamic restructuring of the ecosystem) is used to specify the connections between subsystems, which then – during allocated quotas of activity – directly communicate with other subsystems as local subsystem protocol dictates.

Note that this model, and the notion of energy or resource quotas, are not inherently synchronous or asynchronous. It is entirely possible for an ecosystem model to support a mixture of both. A mixed model may make specific sense in that inter-subsystem communication can be synchronised, while intra-subsystem activities can be parallelised and asynchronous. A mixed synchronisation model introduces a requirement for additional parameters to be specified for communication protocols and the implementation of quota limits.

It should be evident that the ecosystem composition model has unlimited potential for complexity, given that subsystems may each be complex systems or nested compositions. As a general rule, considered in other fields such as design and architecture, additional complexity should only be included in a system when there is a reasonable expectation of benefit. Complexity should not be added simply because a model supports it.

7.3.2 Island Models as Ecosystems

As an immediate example of a traditional EC model that fits simply into the ecosystem framework, consider the classic island EA population model. (See Section 5.4.3 of Chapter 5.)

One common requirement for all "coarse-grain" EC models is the need to clearly define the structure of connections between coarse groups of individuals (demes) and the nature of communication. Common inter-group structures include fully connected panmictic graphs, ring lattices or toroidal grid lattices. The inter-group structure for many applications has been determined by the nature of underlying available hardware supporting parallel processing. A standard communication method is to allow selected individuals (typically based on fitness) to be copied and copies migrated between island groups at fixed generation intervals or "epochs" of relative generation (evolution) time.

The ideal benefit of an island model is that the evolution of independent island populations provides a structured exploration of unique aspects of the search landscape, while within each particular island the pressure of relative competition encouraged exploitation. Is it possible for each island to contain radically different genotypes, each with different relative levels of fitness. A single population model would be unable to support such spe-

cialisation without explicit mechanisms such as crowing or fitness sharing. The migration of individuals between islands supports a mixing of possibly diverse genetic material.

The concept of the island model is an attractive theory, with implicit parallelisation and structured exploration and exploitation. Multimodal problem domains are a motivation for island migration models, and have been applied to such domains with success. Interestingly, there is also evidence that show the model is an effective strategy even for unimodal domains. The negative aspects of the model are the introduction of additional parameters to consider, and no guarantee that subpopulations are in fact exploiting different niche regions of a search landscape.

Subpopulations need not be homogeneous in size or conditions; there may be variation in selection schemes, mutation rates or operators, and so on. This may create an opportunity to support dynamic problem domains and/or niche requirements during different stages of evolution.

As an example, consider one island supporting a large "step-size" range of mutation values which, for a particular problem domain, provides a very useful "raw" material during early stages of evolution. At a later evolutionary stage a different island with a smaller step-size range of mutation values is of greater use in the specialisation (exploitation) of the domain. It would be the role of migration in this example to communicate from the early successful population to the latter specialisation conditions.

It is also possible to apply problem specific enhancements to the island migration model, such as special "seeded" initialisation for some subpopulations. Similarly, it is possible to incorporate the notions of variable subgroup size or occupation. For example, as a start-up process expansion can be used within each subgroup. This creates a phase transition between initially low competitive fitness pressure to that of high pressure as a result of crowding and competition.

Considering subpopulation diversity could lead to expansion or contraction of both subpopulation size or island number. If a particular island group lacks diversity, perhaps its size should be reduced to save computational resources. An entire subpopulation could be removed from the ecosystem if, over time, its contribution is considered negligible. If, however, the diversity of a subpopulation is high, or a means of measuring "species" formation determines the need, the subpopulation could be divided into two or more new replacement subpopulations. Automatic expansion or contraction are difficult and problem domain sensitive features, and yet they are potentially quite useful additions to the model, albeit at the expense of additional parameters and complexity.

7.3.3 Structure and Migration

The island migration model of coarse-grained EC clearly has interesting potential. Eiben and Smith have listed [98] important questions for general island migration models that are adapted and repeated here:

- Subpopulations: How many subpopulations should there be, of what size, and should they be in homogeneous environments?

- Migration topology: Should all subpopulations be connected to all others or should a sparse topology be used?
- Migration frequency: How often should individuals be communicated between islands?
- Migrant group size: How many individuals should be transferred? Is this a fixed size or a dynamic, possibly fitness related, quantity?
- Emigration selection: Which individuals should be selected for exchange? Should a fitness-based measure be used? Should the best always be selected, or should a random sample or biased sample be used instead or as well? Is the method of selection sensitive to the problem domain, or the overall stage of evolution.
- Migrant transfer model: Are migrants moved from one population to another, or are they "copied" to another subpopulation without altering the source subpopulation? Should multiple copies be created to multiply the influence of selected individuals?
- Emigrant integration: Should immigrants compete with their new subpopulation on an equal basis, or should they be allowed special treatment - at least temporarily - to help bias the integration of diverse solution qualities?

If the migration frequency is too high the overhead of communication is similarly high with respect to other computation elements such as fitness evaluation, reproduction operators and so on. Also, a high frequency may result in overall convergence results similar to a single panmictic population model. Alternatively, a low communication frequency may result in isolated and stagnant subpopulations that converge quickly and lead to an overall state of ineffective premature convergence.

The term "epoch" is often used to describe the evolutionary time between migration transfers. Typical epoch frequency values seem to be in the range of $25 - 150$ generations. A simple adaptive strategy, presented by Martin in [19], is to halt individual subpopulation evolution when no improvement has been made for a number of generations (say around 25).

Regarding the size of migrant groups, it seems that many authors have found that in order to limit the result of overall premature convergence it is better to exchange a small number of individuals $(2 - 5)$ as this limits the mixing between each subpopulation. This also suggests that creating multiple copies of good individuals for migration might lead to premature convergence.

Common methods of emigrant selection are "select-the-best" strategies and stochastic fitness-proportional or fitness-rank based selection. Essentially any method used for parent selection or replacement selection can easily be applied to the selection of individuals to migrate. It is also possible to specifically select individuals of a subpopulation to be replaced by new immigrants, to require competitive tournament-based replacement, or to truncate the entire subpopulation.

If individuals are moved between subpopulations rather than copied, then it is possible that emigrant integration is simply a replacement model. For a simple model this requires that the number of emigrants and immigrants balances.

What is clear is that epoch length has a fundamental effect on the influence of migration parameters such as group size, migrant selection and migrant integration policy. For example, if the migration frequency period is low then each subpopulation has a greater tendency to converge, which in effect reduces the influence of selection bias for both migrant selection and integration strategy.

Results have suggested that, provided a practical minimum subpopulation size is respected, more subpopulations are useful when evolution of subpopulations is run in parallel. If there are problem-dependent features that match somewhat to the idea of subpopulations it is possible to select the number and size of subpopulations to match the number of features and their complexity as an act of capacity planning. The initial individuals of a subpopulation might also be "seeded" with known useful qualities so that they will be available for integration.

Each subpopulation island can be exposed to a different fitness landscape, such as a fitness bias to encourage specialisation (exploitation) of unique solution features. Similarly, different subpopulations may be configured to support different evolutionary operators and parameters, such as increased mutation or crossover rates, as another means of encouraging specialisation. Subpopulations are then in competition with each other, and may themselves adapt their strategic role [306].

Injection island models are an example of this, where subpopulations are connected in a hierarchy of levels, and each level is configured for different fitness or operator values. Effectively this can act as a competitive "ladder", where successful solutions evolved at lower levels are migrated up ("injected") to higher levels until the final required complexity is achieved. (For example, see [96].) While this is an elegant metaphor and structure, it again creates a number of new configuration options, many of which are sensitive to specific problem domain features.

A more recent example of large scale distributed evolution is the model of "opportunistic evolution" as proposed by Sullivan, Luke and colleagues. (See [333] and the discussion presented in Section 5.4.3.) Essentially, their island hybrid distributed model strives for efficient use of resources by adaptively evolving subpopulation configurations to make the most of the opportunity and limits created by communication overhead. It is a very effective approach to large scale evolution while practically reducing the multitude of parameters such models create.

7.4 System Configuration

7.4.1 Introduction

The objective of this section is to outline how, at a system level, the community and an ecosystem models of the ESEC framework can be described using the configuration

syntax of the `esec` package. Later sections demonstrate specific community and ecosystem examples using these system configuration details.

7.4.2 Community

For the sake of simplicity, the `esec` package defines two basic models of interspecies interaction within a community: `cooperate` and `compete`. Examples of each interaction policy are presented in later sections. Other complex or domain specific models of interaction can be easily programmed using an event call-back architecture presented by the `esec` package design.

Listing 7.1 shows the essential element syntax of the overall system configuration used for community organisations. As a community is composed of nested populations, the `systems` element contains a list of subsystem configuration dictionaries. The `count` element can either be a simple integer value (specifying the number of instances each system configuration is used to create), or a list of integer values (one for each system configuration) to specify the number of instances that are to be created.

Listing 7.1: Community configuration syntax

```
config = {
    ...
    'system': {
        'type': str, # 'community',
        'systems': list, # of subsystems (populations)
        'count': [int, list], # subsystem instances
        'interact': {
            'policy': str, # ie 'cooperate', 'compete'
            'matchup': [str,dict], # ie 'all', 'best' etc.
            'mapping': dict, #
            'interval': ('on_gen', 'on_gap'),
            'topology?': dict, # if used
            ...
        },
        'quota': [None, list], # quota per subsystem
        'order': str, # ie 'sequence' or 'random'
    },
    'species': list, # configuration details
}
```

The `quota` element value can be simply set as "None", but if it is given a list it must be specified so that it matches either the number of subsystem specifications or the number of instances created (as specified by the count value). In this way the quota limit can be specified for either each group of subsystem instances matching a single configuration, or uniquely for each specific subpopulation instance (regardless of any shared configuration details).

Species configuration details are stored within a list and mapped for use to each nested subsystem. The length of the `species` list must match the length of the `systems` list, and in this way there is a single species configuration (or a reference to another species) for each subsystem.

The `interact` element is a dictionary that specifies the interaction model (`policy`) and the interaction topology used by the community, as well as the `matchup` ("pairing")

model used and the interval (generational or gap). In simple cases the interaction topology can be implied from the `matchup` details.

The `cooperate` interaction policy is suitable for problem domains for which individuals from multiple species are combined to form a solution, analogous to symbiosis. The `compete` interaction policy is applicable to problem domains where a competitive developmental "arms-race" is a valuable means of creating increasing levels of complexity. Seeding the initial complexity of each species' population, and regulating the competitive adaptation rate are important dynamic aspects of competitive models.

Although it is conceivable for interaction between three or more species to involve both cooperation and competition, a general model is not presented here or supported by the default policy models of the `esec` package. Domain specific interaction policies of this form are easily implemented.

Predator-prey models have been applied to EC before, but the strict notion of "consumption" (removal of prey individuals) is not always adhered to. Instead, many EA search models using predator-prey or host-parasite terms are better defined as competitive interaction models. In the predator-prey example presented later in Section 7.5.2, the basic `compete` interaction form is used.

The specification of quota in this model is simple, and it is possible to create dynamic quota models influenced by the relative evolutionary change or contribution to the community. Alternately, population size, or at least the occupancy of the population topology, could be expanded or reduced as a means of altering the resources available to each subsystem.

Interaction `order` determines the sequence that subsystems are allowed to interact and evolve. A simple order is adequate for many communities while a random order can help to reduce unwanted order-based influence.

7.4.3 Ecosystem

Listing 7.2 shows the essential element syntax of the overall system configuration used for ecosystem organisations. It is the same as the community syntax, with the exception of `interact` details which critically describe how individuals from each subsystem will interact. As with the community model, the search domain landscape can be mapped (via interact settings) to multiple subsystem individuals (as in cooperative and competitive models) at the expense of an additional level of interfacing.

Listing 7.2: Ecosystem configuration syntax

```
config = {
    ...
    'system': {
        'type': str, # 'ecosystem',
        'systems': list, # of subsystems
        'count': [int, list], # subsystem instances
        'interact': {
            'policy': str, # ie 'island', 'migrate'
            'interval': ('on_gen', 'on_gap','on_quota'),
            # Other details needed depend on policy value
            'topology?': [str, dict], # if used
            'size?': int, # of the migrant group
            'dest?': str, # ie. 'random', 'graph', 'order'
            'out_selection?': str,
            'in_selection?': str,
            'mode?': ('copy','move'), # migration model
            ...
        },
        'quota': [None, list], # quota per subsystem
        'order': str, # ie 'sequence' or 'random'
    },
    'species': list, # configuration details
}
```

7.5 Community Examples

7.5.1 Cooperative Symbiosis

As a supportive example of cooperative coevolution within the ESEC framework the `esec` package is configured to support a community organisation of two species and two matched populations. It is possible to create multiple populations for each species type if desired, or dynamic population configuration models, however that is beyond the scope of this example.

In a basic parameter separation scheme, each species of a community is allocated a single parameter of the problem domain. This type of parameter separation is simplistic and limited. As a more involved example, a simple string matching problem landscape is created, in which the fitness of any solution is the number of correctly matched characters to a target string. A number of species are used to search this domain, in a manner that requires the species cooperate.

Each solution is composed of two species individuals mapped to form a single solution. A basic textual representation of the domain is presented in Figure 7.3. Each individual is represented by a string of characters. The set of valid characters is inclusive of those in the target string, as well as an additional "-" character. A single individual does not contain enough characters to create a complete solution. Rather, an individual from each species is selected, and the two combined to create a single solution, which is then evaluted and assigned a fitness value. Two obvious concerns are raised: how are the two individuals combined, and how is fitness allocated to each individual?

This example deliberately uses a nontrival mapping of value characters from each individual. Each species is a different length and when combined the characters partially

```
Target Character Set: {A,B,C,D,E,F,G}
Target String Length: 8

Species Character Set: {A,B,C,D,E,F,G,-}
Species 1: Length=6, Offset=0
Species 2: Length=5, Offset=3

Mapping Example
Target String:      'FABCDAFB' (random)
Offset positions:    01234567
Species 1 sample:   'FB-CA-'
Species 2 sample:      '-ABFG'
Combined sample:    'FB-C-BFG'
Target matches:     'F--C--F-' (Fitness = 3)
```

Figure 7.3: String matching problem domain for cooperative species

overlapped. The length and overlap amount are fixed arbitrary values. In regions where only one species contributes a character the solution character is easily resolved. Within the overlap region, when both individuals contribute a valid character, the solution character is considered unknown ("-") to encourage unique species contributions (see Figure 7.3).

The fitness of a solution is the number of matches with the target string. Partial credit is assigned to each particular individual based on the number of contributions it made to the solution. In this simple model there is no direct penalty for contributing a majority of the target characters; it is possible that one species provide most of the final characters. In natural systems resource limitations are often a strong fitness pressure that motivate cooperations; penalising excessive contribution (as "wasteful" behaviour) is a useful fitness consideration.

Within the system configuration section shown in Listing 7.3 we can see how a single population configuration is defined which is used twice (as specified by the `count` list of integers) to create population instances. A simple panmictic subpopulation topology is used for each species.

The species details for each population are defined in the `species` section, and although there are "two" species conceptually (with different length and offset values), the basic details are the same and so the second species configuration simply refers to the first. The "length" needed for each species is actually determined by the function named in `interact.mapping.on_init`. (See the full configuration provided on the CDROM for exact details.) Note that each species uses "matchup" evaluation (`genome.eval`) rather than direct simple evaluation against the landscape.

The four common `interact` details are the `policy`, `matchup`, `interval`, and `mapping` values. Because the `matchup` has been specified as "best" in this case the interaction topology is implied to be a full graph, where each individual of one population is evaluated with the best member of each other species. Each population is considered in sequential order, and with no quota limitations.

Details for how individuals from each species population are combined and evaluated as a single landscape solution are defined by the `mapping` dictionary. This typically involves initialisation details (already mentioned) and a specialised evaluation function. In this case the mapping evaluation function is named which is subsequently called each time a

Listing 7.3: Cooperative ESEC community configuration example

```
config = {
    'EA': 'Cooperative Community Example',
    # Begin with a list of species
    'species': [
        # 1. Species A
        {
            'genome': {
                'type': 'char', 'init': 'random', 'eval': 'matchup'
            },
            'recombine': { ... },
            'mutate': { ... },
        },
        # 2. Species B is the same as species A
        '0 copy',
    ],
    # Specify a community that contains multiple subsystems
    'system': {
        'type': 'community',
        'systems': [ { # list of subsystem details
            # A simple population graph for both species
            'type': 'population',
            'topology': 'panmictic',
            'size': 20, # population size
            'epoch': 'on_gen',
            'breed': {
                'size': 20, # 'all' at once (generational)
            },
            'survive': {
                'selection': 'all',
                'group': 'offspring', # default non-overlapping
            },
        } ],
        'count': [1,1], # each subsystem used once
        'interact': {
            'policy': 'cooperate', #symbiosis
            'matchup': 'best',
            'interval': 'on_gen',
            'mapping': {
                'on_init': 'mapping_on_init', # function name
                'eval_fn': 'mapping_eval_fn', # function name
            },
        },
        'quota': None,
        'order': 'sequence',
    },
    # 'character' value landscape details
    'landscape': { ... },
}
```

group of individuals (selected using the `matchup` settings) needs to be evaluated. Fitness is credited to a single matchup contributor. This means that each individual gets given a single fitness value even though the "best" individuals from each population are selected and used many times during the evaluation of the entire population.

If different fitness values need to be assigned to different species this can also be done using a custom evaluation function, or from a specialised landscape evaluation function

that returns multiple fitness values and working in concert with a special mapping evaluation function.

Figure 7.4 shows a single representative run of the cooperative symbiosis community example described. It can be seen in the overall solution fitness value, and in the subsystem species fitness values, that the overall success increases as a result of incremental improvements and contributions from both species.

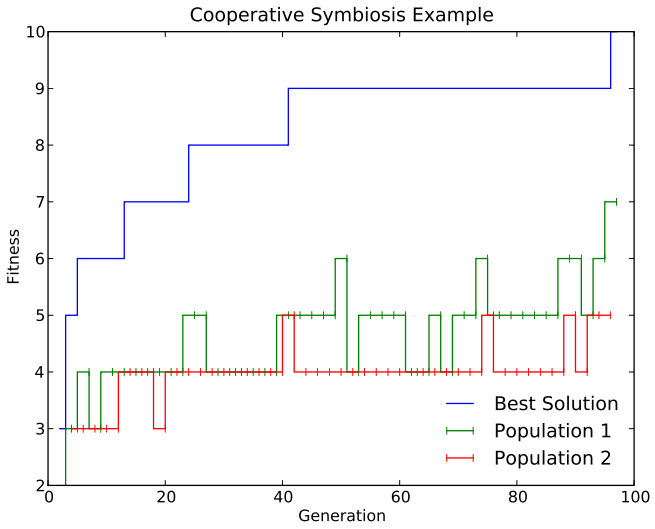


Figure 7.4: A representative plot of the best solution fitness and subpopulation (species) fitness values (solution contributions) from a sample run of the cooperative symbiosis example of a community system within the ESEC framework.

7.5.2 Competitive Predator-Prey

As an example of ESEC support for a classic predator-prey model, the `esec` package is configured for a community cohabitation configuration of two species in competition with each other. In particular, a simple search domain landscape is created to demonstrate a classic competitive "arms-race" scenario of adaptation between two species. In many EC examples the predator species is the main objective function (solution) while the prey species are test cases. It can be important for the test cases to begin at a simple level.

Listing 7.4 shows the essential aspects of the configuration details. Note that two different species and their population subsystems have been specified. Each species has a basic but different population topology; the prey population is a standard Moore-style lattice neighbourhood (eight neighbours), while the predator population is a simple panmictic population. In this way, the prey population is localised to particular regions of the community environment, while the predator species is considered as having a high degree of mobility for reproduction (presented by a full graph).

In contrast to the cooperative example presented earlier the result of evaluation is different for each species. Prey individuals are rewarded (given a good fitness value) for being challenging for predator species, while predator species are rewarded based on the number of success cases they achieve within a sample group of interactions with prey. Un-

like a true predator-prey model, in a competitive policy prey individuals are not removed by predators from the population.

A simple binary competitive problem landscape is contrived to demonstrate arms-race behaviour between two species. A textual example of the domain is shown in Figure 7.5. Predator and prey species are composed of matching length binary value strings. The evaluation of individuals is done by comparing each boolean value, and awarding a positive or negative fitness score. The competition is deliberately weighted so that success for prey individuals is independent of predators (simply the sum of true boolean values), while the predator fitness is dependent on the number of matching true values in prey species. The initial population of both species is seeded with zero value genomes to demonstrate incremental development.

Evaluation of each individual is performed by a sample group of matchup pairings between each species, as in the cooperative community example. The selection of matchup pairs is done using an interaction topology.

Predator individuals are randomly allocated ("scattered") to locations of the prey species topology. Conceptually, the mobile predator species move to an environment location and consume prey, and then move on to other locations.

The process of predator allocation can be triggered on any number of events, such as the generational rate of the predator species, the generational event of the prey species, or an epoch number of system-wide steps. In this case, the prey species' generational event is used to trigger the reallocation of prey species.

Considering the modularity of the design and configuration, the predator and prey species are not specifically aware of the interaction topology. Individuals from either species are simply used within their own "known" neighbourhood context. For a prey species, their survival fitness is based on their interaction with a sample of randomly allocated prey individuals, and reproduction fitness is based on a relative comparison with their neighbours – some of which will have experienced interaction with different predator species.

As can be seen in the simple results of a single simulation run (Figure 7.6), both predator and prey species adapt to achieve the maximum possible fitness for the predator species.

7.6 Ecosystem Examples

7.6.1 Basic Island Model

Consider a simple island model EA example presented as an ecosystem organisation configuration for the ESEC framework. Table 7.1 lists the essential specification details, and Listing 7.5 provides partial configuration details as needed for the `esec` software. The full configuration is available on the CDROM. (See Appendix D for a description of the CDROM contents.)

In Chapter 3 and the presentation of reference EC algorithms, Listing 3.5 showed a simple synchronous island population example EA where migration is controlled by a

System Representation	
Organisation Type	Ecosystem
Subsystem Type(s)	uniform population (island)
No. Subsystems	$d = 5$
Interaction Policy	Panmictic migration
...	...
Individuals	Bit-string (length l)
Subsystem Representation(s)	
Topology	Panmictic (size $= m = 20$)
Individuals	*<match system>*
...	...
Subsystem Selection Pressure	
Generations	Non-overlapping
Parent Selection	Fitness proportional
Mate Selection	Fitness proportional
Survivor Selection	Truncate (age)
Interaction Policy	
Migration Mode	Parallel (synchronised)
Migration Interval	$dt = 20$ (generations)
Emigrant Island	Order
Immigrant Island	Random
Emigrant Selection	Truncate (best)
Immigrant Survival	Truncate (best)
Migrate Group Size	k (individuals)
Communication Mode	Copy

Table 7.1: Island EA configuration as a simple ecosystem

single method. Given an infrequent migration policy (as described) this configuration is an especially good opportunity for decentralised (subprocess and distributed data) island algorithm control.

In this case, the ecosystem contains a small fixed number of subsystem "islands" (five), each a single species population system using a simple panmictic population topology of size 20. All subsystems are of the same population size and support the same (shared) species type allowing simple migration of individuals between islands without adaptation. The ecosystem controller allows each subsystem to evolve in parallel, using an equal-quota model.

There are many possible island interaction policies and topologies. Here a panmictic topology is used with a simple "island" migration policy, specifying that individuals from each island may migrate to any other island. An interval of 20 generations is set as the period of time between migration activity. Selection pressure is applied both to selection of individuals for migration, as well as the selection of individuals that are retained after migration. The migration mode in this case is "copy" so that individuals selected for migration are copied rather than moved and lost from their original island population.

The interaction topology is simply a fully connected graph. Other common forms are ring lattices and grid lattices, but clearly any topology could apply. Practically, and

as already noted, islands are typically distributed and connected in a manner that takes advantage of underlying computation hardware.

A representative sample run of the island ecosystem configuration is presented in Figure 7.7. A simple non-deceptive maximisation problem domain is shared among the islands. Note the influence of migration events on each islands fitness, and the overall best fitness value within the ecosystem.

Another possible interaction policy between islands is a "ladder" model, which creates a fitness-based hierarchy in which migration direction is specified. Source or sink islands can be created such that a bias of emigration or immigration occurs respectively. This could allow islands of "weaker" fitness levels (but perhaps useful genetic diversity) to avoid the disruptive immigrant influence of stronger individuals. Similarly "sink" islands can become useful, highly-competitive and strongly mixed populations that feed off the genetic diversity of source islands that feed into them.

The coordination and selection policies required to manage an island-based ecosystem model is much simpler than those required for multi-species coevolution models (competitive or cooperative) discussed earlier.

7.6.2 Complex Ecosystem

The island model, as researched and discussed by many, has clearly provided a very useful organisational model for evolutionary computation, and so its inclusion in a cohesive ecosystem framework is highly valued. As already described in the introduction of this chapter, the ecosystem organisation model within the ESEC framework is not limited to relatively simple island interaction models.

As a supportive example of the potential of the ESEC framework to describe complex systems, let us consider another, more complex, example. In the introduction, Figure 7.2 showed a conceptual diagram of an ecosystem composed of three subsystems: a simple population, a community and a nested ecosystem. Based around this general capacity, an ecosystem instance with three specific subsystems is defined; a simple single species panmictic population (species A), a cooperative coevolution community of two species (species B and C), and an island-style nested ecosystem (also for species A). A simple problem landscape is defined for which only a single "species" solution type is required. All subsystems will attempt to evolve solutions for the problem in their own manner, limited by evolution quota.

As the problem landscape only requires a single species, the cooperative dual-species community will need a mapping scheme so that its two species are used to form solutions. The specification of this is the same as that used previously in community organisation models. Note that the nested community subsystem contains its own mapping between the external representation requirements and internal representation, in a self-contained manner, which separates their unique concerns from the specification of the overall ecosystem.

The specific subsystem composition is illustrated in Figure 7.8, showing the specific

number of subsystems, their type, and some internal details for each subsystem (such as the specific number of island populations used in the nested ecosystem).

Listing 7.6 is the essential configuration details as required for this example in the ESEC framework. The complete configuration is available on the CDROM. (See Appendix D for details.)

This example is provided as support that the ecosystem model, as implemented by the `esec` package, is capable of describing complex composite systems.

The landscape selected for the example does not contain NKC style epistatic complexities between subsystem species. As already noted, a simple landscape was selected in order to reduce the already contrived complexity of the overall ecosystem. An NKC style problem domain can be applied, in which the relative progress of evolutionary adaptation for each subsystem is limited by both the search landscape complexity and the ecosystem complexity.

7.7 Open Questions

There are many interesting avenues for investigation for both the community and the ecosystem organisation models presented as part of the ESEC framework. Within the brief examples presented, a range of simple and complex interactions has been shown providing an initial view of the open research potential.

By explicitly supporting interaction topology the model supports not only multiple interacting species, but also habitation and other ecological concepts. The number of topology based possibilities in defining interaction between multiple species is broad, and based on existing work, the utilisation of coadapting species is a useful paradigm worth exploring.

The ESEC model of community organisation clearly supports many established coadaptation models, as well as providing a clear context for novel extensions. Understanding and predicting when models of multiple species are a useful EC search paradigm is a difficult question. A number of questions can be asked of a community EC model. Perhaps the most dominant questions are centred around *what exactly the benefit of multiple species is* and *what problem domains are best matched to a community search model?* From these general questions follow many other model and implementation related questions. For example:

- Is there an advantage to dynamic species formation and reduction?
- Does a model occupancy and variable cohabitation resource pressure offer any benefit?
- Are lattice or geometry style lattice population topologies important or required for particular forms of coadaptation? (Would simpler random models suffice?)
- What is the sensitivity of search outcomes to community parameters?

- Could interaction and pairing models be dynamically modelled or formed, and would this be useful for particular problem domains?
- Are community organisations more robust than single species population models, in particular when applied to dynamic search environments?
- When should a community model not be used? Are there clear domains where a simpler organisation model is preferred?

Many of these questions, or aspects of them, have been addressed to some degree in previous work, such as the cooperative coevolutionary systems presented by De Jong and Potter, and the ecological (cellular) algorithm approaches of Kirley (considered in Section 5.4.4).

As the ESEC framework can support existing models, as well as an additional range of interaction topologies not previously explored, it is a somewhat obvious avenue for future research to include, validate and extend prior work. This could include the following:

- Rewired lattices: Many multi-species coevolutionary systems (both competitive and cooperative) have used lattice population topology. It would be interesting to investigate the influence that rewiring has on coevolution performance by first validating existing models and results, and then applying levels of rewiring. Would a random network of similar resources perform differently to basic lattice structures?
- Endosymbiosis: The endosymbiosis model of Kim et al. [192] discussed earlier is an interesting model for both its species distinctions and its community level interactions and compositions. Species composition interaction models are a challenging aspect of coevolution.

The application of coevolutionary search is not without criticisms, most of which are based not on a fundamental flaw of coevolutionary concepts, but rather implementation limits or lack of consideration for appropriate influences (such as selection operators or population structure). In order to clarify particular concerns or points, the ESEC model (and `esec` package) could be used.

The notion of an ecosystem organisational model provides several high-level opportunities. Consider the following features, most of which apply to both ecosystem and community organisations:

- Subsystems: How many subsystems should there be, and of what type and degree of complexity. Should they be homogenous?
- Interaction, of which there are several aspects:
  - Topology: What is an appropriate structure for interactions, and should they be explicitly defined by the ecosystem, or an emergent property of individuals or subsystems?
  - Frequency: Is there an general principle of interaction frequency from macro to micro levels of organisation, and how sensitive are systems to this property?

  - Model: How are individuals of one subsystem influenced by those of another, and how is influence applied? Are there principles underlying the use of average influence, subsystem averages or interaction samples?

- Quota: Energy or resource limited systems can be used to influence evolutionary search outcomes, and so how should this be applied to subsystems? Should quota vales be dynamic? What events are significant?

- System morphology: Should the number, size, type and composition of subsystems change? Should this be coupled with fitness or quota limits as indicators for change?

These questions are aligned to some degree with those presented for community models, complicated by the potential for additional interactions between complex subsystems.

As presented in Section 6.4 of Chapter 6, the variability of population topology creates additional parameters for EAs to utilise. The ESEC model suggests additional community and ecosystem structures, also with additional parameters. Determining appropriate topology form and suitable parameter values creates an extensive open area of research.

Perhaps the most viable avenue to explore these concerns is the use of parameter setting and control [97, 219]. This is one of the most important and interesting research directions relevant to the overall ecosystem model presented in this thesis. Future work should investigate the use and possible benefit of self-adaptive topologies with the ESEC model, not only at the level of population, but also at community and collective ecosystem levels.

7.8 Closing

This chapter has presented community as a specific organisational model within the ESEC framework, incorporating notions of cohabitation and interaction topology between populations of different species. Many different interaction types can be explored using the ESEC framework and the `esec` package, including cooperative coadaptation and classic predator-prey cohabitation models.

As an addition to many coevolutionary examples presented in literature, the community model presented here includes an explicit evolution quota, such that the rate of evolutionary development per species (or subpopulation) can be specified and weighted. This provides a means of changing the rate of adaptation each species experiences, and can be altered as a way of balancing species concerns with respect to overall community performance.

Two relatively simple coevolutionary example configurations, a cooperative symbiosis and a competitive “arms race” model, have been presented to demonstrate the viability of the community abstraction and its practical implementation. Exploring the many possible applications of the community coevolutionary model is outside the scope of this thesis, and indeed the body of work in the area of coadaptation is large and growing; it is a useful paradigm.

Within an entire ecological model of evolutionary computation, a community organisation is a useful system component, and it is possible for the community to be utilised within a larger organisational composition of an ecosystem as a subsystem component.

This chapter has also illustrated how the proposed ecosystem organisational model of evolutionary computation is able to represent the desirable qualities of an ecological model. Considerable past work by many authors supports that distributed and coarse grained EC models are useful and effective. As examples, the ESEC framework is used to represent a simple classic island population dEA model, as well as a complex ecosystem composition using all three system organisational levels as subsystems: population, community and ecosystem.

Thorough experiments and analysis of complex ecosystem models are beyond the scope of this thesis, however the ESEC framework is a viable platform to support investigation in these areas. An important underlying principle for any investigations of a complex system should be clear questions based on a consistent underlying framework. There are many open questions related to the features and potential of complex ecosystem models, in particular at the organisation level of ecosystem.

Listing 7.4: Predator-prey coevolution community example

```
config = {
    'EA': 'Predator-Prey Coevolution Example',
        # Specify community species
    'species': [ # list of subsystem species
        { # 1. Species A: Predator
            'genome': {
                'type': 'binary', 'init': 'zero', 'eval': 'matchup'
            },
            ... # + recombine and mutate settings
        },
        { # 2. Species B: Prey
            'genome': {
                'type': 'binary', 'init': 'zero', 'eval': 'matchup'
            },
            ... # + recombine and mutate settings
        },
    ],
        # Community of interacting subsystems
    'system': {
        'type': 'community',
        'systems': [ # list of subsystem details
            { # Predator species population - simple full graph
                'type': 'population',
                'topology': 'panmictic',
                'size': 20, # number of predators
                ... # + epoch, breed.size, replace, survive
            },
            { # Prey species population - Moore lattice
                'type': 'structured',
                'topology': {
                    'type': 'lattice', 'dim': [10,10], ...
                },
                'size': '#topology', # number of prey
                ... # + epoch, breed.size, replace, survive
            }
        ],
        'count': [1,1], # each subsystem created once
        'interact': {
            'policy': 'compete', # predator-prey
            'interval': 'on_gen',
            'matchup': {
                'count': [1,1],
                0: ['*', {'src': 'nei', 'selection': 'uniform_random' }],
                1: [{'src': 'all', 'selection': 'uniform_random' }, '*'],
            },
            'mapping': {
                'on_init': 'mapping_on_init',
                'eval_fn': 'mapping_eval_fn',
            },
        },
        'quota': None, # or a quota per subpopulation
        'order': 'sequence',
    },
    # Custom landscape details
    'landscape': { 'type: 'custom'; ... },
}
```

```
(Predator,Prey) = Predator score
(0,0) =  0
(0,1) = -1
(1,0) =  0
(1,1) = +2

Example:
Predator     0010
Prey 1       1010, Predator score = -1+0+2+0 =  1
Prey 2       0111, Predator score = +0-1+2-1 =  0
Prey 3       1101, Predator score = -1-1+0-1 = -3
```

Figure 7.5: Binary competitive domain for predator-prey species

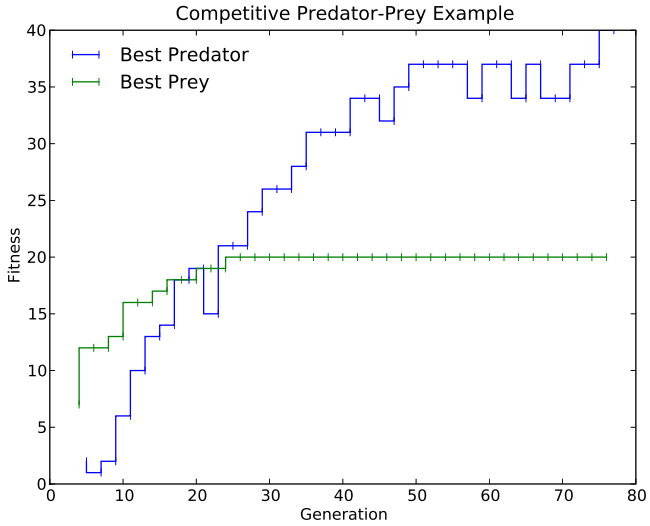


Figure 7.6: A representative plot of subpopulation (species) fitness values from a sample run of the competitive symbiosis example of a community system within the ESEC model. The binary predator-prey problem domain is described in Figure 7.3. Note that the prey species soon achieves maximal fitness, and this then enables the dependent predator species to improve also.

Listing 7.5: Island EA as an ESEC ecosystem configuration

```
config = {
    'EA': 'Island',
    'system': {
        'type': 'ecosystem',
        'systems': [ {
            # A simple generational population model
            'type': 'population',
            # subtype - panmictic generational implied
            'topology': 'panmictic',
            'size': 20, # island size
            'epoch': 'on_gen',
            'breed': {
                'size': 20, # 'all' at once (generational)
            },
            'replace': {
                'selection': 'all', # parents[:] = survivors
            },
            'survive': {
                'selection': 'truncate_best',
                'group': 'offspring+parents', # overlapping
            },
            'adapt': ... # optional
        } ],
        'count': 5, # number of island subsystems
        'interact': {
            'policy': 'island',
            'topology': 'panmictic',
            'interval': 20, # generations
            'size': 5, # number sent per
            'dest': 'random', # or 'order' or 'graph'
            'out_selection': 'truncate_best',
            'in_selection': 'truncate_best',
            'mode': 'copy',
        }
        'order': 'sequence'
    },
    'species': [ { # A single common shared species
        'genome': { 'type': 'binary', ... },
        'recombine': { ... },
        'mutate': { ... },
    } ],
}
```

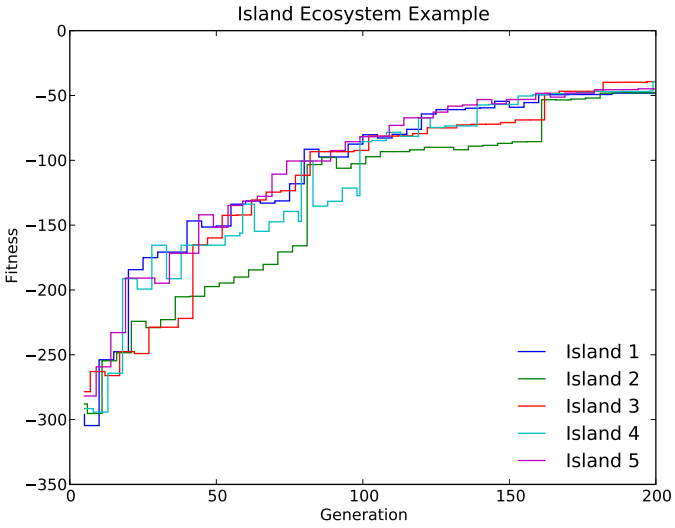


Figure 7.7: Island ecosystem composition example. All islands share a simple maximisation problem domain. Each island is the same size, and migration events occur every 20 generations, which can assist in some cases by transferring useful aspects of a good solution between island populations.

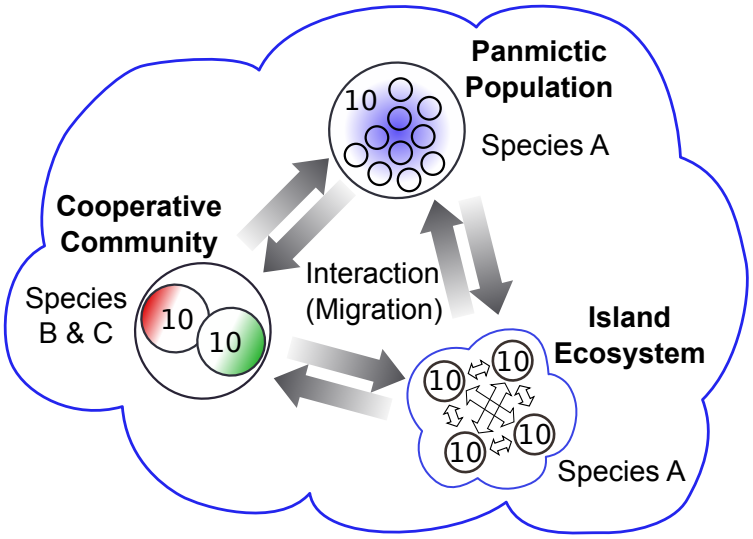


Figure 7.8: Composition of a complex ESEC ecosystem example

Listing 7.6: Complex ESEC ecosystem configuration example

```
config = {
    'EA': 'Ecosystem Example',
    'system': {
        'type': 'ecosystem',
        'systems': [ # list of subsystem details
            { ... }, # 1. simple panmictic population
            { ... }, # 2. two cooperative species
            { ... }, # 3. island EA ecosystem
        ],
        # 'count': 3, # implied
        'interact': {
            'policy': 'migrate',
            'topology': 'panmictic',
            'size': 2,
            ...
        },
        'quota': [ # quota per subsystem
            ...
        ],
        'order': 'sequence',
    },
    'species': [ # list of subsystem species
        { ... }, # 1. species A
        [        # 2. list of co-species
            { ... }, # species B
            { ... }, # species C
        ],
        '0 copy',   # 3. species A
    ],
}
```

Chapter 8

Conclusions

8.1 Overview

The overriding motivation of this thesis was to create a new ecosystem model of evolutionary computation (ESEC) and to investigate the influence that topology and interaction has on the outcome of evolutionary computation search.

A new ESEC model was proposed, using additional inspiration from biology, ecology and ecosystems to explicitly include the notion of structure in a manner that supports complex and efficient topology.

One of the strongest investigative outcomes of this work is that the specification of topology does influence both the EA search efficacy and efficiency. This motivates future investigations to consider in more detail how and why such influence can be used to an advantage as a way of optimising EC search applications.

8.2 Contributions

To address the stated research goals, detailed reviews of three important and interrelated fields were presented in Part I of the thesis:

- Chapter 2 presented concepts and properties of the domains of *ecology, ecosystems and evolution*. The accompanying glossary in Appendix A.1 supports the review. Of particular importance is the *ecosystem* as a structural model, which later provides an organisational architecture for extending simple evolutionary computation models.

- Chapter 3 introduced the field of *evolutionary computation* (EC) and evolutionary algorithms (EA) as a biologically inspired search metaphor, and presented the components of an EA with an emphasis on structural aspects and features that are often obscured or implicit in existing work.

- Chapter 4 examined current research and models of *graph theory and complex systems*, with particular emphasis on graph measurements of topological properties and models of graph development. Appendix A.2 is a detailed glossary to support the review.

All three domains are strongly coupled, with mutual benefit gained through an understanding of each field in greater detail.

Part II of the thesis has presented an ecosystem model for evolutionary computation (ESEC) using the reviewed fields, and explored three organisational levels of the ESEC model (*population*, *community* and *ecosystem*) and their relationship to existing EC models.

The main contributions of this thesis are:

- The development of an Ecosystem model of Evolutionary Computation, presented in Chapter 5, which is inclusive and supportive of prior EC work and supportive of additional topological features.
- Software realisation of the proposed model (the `esec` package) written in Python to provide a flexible platform for algorithmic experimentation. See Appendix E and the CDROM documents (Appendix D) for additional details.
- Demonstrated ability of the proposed model to investigate different organisational models at population (Chapter 6), community and ecosystem (Chapter 7) levels.
- The creation of a set of investigation themes using the proposed model, and a body of empirical work to address some of the questions raised in relation to single species population models of differing population topology qualities (Chapter 6).

To support the presentation of topology models, Appendix C is a detailed survey of the properties and models that are central to the ESEC model and investigation presented.

Similarly, as part of the ESEC realisation in the `esec` package, an extensive collection of classic EC benchmark problem domains were considered and implemented. A large portion of these are detailed in Appendix B including, where applicable, motivations and features of each landscape.

The empirical results presented in Chapter 6 clearly showed that topology properties of a population influence evolutionary process. Subsequently, topology can influence the efficacy of success, the quality of solutions found, and the efficiency (such as the search time required) of the search. Understanding the relationships of topology and EC, and in particular within a ecosystem based framework, provides a useful means of configuring EAs in ways that may improve the likelihood of high quality solutions and/or the efficiency of search for particular domains.

8.3 Future Work

Throughout the thesis a number of questions and extension have been presented. In particular a number of key questions were presented with the ESEC model description in Section 5.5 of Chapter 5, as a model capable of incorporating a broad range of features opens many avenues for investigation. Questions can be grouped into two main areas, biological models and graph theory concepts, although there is typically a close relationship between both.

These include, for example, consideration of the importance or influence of the following topics and questions:

- Ecosystem resources: What is the impact of resource (quota) limits, and should they be dynamically adapted? Could interactions also be limited as an analogy of cost?
- Mobility: Are models of movement useful or important? Are there advantages, and should they be based on particular geometric constraints, or do other models have benefits (such as small-world networks)?
- Gradients and Flow: Are directed interaction models (ie. for population structure) influential, extending the concepts of update order on lattice graphs? How do existing models of population topology "gradients" compare and can the two models be unified?
- Successional/Episodic Change: Consider episodic models, including extinction events, migration, successional change and founder/colonisation models. Do these models offer particular EC search advantages? Are some advantages more pronounced within community (multiple-species) models?
- Layered Populations: Can models of age or interaction structure (hierarchy) within single species populations be used to assist EC search? (Consider the delayed replacement model for juvenile individuals considered in Chapter 6.)
- Dynamic Topology: Are there advantages in the use of dynamic population size and/or structure? Could the interaction of individuals be used to alter population topology in a useful manner? Consider growth models for populations, and phases of behaviour (which links to successional models). How disruptive are dynamic topology changes?
- Motifs: Are there particular motif structures that are observed in natural systems that may relate or be useful for EC search? Is there an advantage to population or interaction topology that is based on a particular distribution or presence of motif components.

The incorporation of a number of existing EC models, in particular those that explicitly utilise topology, is important as this facilitates comparison of any new or alternative EC techniques to existing results, and validation of other work.

Specifying and altering topology provides a means of influencing diversity, "niche" like behaviour with EC search. This applies to single species populations, within community models of multiple interacting species, and ecosystem compositions of nested subsystems. As there are well established explicit niche schemes, the benefit of topology controlled diversity or niche behaviour should be considered with respect to explicit approaches.

Problem generator domains could be used to better understand the relationship between domain modality and population diversity and niche performance, and consequently EC search robustness. For example, the MSG problem domain has been implemented

within the `esec` package to facilitate exactly this form of investigation. The approach could be applied across ESEC organisational models, from populations to communities of multiple interacting species. Consider, for example, if the modality of a problem domain has a relationship with the diversity support of the population structure. A lattice population would be expected to have greater diversity than simple full graph population.

Many topologies and EC operations are amenable to parallel and distributed implementation. The ESEC model and the `esec` package implementation could be extended to support parallel and distributed processing explicitly within configuration details. This has practical benefits in expediting simulation time, but also extends the capabilities of the ESEC model to include, and thus enable comparison with models such as Opportunistic Evolution in which search is distributed and adapted to available computing resources.

Complex search complexity is only beneficial when the search domain presents challenges and where improvements in search performance justify the complexity. Extensions and consideration or future work should be balanced within this practical criteria. Increasing algorithm complexity without qualification can easily reduce effective techniques to poor ones.

8.4 Closing Comment

Some EC investigations are motivated on the basis of characterising ecological models and observations. However the primary intent of the ESEC model is to support evolutionary search within an ecosystem architecture

There continues to be a practical need to develop new and robust search algorithms. Not only can biology provide interesting models and inspiration for search algorithms, but the expanding fields of complex systems and graph topology models and properties can also add to our knowledge and approaches. The ESEC model supports both biological inspiration and lessons from topology, enabling evolutionary search within an ecosystem architecture.

Part III

References and Appendices

References

[1] David H. Ackley. *A Connectionist Machine for Genetic Hillclimbing*. Kluwer, Boston, 1987. 341

[2] David H. Ackley. *Stochastic Iterated Genetic Hillclimbing*. PhD thesis, Carnegie Mellon University, Pittsburge, PA, 1987. 45, 341

[3] Himanshu Agrawal. Extreme Self-Organization in Networks Constructed from Gene Expression Data. *Physical Review Letters*, 89:268702, 2002. 128

[4] Enrique Alba. Parallel Evolutionary Algorithms Can Achieve Superlinear Performance. *Information Processing Letters*, 82(1):7–13, April 2002. 69

[5] Enrique Alba, Mario Giacobini, Marco Tomassini, and Sergio Romero. Comparing Synchronous and Asynchronous Cellular Genetic Algorithms. In Juan J. Merelo Guervós, Panagiotis Adamidis, Hans-Georg Beyer, José Luis Fernández-Villaca nas Martín, and Hans-Paul Schwefel, editors, *PPSN*, volume 2439 of *Lecture Notes in Computer Science*, pages 601–610. Springer, 2002. 218

[6] Enrique Alba and Marco Tomassini. Parallelism and Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation*, 6(5):443–461, October 2002. 63, 64, 68, 69, 153

[7] Enrique Alba and José M. Troya. A Survey of Parallel Distributed Genetic Algorithms. *Complexity*, 4(4):31–52, 1999. 64, 68, 153

[8] Enrique Alba and José M. Troya. Cellular Evolutionary Algorithms: Evaluating the Influence of Ratio. In Marc Schoenauer, Kalyanmoy Deb, Günter Rudolph, Xin Yao, Evelyne Lutton, Juan J. Merelo Guervós, and Hans-Paul Schwefel, editors, *PPSN*, volume 1917 of *Lecture Notes in Computer Science*, pages 29–38. Springer, 2000. 156

[9] Réka Albert and Albert-László Barabási. Statistical Mechanics of Complex Networks. *Reviews of Modern Physics*, 74(47):47–49, 2002. 83, 87, 98, 113, 114, 120

[10] Uri Alon. Network Motifs: Theory and Experimental Approaches. *Nature Reviews Genetics*, 8:450–461, 2007. 101, 102, 126

[11] Peter J. Angeline. Genetic Programming's Continued Evolution. In Peter J. Angeline and K. E. Kinnear, Jr., editors, *Advances in Genetic Programming 2*, chapter 1, pages 1–20. MIT Press, Cambridge, MA, USA, 1996. 65

[12] Peter J. Angeline and Jordan B. Pollack. Competitive Environments Evolve Better Solutions for Complex Tasks. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms and their Applications*, pages 264–270. Morgan Kaufmann, 1993. 161

[13] Daniel Angus and Clinton Woodward. Multiple Objective Ant Colony Optimisation. *Swarm Intelligence*, 3:69–85, 2009. 71

[14] Jaroslaw Arabas, Zbigniew Michalewicz, and Jan J. Mulawka:. GAVaPS - A Genetic Algorithm with Varying Population Size. In *Proceedings of the First IEEE Conference on Evolutionary Computation*, volume 1, pages 73–78, 1994. 51

[15] Robert Axelrod. The Evolution of Strategies in the Iterated Prisoner's Dilemma. In L. D. Davis, editor, *Genetic Algorithms and Simulated Annealing*, pages 32–41. Morgan Kaufmann, Los Altos, CA, 1987. 46

[16] Thomas Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, New York, 1996. 67, 330, 341

[17] Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors. *Handbook of Evolutionary Computation*. Oxford University Press, Oxford, 1997. 46, 47, 48, 49, 65, 66, 67, 354

[18] Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz. *Evolutionary Computation 1: Basic Algorithms and Operators*. Institute of Physics Publishing, Bristol, 2000. 47

[19] Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz. *Evolutionary Computation 2: Advanced Algorithms and Operators*. Institute of Physics Publishing, Bristol, 2000. 47, 155, 254

[20] Thomas Bäck, Frank Hoffmeister, and Hans-Paul Schwefel. A Survey of Evolution Strategies. In Lashon B. Booker and Richard K. Belew, editors, *Proceedings of the 4th International Conference on GAs*, pages 2–9. Morgan Kaufmann, 1991. 65

[21] Thomas Bäck, Gunter Rudolph, and Hans-Paul Schwefel. Evolutionary Programming and Evolution Strategies: Similarities and Differences. In David B. Fogel and J. Wirt Atmar, editors, *Proceedings of the Second Conference on Evolutionary Programming*, pages 11–22, La Jolla, CA, 1993. Evolutionary Programming Society. 66, 341

[22] James E. Baker. Reducing Bias and Inefficiency in the Selection Algorithm. In John J. Grefenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms and Their Applications*, pages 14–21, Hillsdale, NJ, 1987. Lawrence Erlbaum Associates. 55, 56

[23] Shumeet Baluja. Structure and Performance of Fine-grained Parallelism in Genetic Search. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 155–162. Morgan Kaufmann, 1993. 69

[24] Wolfgang Banzhaf, Peter Nordin, Robert E. Keller, and Frank D. Francone. *Genetic Programming - An Introduction; On the Automatic Evolution of Computer Programs and its Applications*. Morgan Kaufmann, San Mateo, CA, 1998. 67

[25] Albert-László Barabási. The Physics of the Web. http://physicsweb.org/articles/world/14/7/09, 2001. 12

[26] Albert-László Barabási. *Linked: The New Science of Networks*. Perseus, Cambridge, MA, 2002. 11, 81, 83, 87, 126

[27] Albert-László Barabási and Réka Albert. Emergence of Scaling in Random Networks. *Science*, 286(5439):509–512, 1999. 108, 109, 119

[28] Albert-László Barabási, Réka Albert, Hawoong Jeong, and Ginestra Bianconi. Power-Law Distribution of the World Wide Web. *Science*, 287:2115, 2000. 98

[29] Albert-László Barabási and Zoltán N. Oltvai. Network Biology: Understanding the Cell's Functional Organization. *Nature Reviews Genetics*, 5(2):101–13, 2004. 123, 127

[30] Mauricio Barahona and Louis M. Pecora. Synchronization in Small-World Systems. *Physical Review Letters*, 89(5):054101, Jul 2002. 106

[31] Mark A. Bedau, Emile Snyder, and Norman J. Packard. A Classification of Long-Term Evolutionary Dynamics. In Christoph Adami, Richard K. Belew, Hiroaki Kitano, and Charles Taylor, editors, *Artifical Life VI*, pages 189–198. MIT Press, Cambridge, MA, 1998. 71

[32] Theodore C. Belding. The Distributed Genetic Algorithm Revisited. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 114–121, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc. 63, 68

[33] Peter J. Bentley. An Introduction to Evolutionary Design by Computers. In Peter J. Bentley, editor, *Evolutionary Design by Computers*, pages 1–73. Morgan Kaufmann, San Francisco, 1999. 46, 48

[34] Peter J. Bentley. *Evolutionary Design by Computers*. Morgan Kaufmann, San Mateo, CA, 1999. 65

[35] Hugues Bersini and Francisco J. Varela. Hints for Adaptive Problem Solving Gleaned from Immune Networks. In *Parallel Problem Solving from Nature, First Workshop PPSW 1*, Dortmund, October 1990. 40, 70

[36] Stefano Boccaletti, Vito Latora, Yamir Moreno, Mario Chavez, and Dong-Uk Hwang. Complex Networks: Structure and Dynamics. *Physics Reports*, 424(4-5):175–308, Fervier 2006. 81, 86, 87, 127, 128

[37] Stefan Boettcher. Extremal Optimization - Heuristics via Co-evolutionary Avalanches. *Computing in Science & Engineering*, 2:75–82, 2000. 70

[38] Stefan Boettcher and Allon G. Percus. Extremal Optimization: Methods Derived from Co-evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 1999. 70

[39] Bela Bollobás. *Modern Graph Theory*. Springer, New York, 1998. 87

[40] Stefan Bornholdt and Heinz Georg Schuster. *Handbook of Graphs and Networks: From the Genome to the Internet*. Wiley-VCH, Berlin, 2003. 87

[41] George E.P. Box. Evolutionary Operation: a Method for Increasing Industrial Productivity. *Applied Statistics*, 6(2):81–101, 1957. 40

[42] Hans J. Bremermann. Optimization through Evolution and Recombination. In M.C. Yovits, G.T. Jacobi, and G.D. Goldstine, editors, *Self-Organizing Systems*, pages 93–106. Spartan Books, 1962. 40

[43] Mark Buchanan. *Nexus: Small Worlds and the Ground-breaking Science of Networks*. Norton, New York, 2002. 87

[44] Larry Bull and Terence C. Fogarty. Artificial Symbiogenesis. *Artificial Life*, 2:269–292, 1995. 159, 161

[45] Raffaele Calabretta, Riccardo Galbiati, Stefano Nolfi, and Domenico Parisi. Two is Better than One: A Diploid Genotype for Neural Networks. *Neural Processing Letters*, 4(3)(4):149–155, 1996. 161

[46] Raffaele Calabretta, Stefano Nolfi, Domenico Parisi, and Günter P. Wagner. A Case Study of the Evolution of Modularity: Towards a Bridge Between Evolutionary Biology, Artificial Life, Neuro- and Cognitive Science. In Christoph Adami, Richard K. Belew, Hiroaki Kitano, and Charles Taylor, editors, *Proceedings of Artifical Life VI, Los Angeles*. MIT Press, 1998. 161

[47] Raffaele Calabretta, Stefano Nolfi, Domenico Parisi, and Günter P. Wagner. An Artificial Life Model for Investigating the Evolution of Modularity. In Y. Bar-Yam, editor, *Proceedings of the International Conference on Complex Systems, Nashua, NH*. Addison-Wesley, 1998. 161

[48] Raffaele Calabretta, Stefano Nolfi, Domenico Parisi, and Günter P. Wagner. Emergence of Functional Modularity in Robots. In Christoph Adami, Richard K. Belew, Hiroaki Kitano, and Charles Taylor, editors, *Proceedings of Artificial Life VI, Los Angeles*, 1998. 161

[49] Erick Cantú-Paz. A Survey of Parallel Genetic Algorithms. Technical Report IlliGAL 97003, Illinois Genetic Algorithms Laboratory, University of Illinois, Urbana-Champaign, 1997. 63, 64, 154

[50] Erick Cantú-Paz. *Designing Efficient and Accurate Parallel Genetic Algorithms*. PhD thesis, University of Illinois, 1999. 154

[51] Erick Cantú-Paz. *Efficient and Accurate Parallel Genetic Algorithms*. Kluwer Academic Publishers, United States, 2000. 51, 60, 63, 64, 69, 153, 154

[52] Erick Cantú-Paz. Migration Policies, Selection Pressure, and Parallel Evolutionary Algorithms. *Journal of Heuristics*, 7(4):311–334, 2001. 51, 60, 78, 154

[53] Lewis Carroll. *Through the Looking-Glass, and What Alice Found There*. Macmillan, United Kingdom, 1872. 161

[54] Walter Cedeño. *The Multi-Niche Crowding Genetic Algorithm: Analysis and Applications*. PhD thesis, Computer Science Department, University of California, Davis CA, September 1995. 57, 152

[55] Hao Chen, Nicholas S. Flann, and Daniel W. Watson. Parallel Genetic Simulated Annealing: A Massively Parallel SIMD Algorithm. *IEEE Transactions on Parallel and Distributed Systems*, 9(2):805–811, February 1998. 360

[56] Jason P. Cohoon, Shailesh U. Hegde, Worthy N. Martin, and Dana S. Richards. Punctuated Equlibria: A Parallel Genetic Algorithm. In John J. Grefenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms*, pages 148–154, Hillsdale, NJ, 1987. Lawrence Earlbaum Associates. 63

[57] Howard Copland and Tim Hendtlass. An Evolutionary Algorithm with a Genetic Encoding Scheme. In José Mira and Angel P. Del Pobil, editors, *Methodology and Tools in Knowledge-Based Systems, 11th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE-98, Castellón, Spain, June 1-4, 1998*, volume 1415:1 of *Lecture Notes in Computer Science*, pages 632–639. Springer, 1998. 50

[58] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, 2nd edition, 2001. 102

[59] Nichael Lynn Cramer. A Representation for the Adaptive Generation of Simple Sequential Programs. In John J. Grefenstette, editor, *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, pages 183–187, Hillsdale, NJ, 1985. Lawrence Erlbaum Associates. 67

[60] Paolo Crucitti, Vito Latora, Massimo Marchiori, and Andrea Rapisarda. Complex Systems: Analysis and Models of Real-World Networks. In Larissa S. Brizhik Francesco Musumeci, Mae-Wan Ho, editor, *Energy and Information Transfer in Biological Systems: How Physics Could Enrich Biological Understanding, Proceedings of the International Workshop Acireale, Catania, Italy 18 - 22 September 2002*, pages 188–204, Singapore, 2003. World Scientific Publishing Co. Pte. Ltd. 84, 104

[61] Gábor Csárdi and Tamás Nepusz. The igraph Software Package for Complex Network Research. In *Proceedings of the International Conference on Complex Systems*, 2006. 96

[62] Charles R. Darwin. *On the Origin of Species by Means of Natural Selection or the Preservation of Favoured Races in the Struggle for Life*. John Murray, London, 1859. 3

[63] Charles R. Darwin and Alfred R. Wallace. *Evolution by Natural Selection*. Cambridge University Press, Cambridge, 1958. 3

[64] Dipankar Dasgupta, editor. *Artificial Immune Systems and Their Applications*. Springer-Verlag, Inc., Berlin, 1999. 70

[65] Yuval Davidor. A Naturally Occuring Niche & Species Phenomenon. In Richard K. Belew and Lashon B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 257–263, San Mateo, 1991. Morgan Kaufmann. 51, 63, 154, 155, 156

[66] Yuval Davidor. Free the Spirit of Evolutionary Computing: The Ecological Genetic Algorithm Paradigm. In Raymond C. Paton, editor, *Computing With Biological Metaphors*, pages 311–322. Chapman and Hall, London, UK, 1994. 63, 154, 155

[67] Yuval Davidor, Takeshi Yamada, and Ryohei Nakano. The ECOlogical Framework II: Improving GA Performance at Virtually Zero Cost. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA)*, pages 171–176, San Mateo, CA, 1993. Morgan Kaufmann. 155

[68] Ron Davidson and David Harel. Drawing Graphs Nicely using Simulated Annealing. Technical Report CS 89-13, Department of Applied Mathematics and Computer Science, The Wiezmann Institute of Science, Rehovot, 1989. 95

[69] Ron Davidson and David Harel. Drawing Graphs Nicely using Simulated Annealing. *ACM Transactions on Graphics*, 15(4):301–331, 1996. 95

[70] Lawrence Davis. *The Handbook of Genetic Algorithms*. Von Nostrand Reinhold, New York, first edition, 1991. 47, 59

[71] Richard Dawkins. *The Selfish Gene*. Oxford University Press, Oxford, 1976. 37

[72] Richard Dawkins. *The Extended Phenotype*. Oxford University Press, Oxford, 1982. 37

[73] Richard Dawkins. *The Blind Watchmaker*, volume ISBN 0-393-31570-3. W. W. Norton & Company, Inc., New York, 1986. 41

[74] Kenneth A. De Jong. *Analysis of Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, Ann Arbor, MI, 1975. 51, 152, 153, 156, 330

[75] Kenneth A. De Jong. Generation Gap Revisited. In *Proceedings of the Foundation of Genetic Algorithms 2*, pages 19–28, San Mateo, CA, 1993. Morgan Kaufmann. 63

[76] Kenneth A. De Jong. Genetic Algorithms Are NOT Function Optimizers. In L. Darrell Whitley, editor, *Foundations of Genetic Algorithms 2*, pages 5–17. Morgan Kaufmann, CA, 1993. 44

[77] Kenneth A. De Jong. *Evolutionary Computation: A Unified Approach*. MIT Press, Cambridge, Massachusetts, 2006. 41, 44, 46, 47, 48, 54, 65, 67, 163

[78] Kenneth A. De Jong and Mitchell A. Potter. Evolving Complex Structures via Cooperative Coevolution. In *Fourth Annual Conference on Evolutionary Programming, San Diego, CA*, pages 307–317, Cambridge, MA, 1995. MIT Press. 156, 159, 160

[79] Kenneth A. De Jong, Mitchell A. Potter, and William M. Spears. Using Problem Generators to Explore the Effects of Epistasis. In *Proceedings of the Seventh International Conference on Genetic Algorithms, Michigan State University, East Lansing, MI,*, pages 338–345. Morgan Kaufmann, 1997. 349, 351

[80] Kalyanmoy Deb and Ram Bhushan Agrawal. Simulated Binary Crossover for Continuous Search Space. *Complex Systems*, 9:115–148, 1995. 75

[81] Kalyanmoy Deb, Ashish Anand, and Dhiraj Joshi. A Computationally Efficient Evolutionary Algorithm for Real-parameter Optimization. *Evolutionary Computation*, 10(4):371–395, 2002. 58, 74, 75, 76, 198

[82] Kalyanmoy Deb, Ashish Anand, and Dhiraj Joshi. A Computationally Efficient Evolutionary Algorithm for Real-Parameter Optimization. Report No. 2002003, KanGAL, April 2002. 67, 74

[83] Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. Algorithms for Drawing Graphs: An Annotated Bibliography. *Computational Geometry: Theory and Applications*, 4:235–282, 1994. 87, 93, 95

[84] Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1999. 93, 95

[85] Laurence Charles Ward Dixon and Gabor P. Szegő. The Optimization Problem: An Introduction. In L. C. W. Dixon and G. P. Szego, editors, *Towards Global Optimization II*. North Holland, 1978. 348

[86] Raul Donangelo and Kim Sneppen. Self-organization of Value and Demand. *Physica A: Statistical Mechanics and its Applications*, 276:572–580, 2000. 126

[87] Luca Donetti, Pablo I. Hurtado, and Miguel A. Muñoz. Entangled Networks, Synchronization, and Optimal Network Topology. *Physical Review Letters*, 95(18):188701–+, October 2005. 105

[88] Luca Donetti, Pablo I. Hurtado, and Miguel A. Muñoz. Synchronization in Network Structures: Entangled Topology as Optimal Architecture for Network Design. In *Computational Science - ICCS 2006*, volume 3993 of *Lecture Notes in Computer Science*, pages 1075–1082. Springer Berlin / Heidelberg, 2006. 105

[89] Kejutan Dontas and Kenneth A. De Jong. Discovery of Maximal Distance Codes using Genetic Algorithms. In *Proceedings of the 2nd International IEEE Conference on Tools for Artificial Intelligence*, number IEEE Cat. No. 90CH2915 in 7, pages 805–811, Herndon, VA, 6-9 Nov 1990. IEEE Computer Society Press, Los Alamitos, CA. 360

[90] Marco Dorigo and Thomas Stützle. *Ant Colony Optimization*. MIT Press, Cambridge, 2004. 71

[91] Sergey N. Dorogovtsev and José Fernando Ferreira Mendes. Evolution of Networks. *Advances in Physics*, 51:1079–1187, 2002. 87, 126

[92] Sergey N. Dorogovtsev and José Fernando Ferreira Mendes. *Evolution of Networks: From Biological Nets to the Internet and WWW*. Oxford University Press, 2003. 81, 87, 114, 119

[93] Peter Eades. A Heuristic for Graph Drawing. *Congressus Numerantium*, 42:149–160, 1984. 94

[94] Peter Eades and Roberto Tamassia. Algorithms for Drawing Graphs: An Annotated Bibliography. Technical report, Department of Computer Science. Providence, RI: Brown University, 1989. 95

[95] Eric E. Easom. A Survey of Global Optimization Techniques. MEng thesis, University of Louisville, 1990. 334

[96] David Eby, Ron C. Averill, Boris Gelfand, William F. Punch, Owen Mathews, and Erik D. Goodman. An Injection Island GA for Flywheel Design Optimization. In *Proceedings of the 5th European Congress on Intelligent Techniques and Soft Computing (EUFIT '97)*, pages 687–691, Aachen, 1997. Verlag Mainz. 255

[97] Agoston E. Eiben, Robert Hinterding, and Zbigniew Michalewicz. Parameter Control in Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2):124–141, 1999. 10, 267

[98] Agoston E. Eiben and James E. Smith. *Introduction to Evolutionary Computation*. Springer-Verlag, Berlin Heidelberg, 2003. 41, 44, 46, 47, 48, 51, 55, 59, 61, 65, 67, 152, 154, 155, 163, 250, 253, 354

[99] Agoston E. Eiben and J. K. van der Hauw. Solving 3-SAT with Adaptive Genetic Algorithms. In *In Proceedings of the 4th IEEE Conference on Evolutionary Computation*, pages 81–86. IEEE Service Center, 1997. 353

[100] Manfred Eigen and Peter Schuster. *The Hypercycle: A Principle of Natural Self-Organization*. Springer, Berlin, 1979. 127

[101] Niles Eldredge and Stephen Jay Gould. Punctuated Equlibria: an Alternative to Phyletic Gradualism. In T J M Schopf, editor, *Models of Paleobiology*, pages 82–115, San Francisco, CA, 1972. Freeman, Cooper. 63, 154

[102] Andries P. Engelbrecht. *Fundamentals of Computational Swarm Intelligence*. Wiley & Sons, 2006. 7, 9, 40, 46, 49, 71

[103] Paul Erdös and Alfréd Rényi. On Random Graphs I. *Publ. Math. Debrecen*, 6:290–297, 1959. 83, 109, 113

[104] J. Doyne Farmer, Norman H. Packard, and Alan S. Perelson. The Immune System, Adaptation and Machine Learning. *Physica D*, 2:187–204, 1986. 40, 70

[105] David E. Featherstone and Kendal Broadie. Wrestling with Pleiotropy: Genomic and Topological Analysis of the Yeast Gene Expression Network. *BioEssays*, 24(3):267–274, 2002. 128

[106] Sevan G. Ficici and Jordan B. Pollack. Challenges in Coevolutionary Learning: Arms-race Dynamics, Open-endedness, and Mediocre Stable States. In Christoph Adami, Richard K. Belew, Hiroaki Kitano, and Charles Taylor, editors, *Proceedings of the Sixth International Conference on Artificial Life*, pages 238–247, Cambridge, MA, 1998. The MIT Press. 162

[107] George B. Field and William C. Saslaw. A Statistical Model of the Formation of Stars and Interstellar Clouds. *The Astrophysical Journal*, 142:568, 1965. 109, 121

[108] J. Michael Fitzpatrick and John J. Grefenstette. Genetic Algorithm in Noisy Environment. *Machine Learning*, 3(2-3):101–120, 1988. 54

[109] David B. Fogel. *Evolving Artificial Intelligence*. PhD thesis, University of California, San Diego, CA, 1992. 66

[110] David B. Fogel. *Evolutionary Computation: Towards a New Philosophy of Machine Intelligence*. IEEE Press, Piscataway, NJ, 1995. 66

[111] David B. Fogel. *Evolutionary Computation: the Fossil Record*. IEEE Press, Piscataway, NJ, 1998. 40, 65

[112] David B. Fogel. *Blondie24: Playing at the Edge of AI*. Morgan Kaufmann, San Francisco, CA, 2002. 158

[113] David B. Fogel. *Evolutionary Computation: Towards a New Philosophy of Machine Intelligence*. IEEE Press, Piscataway, NJ, third edition, 2006. 5, 42, 47, 48, 62, 65, 163

[114] David B. Fogel and J. Wirt Atmar. Comparing Genetic Operators with Gaussian Mutations in Simulated Evolutionary Processes using Linear Systems. *Biological Cybernetics*, 63(1):111–114, 1990. 66

[115] David B. Fogel and L. C. Stayton. On the Effectiveness of Crossover in Simulated Evolutionary Optimization. *BioSystems*, 32(3):171–182, 1994. 66

[116] Lawerence J. Fogel. *Biotechnology: Concepts and Applications*. Prentice Hall, Englewood Cliffs, NJ, 1963. 66

[117] Lawerence J. Fogel, Alvin J. Owens, and Michael J. Walsh. *Artificial Intelligence through Simulated Evolution*. John Wiley & Sons, New York, 1966. 66, 67

[118] Carlos M. Fonseca and Peter J. Fleming. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In Stephanie Forrest, editor, *Genetic Algorithms: Proceedings of the Fifth International Conference*, pages 141–153, San Mateo, CA, 1993. Morgan Kaufmann. 53

[119] Carlos M. Fonseca and Peter J. Fleming. An Overview of Evolutionary Algorithms in Multiobjective Optimization. *Evolutionary Computation*, 3(1):1–16, 1995. 53

[120] Carlos M. Fonseca and Peter J. Fleming. Multiobjective Optimisation. In Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors, *Evolutionary Computation 2: Advanced Algorithms and Operators*, pages 25–37. Institute of Physics Publishing, Bristol, 2000. 53, 54

[121] Alex S. Fraser. Simulaton of genetic systems by automatic digital computers 1: Introduction. *Australian Journal of Biological Sciences*, 10:484–491, 1957. 40

[122] Thomas M. J. Fruchterman and Edward M. Reingold. Graph Drawing by Force-Directed Placement. *Software-Practices and Experience*, 21(11):1129–1164, 1991. 94, 95, 120

[123] John Fulcher and Lakhmi C. Jain, editors. *Computational Intelligence: A Compendium*, volume 115 of *Studies in Computational Intelligence*. Springer, 2008. 40, 41

[124] Marcus Gallagher and Bo Yuan. A General-Purpose Tunable Landscape Generator. *IEEE Transactions on Evolutionary Computation*, 10(5):590–603, October 2006. 189, 349, 362

[125] Abbas A. El Gamal, Lane A. Hemachandra, Itzhak Shperling, and Victor K. Wei. Using Simulated Annealing to Design Good Codes. *IEEE Transactions on Information Theory*, 33(1):116–123, 1987. 360

[126] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, New York, 1979. 352, 358

[127] Simon Garrett. How Do We Evaluate Artificial Immune Systems? *Evolutionary Computation*, 13(2):145–178, 2005. 70

[128] Ashish Ghosh and Shigeuoshi Tsutsui. *Advances in Evolutionary Computing: Theory and Applications*. Springer-Verlag, Germany, 2003. 41, 65

[129] Mario Giacobini, Enrique Alba, Andrea Tettamanzi, and Marco Tomassini. Modeling Selection Intensity for Toroidal Cellular Evolutionary Algorithms. In Kalyanmoy Deb, Riccardo Poli, Wolfgang Banzhaf, Hans-Georg Beyer, Edmund K. Burke, Paul J. Darwen, Dipankar Dasgupta, Dario Floreano, James A. Foster, Mark Harman, Owen Holland, Pier Luca Lanzi, Lee Spector, Andrea Tettamanzi, Dirk Thierens, and Andrew M. Tyrrell, editors, *GECCO (1)*, volume 3102 of *Lecture Notes in Computer Science*, pages 1138–1149. Springer, 2004. 218

[130] Michelle Girvan and Mark E. J. Newman. Community Structure in Social and Biological Networks. *Proceedings of the National Academy of Sciences U. S. A.*, 99:7821–7826, 2002. 102

[131] David Eugene Glover. *Experimentation with an Adaptive Search Strategy for Solving a Keyboard Design/Configuration Problem*. PhD thesis, University of Iowa, 1986. 67

[132] David E. Goldberg. *Computer-Aided Gas Pipeline Operation using Genetic Algorithms and Rule Learning*. PhD thesis, University of Michigan, 1983. 67

[133] David E. Goldberg. Simple Genetic Algorithms and the Minimal Deceptive Problem. In Lawrence Davis, editor, *Genetic Algorithms and Simulated Annealing, Research Notes in AI*, pages 74–88. Pitman, London, 1987. 327

[134] David E. Goldberg. Genetic Algorithms and Walsh Functions: Part I, A Gentle Introduction. *Complex Systems*, pages 129–152, 1989. 159, 327

[135] David E. Goldberg. Genetic Algorithms and Walsh Functions: Part II, Deception and Its Analysis. *Complex Systems*, pages 153–171, 1989. 327

[136] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, Reading, MA, 1989. 40, 46, 61, 66, 67

[137] David E. Goldberg. Accounting for Noise in the Sizing of Populations. In L. Darrell Whitley, editor, *Foundations of Genetic Algorithms 2*, pages 127–140. Morgan Kaufmann, Foundations of Genetic Algorithms 2, 1993. 51

[138] David E. Goldberg and Kalyanmoy Deb. A Comparative Analysis of Selection Schemes used in Genetic Algorithms. In Gregory J. E. Rawlins, editor, *Proceedings of the First Workshop on Foundations of Genetic Algorithms*, pages 69–93. Morgan Kaufmann, San Mateo, CA, 1991. 54

[139] David E. Goldberg, Kalyanmoy Deb, and Jeffrey Horn. Massive Multimodality, Deception, and Genetic Algorithms. In Reinhard Männer and Bernard Manderick, editors, *Parallel Problem Solving from Nature, 2*, Amsterdam, 1992. Elsevier Science Publishers, B. V. 349, 350

[140] David E. Goldberg, Kalyanmoy Deb, and Bradley Korb. Don't Worry, Be Messy. In Richard K. Belew and Lashon B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 24–30. Morgan Kaufmann, 1991. 50

[141] David E. Goldberg, Kalyanmoy, and Dirk Thierens. Towards a Better Understanding of Mixing in Genetic Algorithms. *Journal of the Society of Instrumentation and Control Engineers*, 32(1):10–16, 1993. 51

[142] David E. Goldberg and Jon Richardson. Genetic Algorithms with Sharing for Multimodal Function Optimization. In *Proceedings of the Second International Conference on Genetic Algorithms and their Application*, pages 41–49, Mahwah, NJ, USA, 1987. Lawrence Erlbaum Associates, Inc. 152

[143] V. Scott Gordon and L. Darrell Whitley. Serial and Parallel Genetic Algorithms as Function Optimizers. In Stephanie Forrest, editor, *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 177–183, 1993. 69

[144] Martina Gorges-Schleuter. ASPARAGOS - an Asynchronous Parallel Genetic Optimization Strategy. In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 422–427, San Mateo, CA, 1989. Morgan Kaufmann. 154, 156

[145] Martina Gorges-Schleuter. Explicit Parallelism of Genetic Algorithms through Population Structures. In Hans-Paul Schwefel and Reinhard Männer, editors, *PPSN*, volume 496 of *Lecture Notes in Computer Science*, pages 150–159. Springer, 1990. 63, 154

[146] David G. Green, David Newth, and Michael A. Kirley. Connectivity and Catastrophe - Towards a General Theory of Evolution. In Mark A. Bedau et al, editor, *Artificial Life VII: Proceedings of the Seventh International Conference*, pages 153–161. MIT Press, 2000. 160

[147] John J. Grefenstette. GENSIS: A System For Using Genetic Search Procedures. In *In Proceedings of the Conference on Intelligent Systems and Machines*, 1984. 59

[148] John J. Grefenstette. Optimization of Control Parameters for Genetic Algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 16(1):122–128, 1986. 51

[149] John J. Grefenstette. A System for Learning Control Strategies with Genetic Algorithms. In J. David Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 183–190. Morgan Kaufmann, 1989. 158

[150] John J. Grefenstette. Strategy Acquisition with Genetic Algorithms. In Lawrence Davis, editor, *Handbook of Genetic Algorithms*, pages 186–201. Van Nostrand Reinhold, New York, 1991. 46, 67

[151] John J. Grefenstette, Connie Loggia Ramsey, and Alan C. Schultz. Learning Sequential Decision Rules Using Simulation Models and Competition. *Machine Learning*, 5:355–381, 1990. 158

[152] Paul Bryant Grosso. *Computer Simulation of Genetic Adaptation: Parallel Subcomponent Interation in a Multilocus Model*. PhD thesis, University of Michigan, Computer and Communication Sciences Department, 1985. 160

[153] John Guare. *Six Degrees of Separation: A Play*. Vintage Books, New York, 1990. 84

[154] Frank Harary. *Graph Theory*. Perseus, Cambridge, MA, 1995. 87

[155] Georges R. Harik. Finding Multimodal Solutions Using Restricted Tournament Selection. In Larry J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 24–31, San Francisco, CA, 1995. Morgan Kaufmann. 57, 152

[156] Dietrich Hartmann. *Optimierung balkenartiger Zylinderschalen aus Stahlbeton mit elastischem und plastischem Werkstoffverhalten*. PhD thesis, University of Dortmund, 1974. 65

[157] Inman Harvey. Cognition is Not Computation; Evolution is Not Optimisation. In Wulfram Gerstner, Alain Germond, Martin Hasler, and Jean-Daniel Nicoud, editors, *Proceedings of the 7th International Conference on Artificial Neural Networks (ICANN '97)*, pages 685–690, Berlin, 1997. Springer Verlag. 44

[158] Brian Hayes. Graph Theory in Practice: Part I. *American Scientist*, 88(1):9–13, 2000. 87

[159] Brian Hayes. Graph Theory in Practice: Part II. *American Scientist*, 88(2):104–109, 2000. 87

[160] Francisco Herrera and Manuel Lozano. Gradual Distributed Real-coded Genetic Algorithms. *IEEE Transactions on Evolutionary Computation*, 4:43–63, 2000. 69

[161] Francisco Herrera, Manuel Lozano, and Claudio Moraga. Hybrid Distributed Real-coded Genetic Algorithms. In *PPSN V: Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*, pages 603–612, London, UK, 1998. Springer-Verlag. 69

[162] Daniel W. Hillis. Co-Evolving Parasites Improve Simulated Evolution as an Optimization Procedure. *Physica D*, 42:228–234, 1990. 158, 161, 162

[163] Daniel W Hillis. Co-Evolving Parasites Improve Simulated Evolution as an Optimization Procedure. In Christopher G. Langton, C. Taylor, J. Doyne Farmer, and Steen Rasmussen, editors, *Artificial Life II*, volume X, pages 313–324, Redwood City, CA, 1991. Santa Fe Institute Studies in the Sciences of Complexity, Addison-Wesley. 158

[164] John H. Holland. Outline for a Logical Theory of Adaptive Systems. *Journal of the Association of Computing Machinery*, 9(3):297–314, 1962. 66

[165] John H. Holland. Genetic Algorithms and the Optimal Allocations of Trials. *SIAM Journal of Computing*, 2(2):88–105, 1973. 66

[166] John H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, 1 edition, 1975. 45, 55, 66, 154, 160

[167] John H. Holland. Escaping Brittleness: The Possibilities of General-Purpose Learning Algorithms Applied to Parallel Rule-Based Systems. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach: Volume II*, pages 593–623, Los Altos, CA, 1986. Morgan Kaufmann. 158

[168] John H. Holland. Genetic Algorithms. *Scientific American*, 267(1):44–50, 1992. 40, 158

[169] John H. Holland. Echoing Emergence: Objective, Rough Defintions, and Speculations for Echo-Class Models. In G. Cowan, D. Pines, and D. Melzner, editors, *Complexity: Metaphors, Models and Reality*. Addison-Wesley, Reading, MA, 1994. 71, 158

[170] John H. Holland and Judith S. Reitman. Cognitive Systems based on Adaptive Algorithms. In D. A. Waterman and F. Hayes-Roth, editors, *Pattern Directed Interface Systems*, pages 313–329. Academic Press, New York, 1978. 158

[171] Kurt Hornik. The R FAQ, 2007. 96

[172] Phil Husbands. Distributed Coevolutionary Genetic Algorithms for Multi-Criteria and Multi-Constraint Optimisation. In Terence C. Fogarty, editor, *Evolutionary Computing, AISB Workshop Selected Papers*, volume 865 of *LNCS*, pages 150–165. Springer-Verlag, 1994. 159, 162

[173] Phil Husbands, Giles Jermy, Malcolm McIlhagga, and Robert Ives. Two Applications of Genetic Algorithms to Component Design. In Terence C. Fogarty, editor, *Selected Papers from AISB Workshop on Evolutionary Computing*, pages 50–61. Springer-Verlag, 1996. 46

[174] Takashi Ito, Tomoko Chiba, Ritsuko Ozawa, Mikio Yoshida, Masahira Hattori, and Yoshiyuki Sakaki. A Comprehensive Two-Hybrid Analysis to Explore the Yeast Protein Interactome. *Proceedings of the National Academy of Sciences U. S. A.*, 98(8):4569Ű4574, 2001. 127

[175] Sanjay Jain and Sandeep Krishna. Autocatalytic Sets and the Growth of Complexity in an Evolutionary Model. *Physical Review Letters*, 81(25):5684–5687, Dec 1998. 127

[176] Sanjay Jain and Sandeep. Krishna. A Model for the Emergence of Cooperation, Interdependence, and Structure in Evolving Networks. *Proceedings of the National Academy of Sciences U. S. A.*, 98:543–547, January 2001. 127

[177] Hawoong Jeong, Bálint Tombor, Réka Albert, Zoltán N. Oltvai, and Albert-László Barabási. The Large-Scale Organization of Metabolic Networks. *Nature*, 407:651–654, 2000. 127, 129

[178] Ari Juels and Martin Wattenberg. Stochastic Hillclimbing as a Baseline Method for Evaluating Genetic Algorithms. Technical Report CSD-94-834, Computers Science Department, University of California at Berkeley, USA, 18 1995. 45

[179] Hugues Juillé. Incremental Co-evolution of Organisms: A New Approach for Optimization and Discovery of Strategies. In *Proceedings of the Third European Conference on Artifical Life*, pages 246–260, Granada, Spain, 1995. 160

[180] Hugues Juillé and Jordan B. Pollack. Co-evolving Intertwined Spirals. In *Proceedings of the Fifth Annual Conference on Evolutionary Programming, San Diego, CA*, pages 461–468. MIT Press, 1996. 160, 161

[181] Hugues Juillé and Jordan B. Pollack. Coevolving the 'Ideal' Trainer: Application to the Discovery of Cellular Automata Rules. In *Genetic Programming 1998. Proceedings of the Third Annual Conference*, San Francisco, CA, 1998. Morgan Kaufmann. 160, 161

[182] Hugues Juillé and Jordan B. Pollack. Coevolutionary Learning and the Design of Complex Systems. *Advances in Complex Systems*, 2(4):371–394, 2000. 160, 162

[183] Tomihisa Kamada and Satoru Kawai. An Algorithm for Drawing General Undirected Graphs. *Information Processing Letters*, 31(1):7–15, 1989. 120

[184] Laveen Kanal and Vipin Kumar, editors. *Search in Artificial Intelligence*. Springer-Verlag, 1988. 45

[185] Richard M. Karp. Reducibility Among Combinatorial Problems. In Richard E. Miller and James W. Thatcher, editors, *Complexity of Computer Computations*, page 85Ű103. Plenum Press, New York, 1972. 358, 359

[186] Nadav Kashtan, Shalev Itzkovitz, Ron Milo, and Uri Alon. Efficient Sampling Algorithm for Estimating Subgraph Concentrations and Detecting Network Motifs. *Bioinformatics*, 20:1746–58, 2004. 101

[187] Stuart A. Kauffman. Adaptation on Rugged Fitness Landscapes. In Daniel L. Stein, editor, *Lecture in the Sciences of Complexity*, volume 1, pages 527–618. Addison-Wesley, 1989. 53, 161, 349, 354

[188] Evelyn Fox Keller. Revisiting "Scale-free" Networks. *BioEssays*, 27(10):1060–1068, 2005. 98, 109, 122

[189] James Kennedy, Russell C. Eberhart, and Yuhui Shi. *Swarm Intelligence*. Morgan Kaufmann, San Francisco, CA, 2001. 71

[190] Sami Khuri, Thomas Bäck, and Jörg Heitkötter. An Evolutionary Approach to Combinatorial Optimization Problems. In *Proceedings of the 1994 Computer Science Conference (CSC'94)*, pages 66–73. ACM Press, 1994. 349, 358, 359, 361

[191] Beom Jun Kim, Ala Trusina, Petter Minnhagen, and Kim Sneppen. Self Organized Scale-Free Networks from Merging and Regeneration. *European Physical Journal B*, 43(3):669–672, 2005. 121, 122

[192] Jae Yun Kim, Yeongho Kim, and Yeo Keun Kim. An Endosymbiotic Evolutionary Algorithm for Optimization. *Applied Intelligence*, 15:117–130, 2001. 161, 266

[193] Michael A. Kirley. A Coevolutionary Genetic Algorithm for Job Shop Scheduling Problems. In Lakhmi C. Jain, editor, *The Proceedings of The Third International Conference on Knowledge-Based Intelligent Information Engineering Systems*, pages 84–87. IEEE Press, 1999. 160

[194] Michael A. Kirley. MEA: A Metapopulation Evolutionary Algorithm for Multi-objective Optimisation Problems. In *In Proceedings of Congress on Evolutionary Computation (CEC2001), Korea* , pages 949–956. IEEE Press, 2001. 160, 242

[195] Michael A. Kirley. A Cellular Genetic Algorithm with Disturbances: Optimisation Using Dynamic Spatial Interactions. *Journal of Heuristics*, 8(3):321–342, 2002. 242

[196] Michael A. Kirley. *Ecological Algorithms: Investigation of Adaptation, Diversity and Spatial Patterns in Complex Optimisation Problems.* PhD thesis, Charles Stuart University, NSW, Australia, 2002. 159, 160, 242

[197] Michael A. Kirley and David G. Green. An Empirical Investigation of Optimisation in Dynamic Environments Using the Cellular Genetic Algorithm. In L. Darrell Whitley et al., editor, *The Proceedings of Genetic and Evolutionary Computation Conference (GECCO-2000)*, pages 11–18. Morgan Kaufmann, 2000. 160

[198] Michael A. Kirley, David G. Green, and David Newth. Multi-objective Problem, Multi-species Solution: An Application of the Cellular Genetic Algorithm. In M. Mohammadian, editor, *Proceedings of International Conference on Advances in Intelligent Systems: Theory and Applications (ICAIS 2000)*, pages 129–135. IOS press, 2000. 160

[199] Michael A. Kirley, X. Li, and David G. Green. Investigation of a Cellular Genetic Algorithm that Mimics Landscape Ecology. In R. McKay et al., editor, *Simulated Evolution and Learning -SEAL98, volume 1585 Lecture Notes in Artificial Intelligence*, pages 90–97. Springer, 1998. 51, 242

[200] Hajime Kita. A Comparison Study of Self-Adaptation in Evolution Strategies and Real-Coded Genetic Algorithms. *Evolutionary Computation*, 9(2):223–241, 2001. 74

[201] Hajime Kita, Isao Ono, and Shigenobu Kobayashi. Multi-parental Extension of the Unimodal Normal Distribution Crossover for Real-coded Genetic Algorithms. In Vicent W. Porto, editor, *Proceedings of the 1999 Congress on Evolutionary Computation*, pages 1581–1587, Piscataway, New Jersey, 1999. IEEE Press. 74

[202] Sachio Kizu, Hidefumi Sawai, and Susumu Adachi. Parameter-free Genetic Algorithm (PfGA) Using Adaptive Search with Variable-size Local Population and its Extension to Parallel Distributed Processing. In *Trans. On IEICE*, volume J82-D-II, No. 3, pages 1–10, 1999. 10

[203] Sachio Kizu, Hidefumi Sawai, and Tetsuro Endo. Parameter-free Genetic Algorithm: GA without Setting Genetic Parameters. In *Proc. Of the 1997 Int. Symp. On Nonlinear Theory and its Applications*, volume 2, pages 1273–1276, 1997. 10

[204] Jon M. Kleinberg. The Small-World Phenomenon: An Algorithmic Perspective. Technical Report 99-1776, Cornell Computer Science, 1999. 85, 116

[205] Jon M. Kleinberg. Navigation in a Small World. *Nature*, 406:845, 2000. 116

[206] Jon M. Kleinberg. Small-World Phenomena and the Dynamics of Information. In Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 14, pages 431–438. MIT Press, 2001. 85

[207] Konstantin Klemm and Victor M. Eguiluz. Growing Scale-Free Networks with Small World Behavior. *Physical Review E*, 65:057102, 2002. 108

[208] Manfred Kochen, editor. *The Small World*. Ablex, Norwood, NJ, 1989. 84

[209] Charles Korte and Stanley Milgram. Acquaintance Linking Between White and Negro Populations: Application of the Small World Problem. *Journal of Personality and Social Psychology*, 15(101-118):101, 1970. 84

[210] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, 1992. 41, 67

[211] John R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs*. MIT Press, Cambridge, MA., 1994. 67

[212] John R. Koza, Forrest H. Bennett III, David Andre, and Martin A. Keane. Genetic Programming III: Darwinian Invention and Problem Solving [Book Review]. *IEEE Transactions on Evolutionary Computation*, 3(3):251–253, 1999. 67

[213] Paul L. Krapivsky and Sidney Redner. A Statistical Physics Perspective on Web Growth. *Computer Networks*, 39:261–276, 2002. 119

[214] William B. Langdon. *Data Structures and Genetic Programming: Genetic Programming + Data Structures = Automatic Programming!* Kluwer Academic Publishers, Boston, 1998. 67

[215] Vito Latora and Massimo Marchiori. Efficient Behavior of Small-world Networks. *Physical Review Letters*, 87(19):198701, 2001. 100, 103, 105, 107

[216] Vito Latora and Massimo Marchiori. Economic Small-world Behavior in Weighted Networks. *European Physical Journal B*, 32:249–263, 2003. 103, 104, 107

[217] Vito Latora and Massimo Marchiori. The Architecture of Complex Systems. In *Interdisciplinary Applications of Ideas from Nonextensive Statistical Mechanics and Thermodynamics*. Oxford University Press, Santa Fe Institute for Studies of Complexity, 2003. 83, 103, 104, 107

[218] David Levine. *A Parallel Genetic Algorithm for the Set Partitioning Problem*. PhD thesis, Argonne National Laboratory, Illinois Institute of Technology, Illinois, USA, 1994. 46

[219] Fernando G. Lobo, Cludio F. Lima, and Zbigniew Michalewicz, editors. *Parameter Setting in Evolutionary Algorithms*. Studies in Computational Intelligence. Springer-Verlag, Berlin, 2007. 10, 267

[220] Charles Lyell. *Principles of Geology, Being an Attempt to Explain the Former Changes of the Earth's Surface, by Reference to Causes Now in Operation*. John Murray, London, 1830-1933. 3

[221] Florence Jessie MacWilliams and Neil J. A. Sloane. *The Theory of Error-correcting Codes*. North-Holland, 1977. 349, 360

[222] Samir W. Mahfoud. Crowding and Preselection Revisited. In Reinhard Männer and Bernard Manderick, editors, *Parallel problem solving from nature 2*, pages 27–36, Amsterdam, 1992. North-Holland. 153

[223] Kim-Fung Man, Kit Sang Tang, and Sam Kwong. *Genetic Algorithms: Concepts and Designs*. Springer-Verlag, London, UK, 1999. 47

[224] Bernard Manderick and Piet Spiessens. Fine-grained Parallel Genetic Algorithms. In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 428–433, San Mateo, CA, 1989. Morgan Kaufmann. 51, 63, 154

[225] Shmoolik Mangan and Uri Alon. Structure and Function of the Feed-Forward Loop Network Motif. *Proceedings of the National Academy of Sciences U. S. A.*, 100:11980–11985, 2003. 101

[226] Henry B. Mann and D. R. Whitney. On a Test Whether One of Two Random Variables is Stochastically Larger than the Other. *Annals of Mathematical Statistics*, 18:50–60, 1947. 166

[227] Sergei Maslov and Kim Sneppen. Specificity and Stability in Topology of Protein Networks. *Science*, 296:910–913, 2002. 127, 128

[228] Ernst W. Mayr. *Animal Species and Evolution*. Belknap, Cambridge, MA, 1963. 48

[229] Ole J. Mengshoel and David E. Goldberg. Probabilistic Crowding: Deterministic Crowding with Probabilistic Replacement. IlliGAL Report 99005, University of Illinois, 1999. 57, 152

[230] Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, Berlin, Germany, 1992. 343

[231] Kaisa Miettinen, Pekka Neittaanmäki, Marko M. Mäkelä, and Jacques Périaux, editors. *Evolutionary Algorithms in Engineering and Computer Science*. John Wiley and Sons, New York, 1999. 46

[232] Stanley Milgram. The Small World Problem. *Psychology Today*, 2:60–67, 1967. 11, 84, 126

[233] Brad L. Miller and Michael J. Shaw. Genetic Algorithms with Dynamic Niche Sharing for Multimodal Function Optimization. In *International Conference on Evolutionary Computation*, pages 786–791, 1996. 57, 152

[234] Ron Milo, Shalev Itzkovitz, Nadav Kashtan, Reuven Levitt, Shai S. Shen-Orr, Inbal Ayzenshtat, Michal Sheffer, and Uri Alon. Superfamilies of Designed and Evolved Networks. *Science*, 303:1538–42, 2004. 101

[235] Ron Milo, Shai S. Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. Network Motifs: Simple Building Blocks of Complex Networks. *Science*, 298:824–827, 2002. 101, 102, 128

[236] David G. Mitchell, Bart Selman, and Hector J. Levesque. Hard and Easy Distributions for SAT Problems. In Paul Rosenbloom and Peter Szolovits, editors, *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 459–465, Menlo Park, California, 1992. AAAI Press. 352

[237] Melanie Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, 1998. 46, 47, 67

[238] Melanie Mitchell and Stephanie Forrest. Fitness Landscapes: Royal Road Functions. In Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors, *Handbook of Evolutionary Computation*. Institute of Physics Publishing and Oxford University, 1997. 326

[239] Melanie Mitchell, Stephanie Forrest, and John H. Holland. The Royal Road for Genetic Algorithms: Fitness Landscapes and GA Performance. In Francisco J. Varela and Paul Bourgine, editors, *Proceedings of the First European Conference on Artificial Intelligence*, Cambridge, MA, 1992. MIT Press. 326

[240] Melanie Mitchell, Michale D. Thomure, and Nathan L. Williams. The Role of Space in the Success of Coevolutionary Learning. In *Proceedings of Artificial Life X: Tenth Annual Conference on the Simulation and Synthesis of Living Systems*, Cambridge, MA, 2006. MIT Press. 162

[241] M. Molloy and B. Reed. A critical point for random graphs with a given degree sequence. *Random Structures and Algorithms*, 6:161–179, 1995. 109, 114

[242] M. Molloy and B. Reed. The size of the giant component of a random graph with a given degree sequence. *Combinatorics, Probability and Computing*, 7:295Ú305, 1998. 114

[243] P.A.P. Moran. The theory of some genetical effects of population sub-division. *Australian Journal of Biological Sciences*, 12:109–116, 1959. 154

[244] David E. Moriarty and Risto Miikkulainen. Forming Neural Networks Through Efficient and Adaptive Coevolution. *Evolutionary Computation*, 5:373–399, 1997. 160

[245] Heinz Mühlenbein. Parallel Genetic Algorithms, Population Genetics and Combinatorial Optimization. In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 416–421, San Mateo, CA, 1989. Morgan Kaufmann. 63, 154

[246] Heinz Mühlenbein and Dirk Schlierkamp-Voosen. Predictive Models for the Breeder Genetic Algorithm. *Evolutionary Computation*, 1(1):39–80, 1993. 337, 339

[247] Heinz Mühlenbein, M. Schomisch, and Joachim Born. The Parallel Genetic Algorithm as Function Optimizer. *Parallel Computing*, 17(6-7):619–632, 1991. 154, 337

[248] Andrew Gerard W. Murray. *Micro-net: The Parallel Path Artificial Neuron*. PhD thesis, Swinburne University of Technology, Melbourne, Victoria, Australia, 2006. 95

[249] Gerard Murray and Tim Hendtlass. Enhanced Artificial Neurons for Network Applications. In *IEA/AIE '01: Proceedings of the 14th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, pages 281–289, London, UK, 2001. Springer-Verlag. 95

[250] Gerard Murray, Tim Hendtlass, and John R. Podlena. The Parallel Path Artificial Micronet. In Ibrahim F. Imam, Yves Kodratoff, Ayman El-Dessouki, and Moonis Ali, editors, *Multiple Approaches to Intelligent Systems, 12th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE-99, Cairo, Egypt, May 31 - June 3, 1999*, volume 1611:1 of *Lecture Notes in Computer Science*, pages 111–117. Springer, 1999. 95

[251] Tomoharu Nakashima, Takanobu Ariyama, and Hisao Ishibuchi. Combining Multiple Cellular Genetic Algorithms for Efficient Search. In Lipo Wang, Kay Chen Tan, Takeshi Furuhashi, Jong-Hwan Kim, and Xin Yao, editors, *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL'02)*, pages 712–716, Singapore, 2002. 154

[252] V. Nannen, S. K. Smit, and A. E. Eiben. Costs and benefits of tuning parameters of evolutionary algorithms. In G. Rudolph, Th. Jansen, S.M. Lucas, C. Poloni, and N. Beume, editors, *Parallel Problem Solving from Nature Ű PPSN X*, volume 5199 of *Lecture Notes in Computer Science*, page 528Ű538, Berlin / Heidelberg, 2008. Springer. 10

[253] Mark E. J. Newman. Models of the Small World: A Review. *Journal of Statistical Physics*, 101:819–841, 2000. 87

[254] Mark E. J. Newman. Assortative Mixing in Networks. *Physical Review Letters*, 89:208701, 2002. 99

[255] Mark E. J. Newman. The Structure and Function of Networks. *Computer Physics Communications*, 147:40–45, 2002. 84

[256] Mark E. J. Newman. Mixing Patterns in Networks. *Physical Review E*, 67:026126, 2003. 99

[257] Mark E. J. Newman. The Structure and Function of Complex Networks. *SIAM Review*, 45(2):167–256, 2003. 12, 81, 82, 83, 85, 87, 98, 100, 103, 108, 114, 117, 118, 119, 124, 127

[258] Mark E. J. Newman. Power Laws, Pareto Distributions and Zipf's Law. *Contemporary Physics*, 46:323–351, 2005. 87

[259] Mark E. J. Newman, Albert-László Barabási, and Duncan J. Watts. *The Structure and Dynamics of Networks*. Princeton University Press, 2006. 11, 81, 86, 87, 97, 126

[260] Stefano Nolfi and Dario Floreano. Coevolving Predator and Prey Robots: Do Arms Races Arise in Artificial Evolution. *Artificial Life*, 4:311–335, 1998. 161

[261] Stefano Nolfi and Dario Floreano. How Co-Evolution can Enhance the Adaptation Power of Artificial Evolution: Implications for Evolutionary Robotics. In P. Husbands and Jean-Arcady Meyer, editors, *Proceedings of the First European Workshop on Evolutionary Robotics*, volume 468 of *LNCS*, Berlin, 1998. Springer. 161

[262] Stefano Nolfi and Dario Floreano. *Evolutionary Robotics: The Biology, Intelligence and Technology of Self-Organizing Machines*. MIT Press, Cambridge, MA, 2000. 161

[263] Mariusz Nowostawski and Riccardo Poli. Parallel Genetic Algorithm Taxonomy. *Knowledge-Based Intelligent Information Engineering Systems*, pages 88–92, 1999. 63, 153

[264] Eugene Odum, Richard Brewer, and Gary W Barret. *Fundamentals of Ecology*. Brooks Cole, 5 edition, 2004. 18

[265] Eugene P. Odum. *Fundamentals of Ecology*. W. B. Saunders Company, Philadelphia, 1st edition, 1971. 18

[266] Travis E. Oliphant. Python for Scientific Computing. *Computing in Science & Engineering*, 9(3):10–20, May/June 2007. 462

[267] Jukka-Pekka Onnela. *Complex Networks in the Study of Financial and Social Systems*. PhD thesis, Helsinki University of Technology, Department of Electrical and Communications Engineering, 2006. 121

[268] Ludo Pagie and Paulien Hogeweg. Evolutionary Consequences of Coevolving Targets. *Evolutionary Computation*, 5(4):401–418, 1997. 161, 162

[269] Ludo Pagie and Paulien Hogeweg. Information Integration and Red Queen Dynamics in Coevolutionary Optimization. In *Proceedings CEC 2000*, pages 1260–1267, 2000. 162

[270] Ludo Pagie and Melanie Mitchell. A Comparison of Evolutionary and Coevolutionary Search. *International Journal of Computational Intelligence and Applications*, 2(1):53–69, 2002. 161

[271] Jan Paredis. The Symbiotic Evolution of Solutions and their Representations. In Larry J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 359–365. Morgan Kaufmann, 1995. 159

[272] Jan Paredis. Coevolving Cellular Automata: Be Aware the Red Queen! In Thomas Bäck, editor, *Proceedings ICGA VII*, pages 393–400, 1997. 161, 162

[273] Romualdo Pastor-Satorras, Alexei Vázquez, and Alessandro Vespignani. Dynamical and Correlation Properties of the Internet. *Physical Review Letters*, 87(25):258701, Nov 2001. 99

[274] Raymond C. Paton. *Computing With Biological Metaphors*. Chapman & Hall, 1994. 40, 48, 65

[275] Raymond C. Paton. Enhancing Evolutionary Computation using Analogues of Biological Mechanisms. In *Evolutionary Computation, AISB Workshop*, pages 51–64. Springer-Verlag, 1994. 41

[276] Jordan B. Pollack, A. D. Blair, and M. Land. Coevolution of a Backgammon Player. In Christopher G. Langton and Katsunori Shimohara, editors, *Proceedings of the Fifth Artificial Life Conference*, pages 92–98, Cambridge, MA, 1997. MIT Press. 158, 162

[277] Ithiel de Sola Pool and Manfred Kochen. Contacts and Influence. *Social Networks*, 1:5Ű51, 1978. 84

[278] Mitchell A. Potter. *The Design and Analysis of a Computational Model of Cooperative CoEvolution*. PhD thesis, George Mason University, Fairfax, Virginia, 1997. 52, 159, 160

[279] Mitchell A. Potter and Kenneth A. De Jong. Evolving Neural Networks with Collaborative Species. In *Proceedings of the 1995 Summer Computer Simulation Conference, Ottawa, Ontario, Canada*, pages 340–345. The Society for Computer Simulation, 1995. 160

[280] Mitchell A. Potter and Kenneth A. De Jong. The Coevolution of Antibodies for Concept Learning. In *Proceedings of the Fifth International Conference on Parallel Problem Sovling From Nature, Amsterdam, The Netherlands*, pages 530–539. Springer-Verlad, 1998. 160

[281] Mitchell A. Potter and Kenneth A. De Jong. Cooperative Coevolution: An Architecture for Evolving Coadapted Subcomponents. *Evolutionary Computation*, 8(1):1–29, 2000. 52, 159

[282] Derek John de Solla Price. Networks of Scientific Papers. *Science*, 149:510–515, 1965. 83, 109, 117

[283] Derek John de Solla Price. A General Theory of Bibliometric and Other Cumulative Advantage Processes. *Journal of the American Society for Information Science*, 27:292–306, 1976. 117

[284] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2007. ISBN 3-900051-07-0. 96

[285] Nicholas J. Radcliffe and Patrick D. Surry. Co-operation through Hierarchical Competition in Genetic Data Mining. Technical Report EPCC-TR94-09, Edinburgh Parallel Computing Centre„ University of Edinburgh, Scotland, 1994. 46

[286] Erzsebet Ravasz, A. L. Somera, D. A. Mongru, Zoltán N. Oltvai, and Albert-László Barabási. Hierarchical Organization of Modularity in Metabolic Networks. *Science*, 297:1551, 2002. 113, 122

[287] Thomas S. Ray. An Evolutionary Approach to Synthetic Biology: Zen and the Art of Creating Life. *Artificial Life*, 1(1/2):195–226, 1994. 71, 158

[288] Ronald C. Read and Robin J. Wilson. *An Altas of Graphs*. Oxford University Press, 1998. 102

[289] Ingo Rechenberg. *Evolutionstrategie: Optimierung Technisher Systeme nach Prinzipien der Biologischen Evolution*. Frommann-Holzboog Verlag, Stuttgart, 1973. 65, 330

[290] Ingo Rechenberg. Cybernetic Solution Path of an Experimental Problem. Royal Aircraft Establishment, Library Translation 1122, 1965. In David B. Fogel, editor, *Evolutionary Computation – The fossil record*, chapter 6, pages 297–309. IEEE Press, 1998. 65

[291] Jon Reed, Robert Toombs, and Nils Aall Barricelli. Simulation of Biological Evolution and Machine Learning. *Journal of Theoretical Biology.*, 17:319–342, 1967. 40, 158

[292] Edward M. Reingold and John S. Tilford. Tidier Drawing of Trees. *IEEE Transactions on Software Engineering*, 7:223–228, 1981. 95

[293] Robert G. Reynolds. Cultural Algorithms: Theory and Application. In David Corne, Marco Dorigo, and Fred Glover, editors, *New Ideas in Optimisation*, page 367. McGraw-Hill, 1999. 40, 71

[294] Howard H. Rosenbrock. An Automatic Method for Finding the Greatest or Least Value of a Function. *Computer Journal*, 3:175–184, 1960. 335

[295] Christopher D. Rosin and Richard K. Belew. Methods for Competitive Co-evolution: Finding Opponents Worth Beating. In Larry J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 373–380, San Francisco, CA, 1995. Morgan Kaufmann. 158

[296] Christopher D. Rosin and Richard K. Belew. New Methods for Competitive Coevolution. *Evolutionary Computation*, 5(1):1–19, 1997. 158, 161, 162

[297] Martin Rosvall. *Information Horizons in a Complex World*. PhD thesis, Department of Physics, Umeå University, 2006. 121, 126

[298] Martin Rosvall and Kim Sneppen. Modeling Dynamics of Information Networks. *Physical Review Letters*, 91(17):178701, Oct 2003. 121, 126

[299] Jayshree Sarma and Kenneth A. DeJong. An Analysis of Local Selection Algorithms in a Spatially Structured Evolutionary Algorithm. In Thomas Bäck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 181–187. Morgan Kaufmann, 1997. 69, 156

[300] Jayshree Sarma and Kenneth A. De Jong. An Analysis of the Effects of Neighborhood Size and Shape on Local Selection Algorithms. In *Proceedings of the Fourth PPSN*, volume 1141 of *LNCS*, pages 236–244. Springer-Verlag, 1996. 69, 156

[301] Jayshree A. Sarma. *An Analysis of Decentralized and Spatially Distributed Genetic Algorithms*. PhD thesis, George Mason University, 1998. 156

[302] Hiroshi Satoh, Masayuki Yamamura, and Shigenobu Kobayashi. Minimal Generation Gap Model for GAs Considering Both, Exploration and Exploitation. In *Proceedings of IIZUKA: Methodologies for the Conception, Design, and Application of Intelligent Systems*, pages 494–497, Singapore, 1996. World Scientific. 74

[303] Hidefumi Sawai and Sachio Kizu. Parameter-free Genetic Algorithm Inspired by "Disparity Theory of Evolution". In *Proceedings of the 1997 International Conference on Parallel Problem Solving from Nature*, pages 702–711, 1998. 10

[304] J. David Schaffer and Larry J. Eshelman. On Crossover as an Evolutionary Viable Strategy. In Richard K. Belew and Lashon B. Booker, editors, *Proceedings of the 4th International Conference on Genetic Algorithms*, pages 61–68. Morgan Kaufmann, 1991. 325

[305] J. David Schaffer and Larry J. Eshelman. Combinatorial Optimization by Genetic Algorithms: The Value of the Genotype/Phenotype Distinction. In *Proceedings of the Conference on Applied Decision Technologies (ADT'95). Volume 1: Computational Learning and Probabilistic Reasoning* , pages 29–40, Uxbridge, UK, 1995. Unicom Seminars. 46

[306] Dirk Schlierkamp-Voosen and Heinz Mühlenbein. Strategy Adaptation by Competing Subpopulations. In R. Männer Yuval Davidor, Hans-Paul Schewefel, editor, *Proceedings of the 3rd Conference on Parallel Problem Solving from Nature*, number 866 in Lecture Notes in Computer Science, pages 199–209, Berlin, Heidelberg, New York, 1994. Springer. 255

[307] Birgitt Schönfisch and André de Roos. Synchronous and Asynchronous Updating in Cellular Automata. *BioSystems*, 51:123–143, 1999. 218

[308] Hans-Paul Schwefel. *Evolutionsstrategie und numerische Optimierung*. PhD thesis, Technical University of Berlin, Berlin, Germany, 1975. 65

[309] Hans-Paul Schwefel. *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*. Basel and Stuttgart, Birkhäuser, 1977. 65

[310] Hans-Paul Schwefel. *Numerical Optimization of Computer Models*. John Wiley & Sons, New York, 1981. 65, 188, 342

[311] Hans-Paul Schwefel. *Evolution and Optimum Seeking*. John Wiley & Sons, New York, 1995. 332

[312] Hans-Paul Schwefel. Advantages (and Disadvantages) of Evolutionary Computation Over Other Approaches. In Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors, *Handbook of Evolutionary Computation*, page A1.3. Institute of Physics Publishing Ltd and Oxford University Press, UK and USA, release 97/1 edition, 1997. 65

[313] Anthony V. Sebald and Jennifer Schlenzig. Minimax Design of Neural Net Controllers for Highly Uncertain Plants. *IEEE Transactions of Neural Networks*, 5(1):73–82, 1996. 66

[314] Jonathan L. Shapiro. Does Data-Model Co-evolution Improve Generalization Performance of Evolving Learners? In Agoston E. Eiben, Thomas Bäck, Marc Schoenauerr, and Hans-Paul Schwefel, editors, *Parallel Problem Sovling from Nature (PPSN) V*, volume 1498 of *LNCS*, pages 540–549, 1998. 162

[315] Shai S. Shen-Orr, Ron Milo, Shmoolik Mangan, and Uri Alon. Network Motifs in the Transcriptional Regulation Network of Escherichia coli. *Nature Genetics*, 31:64–68, 2002. 101, 128

[316] Herbert A. Simon. On a Class of Skew Distribution Functions. *Biometrika*, 42(3/4):425–440, 1955. 83, 117

[317] Karl Sims. Evolving 3D Morphology and Behavior by Competition. In Rodney A. Brooks and Pattie Maes, editors, *Proceedings Artifical Life IV*, pages 28–39, 1994. 161

[318] Moshe Sipper, Marco Tomassini, and Mathieu S. Capcarrère. Evolving Asynchronous and Scalable Non-uniform Cellular Automata. In G. D. Smith, N. C. Steele, and R. F. Albrecht, editors, *Proceedings of International Conference on Arti?cial Neural Networks and Genetic Algorithms (ICANNGA97)*, pages 67–71, Vienna, 1997. Springer-Verlag. 218

[319] Zbigniew M. Skolicki. *An Analysis of Island Models in Evolutionary Computation*. PhD thesis, George Mason University, Fairfax, VA, USA, 2007. 154, 157

[320] Zbigniew M. Skolicki and Kenneth A. De Jong. Improving Evolutionary Algorithms with Multi-representation Island Models. In Xin Yao, Edmund K. Burke, José Antonio Lozano, Jim Smith, Juan J. Merelo Guervós, John A. Bullinaria, Jonathan E. Rowe, Peter Tiňo, Ata Kabán, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature - PPSN VIII, 8th International Conference*, number 3242 in Lecture Notes in Computer Science, pages 420–429. Springer, 2004. 154

[321] Selmar K. Smit and Agoston E. Eiben. Comparing parameter tuning methods for evolutionary algorithms. In *CEC'09: Proceedings of the Eleventh conference on Congress on Evolutionary Computation*, pages 399–406, Piscataway, NJ, USA, 2009. IEEE Press. 10

[322] Robert E. Smith and Jim Smith. An Examination of Tunable, Random Landscapes. In Worthy N. Martin and William M. Spears, editors, *Foundations of Genetic Algorithms 6*, pages 47–67. Morgan Kaufmann, San Francisco, 2001. 53, 354

[323] Stephen F. Smith. Flexible Learning of Problem Solving Heuristics Through Adaptive Search. In Alan Bundy, editor, *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 422–425. Morgan Kaufmann, 1983. 158

[324] Artem Sokolov and L. Darrell Whitley. Unbiased Tournament Selection. In *Genetic and Evolutionary Computation Conference (GECCO 2005)*. ACM Press, 2005. 56

[325] Ray Solomonoff and Anatol Rapoport. Connectivity of Random Nets. *Bulletin of Mathematical Biophysics*, 14:107–117, 1951. 83

[326] William M. Spears, Kenneth A. De Jong, Thomas Bäck, David B. Fogel, and Hugo de Garis. An Overview of Evolutionary Computation. In *Proceedings of European Conference on Machine Learning*, pages 442–459, 1993. 66, 198

[327] Piet Spiessens and Bernard Manderick. A Genetic Algorithm for Massively Parallel Computers. In Rolf Eckmiller, Gert Hartmann, and Georg Hauske, editors, *Parallel Processing in Neural Systems and Computers*, page 31Ű36, Amsterdam, 1990. North Holland. 154

[328] Piet Spiessens and Bernard Manderick. A Massively Parallel Genetic Algorithm: Implementation and First Analysis. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 279–285, La Jolla, CA, 1991. Morgan Kaufmann. 154

[329] Douglas R. Stinson. *An Introduction to the Design and Analysis of Algorithms*. The Charles Babbage Research Center, Winnipeg, Manitoba, Canada, second edition, 1987. 361

[330] Rainer Storn and Kenneth Price. Differential Evolution - A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces. Technical Report TR-95-012, Berkeley, Berkeley, CA, 1995. 65, 70

[331] Rainer Storn and Kenneth Price. Differential Evolution: A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces. *Journal of Global Optimization*, 11:341–359, 1997. 65, 70

[332] Steven H. Strogatz. Exploring Complex Networks. *Nature*, 410:268–276, 2001. 87

[333] Keith Sullivan, Sean Luke, Curt Larock, Sean Cier, and Steven Armentrout. Opportunistic Evolution: Efficient Evolutionary Computation on Large-scale Computational Grids. In *GECCO '08: Proceedings of the 2008 GECCO conference companion on Genetic and evolutionary computation*, pages 2227–2232, New York, NY, USA, 2008. ACM. 156, 255

[334] Gilbert Syswerda. A Study of Reproduction in Generational and Steady-state Genetic Algorithms. In Gregory J. E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 94–101. Morgan Kaufmann, San Mateo, CA, 1991. 63

[335] Kazuhiro Takemoto and Chikoo Oosawa. Evolving Networks by Merging Cliques. *Physical Review E*, 72:046116, 2005. 122

[336] Roberto Tamassia. Graph Drawing. In Jörg Rüdiger Sack and Jorge Urrutia, editors, *Handbook of Computational Geometry*, chapter 21, pages 937–971. North-Holland, Amsterdam, Netherlands, 2000. 95

[337] Reiko Tanese. Distributed Genetic Algorithms. In J. David Schaffer, editor, *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 434–439, 1989. 63, 68

[338] Arthur George Tansley. The Use and Abuse of Vegetational Concepts and Terms. *Ecology*, 16:284–307, 1935. 18

[339] Dirk Thierens and David E. Goldberg. Mixing in Genetic Algorithms. In *Proceedings of the Fifth International Conference on Genetic Algorithms.*, pages 38–45, Urbana-Champaign, 1993. Morgan Kaufmann. 51

[340] Adrian Thompson. Evolving Fault Tolerant Systems. In *Proceedings of the First IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA'95)*, number 414 in IEE Conf. Publication, pages 524–529, 1995. 46

[341] Adrian Thompson. Evolutionary Techniques for Fault Tolerance. In *Proceedings of the UKACC International Conference on Control*, pages 693–698, 1996. 46

[342] Aimo Törn and Antanas Zilinskas. Global Optimization. In *Lecture Notes in Computer Science 350*, Berlin Heidelberg, 1989. Springer-Verlag. 337, 339

[343] Shigeyoshi Tsutsui and Yoshiji Fujimoto. Forking Genetic Algorithm with Blocking and Shrinking Modes (fGA). In Stephanie Forrest, editor, *5th International Conference of Genetic Algorithms (ICGA)*, pages 206–215, San Mateo, CA, 1993. Morgan Kaufmann. 344

[344] Shigeyoshi Tsutsui, Masayuki Yamamura, and Ashish Ghosh. Forking GAs: GAs with Search Space Division Schemes. *Evolutionary Computation*, 5(1):61–80, 1997. 344

[345] Shigeyoshi Tsutsui, Masayuki Yamamura, and Takahide Higuchi. Multi-parent Recombination with Simplex Crossover in Real Coded Genetic Algorithms. In *Proceedings of the 1999 Genetic and Evolutionary Computation Conference (GECCO-99)*, pages 657–664, 1999. 74

[346] Leigh Van Valen. A New Evolutionary Law. *Evolutionary Theory*, 1:1–30, 1973. 161

[347] Rajesh Vasa, Jean-Guy Schneider, Oscar Nierstrasz, and Clinton Woodward. On the Resilience of Classes to Change. *ECEASST*, 8, 2007. 126

[348] Rajesh Vasa, Jean-Guy Schneider, Clinton Woodward, and Andrew Cain. Detecting Structural Changes in Object Oriented Software Systems. In *2005 International Symposium on Empirical Software Engineering (ISESE 2005), 17-18 November 2005, Noosa Heads, Australia*, pages 479–486. IEEE, 2005. 126

[349] Andreas Wagner. Energy Constraints on the Evolution of Gene Expression. *Molecular Biology and Evolution*, 22(6):1365–1374, 2005. 129

[350] Andreas Wagner and David A. Fell. The Small World Inside Large Metabolic Networks. Tech. Rep. 00-07-041, Santa Fe Institute, 2000. 127, 129

[351] Andreas Wagner and David A. Fell. The Small World Inside Large Metabolic Networks. *Proceedings Biological Sciences / The Royal Society*, 268(1478):1803–1810, 2001. 127, 129

[352] M. Mitchell Waldrop. *Complexity: The Emerging Science at the Edge of Order and Chaos*. Simon and Schuster, New York, 1992. 12

[353] Toby Walsh. Search in a Small World. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI'99)*, pages 1172–1177. Morgan Kaufmann, 1999. 105

[354] Jean-Paul Watson. A Performance Assessment of Modern Niching Methods for Parameter Optimization Problems. In Wolfgang Banzhaf, Jason Daida, Agoston E. Eiben, Max H. Garzon, Vasant Honavar, Mark Jakiela, and Robert E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 1, pages 702–709, Orlando, Florida, USA, 13–17 1999. Morgan Kaufmann. 57, 152

[355] Richard A. Watson. *Compositional Evolution: Interdisciplinary Investigations in Evolvability, Modularity, and Symbiosis*. PhD thesis, Brandeis University, Waltham, MA, USA, 2002. 157, 160

[356] Richard A. Watson, Gregory S. Hornby, and Jordan B. Pollack. Modeling Building-block Interdependency. In Agoston E. Eiben, T. Bäck, Marc Schoenauer, and Hans-Paul Schwefel, editors, *Proceedings of Parallel Problem Solving from Nature*, pages 97–106. Springer, 1998. 160

[357] Richard A. Watson and Jordan B. Pollack. How Symbiosis Can Guide Evolution. In Dario Floreano, Jean-Daniel Nicoud, and Francesco Mondada, editors, *Fifth European Conference on Artificial Life*. Springer, 1999. 160

[358] Duncan J. Watts. *Small Worlds: The Dynamics of Networks between Order and Randomness*. Princeton University Press, Princeton University Press, 1999. 85

[359] Duncan J. Watts. *Six Degrees: The Science of a Connected Age*. W. W. Norton and Co., New York, 2003. 81, 84, 87

[360] Duncan J. Watts and Steven H. Strogatz. Collective Dynamics of 'Small-World' Networks. *Nature*, 393:440–442, 1998. 11, 84, 100, 102, 105, 108, 109, 114, 115, 116, 117, 467, 469

[361] Joshua S. Weitz, Philip N. Benfey, and Ned S. Wingreen. Evolution, Interactions, and Biological Networks. *PLoS Biol.*, 5(1):doi:10.1371/journal.pbio.0050011, 2007. 81

[362] Justin Werfel, Melanie Mitchell, and James P. Crutchfield. Resource Sharing and Coevolution in Evolving Cellular Automata. *IEEE Transactions on Evolutionary Computation*, 4(4):388–393, 2000. 161

[363] Sebastian Wernicke and Florian Rasche. FANMOD: A Tool for Fast Network Motif Detection. *Bioinformatics*, 22(9):1152–1153, 2006. 102

[364] L. Darrell Whitley. The GENITOR Algorithm and Selection Pressure: Why Rank-based Allocation of Reproductive Trials is Best. In J. David Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 116–123. Morgan Kaufmann, 1989. 44, 56

[365] L. Darrell Whitley. Fundamental Principles of Deception in Genetic Search. In G. J. E Rawlings, editor, *Foundations of Genetic Algorithms*. Morgan Kaufmann, San Mateo, 1991. 328

[366] L. Darrell Whitley, Soraya B. Rana, John Dzubera, and Keith E. Mathias. Building Better Test Functions. In Larry J. Eshelman, editor, *International Conference on Genetic Algorithms*. Morgan Kaufmann, 1995. 339

[367] L. Darrell Whitley, Soraya B. Rana, and Robert B. Heckendorn. The Island Genetic Algorithm: On Separability, Population Size and Convergence. *Journal of Computing and Information Technology*, 2(1):33–47, 1999. 154

[368] Frank Wilcoxon. Individual Comparisons by Ranking Methods. *Biometrics*, 1:80–83, 1945. 166

[369] Nathan L. Williams and Melanie Mitchell. Investigating the Success of Spatial Coevolutionary Learning. In Hans-Georg Beyer et al., editor, *Proceedings of the 2005 Genetic and Evolutionary Computation Conference, GECCO-2005*, pages 523–530. New York: ACM Press, 2005. 162

[370] David H. Wolpert and William G. Macready. No Free Lunch Theorems for Search. Technical Report Technical Report SFI-TR-95-02-010, Santa Fe Institute, 1995. 5, 46

[371] Clinton Woodward and Tim Hendtlass. Dynamic Trait Expression for Multiploid Individuals of Evolutionary Algorithms. In Laszlo Monostori, József Váncza, and Moonis Ali, editors, *Engineering of Intelligent Systems, 14th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE 2001, Budapest, Hungary, June 4-7, 2001*, volume 2070 of *Lecture Notes in Computer Science*, pages 374–382. Springer, 2001. 50

[372] Clinton Woodward and Gerard Murray. Visualising the Internal Components of Networks. In Paul W. H. Chung, Chris J. Hinde, and Moonis Ali, editors, *Developments in Applied Artificial Intelligence, 16th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE 2003, Laughborough, UK, June 23-26, 2003*, volume 2718 of *Lecture Notes in Computer Science*, pages 555–564. Springer, 2003. 95

[373] Sewall Wright. The Roles of Mutation, Inbreeding, Crossbreeding, and Selection in Evolution. In *Proceedings of the Sixth International Congress on Genetics*, volume 1, pages 356–366, 1932. 40

[374] Sewall Wright. Stochastic processes in evolution. In John Gurland, editor, *Stochastic Models in Medicine and Biology*, pages 199–241, MAdison, WI, 1964. University of Wisconsin Press. 40

[375] Sewall Wright. *Evolution and the Genetics of Populations*. Chicago University Press, Chicago, USA, 1968. 40

[376] Stefan Wuchty. Scale-Free Behavior in Protein Domain Networks. *Molecular Biology and Evolution*, 18(9):1694–1702, 2001. 127, 128

[377] Takeshi Yamada and Ryohei Nakano. A Genetic Algorithm with Multi-step Crossover for Job-shop Scheduling Problems. In *1st IEE/IEEE International Conference on Genetic ALgorithms in Engineering Systems (GALESIA '95)*, pages 146–151, 1995. 46

[378] George Kingsley Zipf. *Human Behavior and The Principles of Least Effort*. Addison Wesley, Cambridge, 1949. 117

Appendix A

Glossaries

A.1 Ecology, Ecosystems and Evolution

Abiotic The non-living factors (material) of an environment. 19, 22

Adaptation A process of change (adjustment) to take advantage of environmental factors. Evolution. 29

Allele One of the variant forms of a gene at a particular locus (chromosome location). 28, 29

Assortative Mating Sexually reproducing organisms tending to mate with individuals like themselves. A possible mechanism of speciation. 31

Biome Regional (large scale) ecosystem, composed of similar types of dynamic communities. 17

Biosphere Earth's largest ecosystem. The zone of air, land and water at the surface of the earth that is occupied by organisms. 17

Biotic Living organisms (the biological aspects of an environment). 19

Carrying Capacity A measure of equilibrium between births and deaths at the maximum sustainable population size. 24, 25

Cell The structural and functional unit of organisms, and the smallest unit classified as living. 17

Change Typically, the alteration of environment (conditions). 22

- **Cyclic C.** Periodic (rhythmic) changes, such as seasonal variations, day/night cycles and lunar (tidal) influences. 22
- **Directional C.** A sustained direction of change, typically over a long period of time, such as glaciers, erosion, siltation, salination. 22
- **Erratic C.** Change without rhythm or periodic nature, such as earthquakes, tsunami, volcanoes, fires, land-slides, cyclones or hurricanes. 22

Chromosome A structure in the nucleus of cells which contains genes (of DNA). Linear and sometimes circular. See haploid and diploid. 32

Co-adaptation (or co-adaptation) The adaptation or evolution of characteristics of two or more species to their mutual advantage. 31, 38

Coevolution (or co-evolution) Characteristics of species evolve together (in concert). 27, 31, 38

Community Different species interacting in common space and time. 17

- **C. Density** The number of individuals (of all community species) per unit of space. Absolute or Relative. 23
- **C. Interaction** The number of species that can sustainably flourish within a community. 23
- **C. Resilience** The speed of an community to return to a stable state. 23

Diploid An organism having two sets (pairs) of chromosomes. 20, 32

Disease, Endemic A disease affects a small proportion of the overall population all the time. The disease is retained in the community. 26

Disease, Epidemic A disease affects a large proportion of the population at the same time. 26

Disease, Pandemic A disease affects an entire population at the at the same time. 26

Distribution The spatial range of a species, or the spatial arrangement (pattern) of a species in a habitat. 26, 96

D. Clustering The grouping of individuals for mutual benefit. 25, 26

D. Dispersal The spread of individuals away from each other (from parent or birthplace to breeding locations). 25, 26

Island D. An isolated (geographic) habitat. 26

Regular D. A spatial distribution of organisms with a regular (consistent) pattern. 25

Spatial D. The disposition of organisms according to a spatial property (such as altitude, latitude or territorial regions). 25

DNA Deoxyribonucleic acid. Nucleic acid molecules that contain genetic instructions used in all known living organisms. See Genome, RNA. 17, 33

Ecology The scientific study of environmental systems including the distribution and abundance of life and the interactions between organisms and their environment. 15

Ecosystem The set of biotic factors, abiotic factors, interactions and processes between the organisms of multiple species and their environment. 17, 18

Ensemble A group of organisms (of uniform or mixed species). 27

Evolution A change in *allele* frequencies in a population of individuals over time. 28

Fertilisation The physical union of male and female gametes. 34

Fitness A representation of how "good" an individual is in its current environment, or, the capability of a particular genotype to reproduce. 29, 30

Food Chain A listing of organisms in order of primary producers, secondary consumers and so on within an ecosystem. 19

Food Web All feeding interactions between organisms (at a species level) within an ecosystem. 19

Founder Effect The principle that the founders of a new colony carry only a fraction of the total genetic variation in the source population. 33, 34

Gamete A specialised reproductive cell, such as a sperm or egg cell. 32

Gametogenesis A process by which diploid or haploid precursor cells undergo cell division and differentiation to form mature haploid gametes. 32, 34, 67

Gene The basic biological units of heredity variation, composed of DNA sequences and organised into long chain chromosome molecules. 29

Gene Flow The migration of genetic material from one population to another. 29, 33

Genetic Drift Generational variations in *allele* frequency due to *stochastic* (random) processes. 29, 31

Genome A full set of chromosomes and genes for an organism (a complete genetic sequence). The genome can be divided into chromosome and gene components (DNA and/or RNA). 17

Genotype The genetic make-up of an organism, as distinguished from its physical appearance (the phenotype). See Phenotype. 29, 30

Growth, Exponential A rate of growth that increases as the size of the population increases (an exponent). 24

Growth, Logistic A growth curve with a characteristic "S" shape that indicates environmental (carrying capacity) growth constraints. 24

Habitat The place an organism lives, or is usually found. 25, 26

Habitat Cycle Cyclic changes in habitat factors including regular (such as seasonal variations, day/night cycles, tidal) and irregular (weather, earthquakes) quality. 25

Habitat Diversity The range (number) of habitats present in a particular region. 25

Haploid An organisms having only one set of chromosomes. 34

Individual See organism. 17

Insertion The placement of new (offspring) individuals into the population. 20

Interaction Interaction modes between species within a community. See Symbiosis, Predation and Competition. 27

Intrinsic Growth Rate The combined rates or natality, mortality, immigration and emigration contributing to population growth. 24

Juvenile An organism that has not yet reached its adult form, size or sexual maturity. 37

Life Cycle The cycle of life stages for the development and reproduction of organisms. 20

Life Table A tabular summary of age survivorship, based on natality and mortality rates. 23

Macro-evolution A high-level view of evolution, where the individual is an aggregate and the population contains multiple high-level aggregate groups. 37, 38

Matter Cycle The closed movement of material in an ecosystem model. Well known examples include the carbon cycle, nitrogen and oxygen cycles. 19

Meiotic The process of cell division that results in the formation of gametes. 32

Memetic Of or pertaining to memes; the replication of concepts. 21

Micro-evolution A low-level gene centric view of evolution, where a chromosome represents the population and gene the individuals. 37

Migration The movement of individuals from one location (population) to another. 24, 29

- **Emigration** Migration of individuals leaving a population. 24
- **Immigration** Migration of individuals into a population. 24

Molecule A sufficiently stable group of two or more atoms held together by a strong chemical bond. 17

Mortality Death rate, or the number of deaths during a certain period of time. 24

Mutation Heritable changes to genetic information including both small scale and large structural changes in genetic material. 29, 33

Natality Birth rate, or the number of lives births during a certain period of time. 24

Niche The role or functional position of a species in the community, or a description of the range of conditions required for sustained survival of a species. 21, 26

- **N. Breadth** The potential (idealised) range of all environment conditions in which an organism can thrive. 26
- **Complimentary N.** The tendency for coexisting niche species to differ along another niche dimension (resource). 26
- **Fundamental N.** The potential (idealised) range of all environment conditions in which an organism can thrive. 26
- **Realised N.** The range of the fundamental niches that a species occupies due to competition limits. 26

Occupation Density The ratio of occupied to unoccupied (available) habitat within an ecosystem community environment). Depends on species habitat requirements. 21

Organ A group of tissues that perform a specific function or set of functions. 17

Organelle A confined and specialised sub-unit of a cell with a specific function. ("elle": small, a "little-organ" of a cell). 17

Organism An individual. Any living thing; unicellular or multicellular. 17

Phenotype The visible appearance or set of traits of an organism, resulting from the combined action of genotype and environment. See Genotype. 29, 30

Pleiotropy A single gene influencing multiple phenotype traits. 36

Polygeny A phenotype feature that can be attributed to two or more genes interacting. 36

Polymorphism The occurrence of many forms of the same species. 25

Population Group of organisms (usually) of a single species occupying a given area at the same time. Typically, organisms with homologous alleles. 17, 23, 29

Symbiont Typically a smaller organism that is living *in* or *on* a host. 27

Tissue An ensemble of cells from the same origin that carry out the same function. Typically tissue cells are of the same type but not always. 17

Topology The way in which geographical or similar elements relate to one another. The configuration of a network, or the manner in which components are arranged and interrelated. 30

Trait Any observable or measurable characteristic of an organism, such as physical features (phenotype). 29, 30

Transcription The synthesis of RNA under the direction of DNA. 33

Utility A measure (value) of relative or practical usefulness. 30

Zygote A fertilised egg (combined egg and sperm cells) before cell division begins. 32

A.2 Graphs and Topology

Acyclic A directed graph that does not contain cycles. 94

Adaptability The ability to change form or function (adapt) in response to outside influence or stimulus. 82

Adjacency Matrix A representation of graph vertex connections in matrix form. The adjacency matrix is symmetric for undirected graphs, and asymmetric for directed graphs. 96

Adjacent Two vertices are adjacent if they are connected by one (or more) edges. 88

Affiliation Networks An affiliation is a type of connection or partnership between two entities. We can represent social (and other) systems as affiliation networks based on their social affiliation behaviours. 106

Assortative Coefficient A single measure of degree correlation using a Pearson correlation coefficient of the degree at either ends of an edge. This results in a value that is positive for assortative mixing, and negative for disassortative mixing. 99

Assortative Mixing A network whose edges are biased towards connecting similar vertices. In social networks, it is the behaviour of individuals to associate with others of similar age, race, location, religion and so on. 99, 106

  Disassortative Mixing A network whose edges are biased towards connections of dissimilar vertices. Sexual contact networks (by gender). 99

Bridge An edge whose removal from a graph results in a disconnected graph. 89

C.elegans *Caenorhabditis elegans*. A small (1mm) free-living nematode (roundworm) which lives in temperate soil environments. Well known because of the extensive research applied to it, including a complete mapping of the nervous system which exhibits small-world properties. 107

Chain See Path. 89

Citation Network A graph of publication as vertices, with edges indicated by publication citations of previous work. Classic examples are academic journals. 83

Cliquishness The tendency to associate with only a select group. See Assortative Mixing and Clustering Coefficient. 99

Cluster (Clustering) A term used in both the formal sense (as in the definition of the clustering coefficient), as well as an informal notion of general community structures with vague boundary definition. 85

Clustering Coefficient A measure of (quantity) of how likely the neighbour of a vertex are to be directly connected also. In social networking terms, two of your friends are also likely to be friends to each other. Also known as Transitivity or loosely as Network Density. 96, 100, 115

Complex System A system that can be broken into simpler parts, without the overall system itself being simple. Ideally a complex system can be broken down into smaller solvable pieces. Emergent qualities of a complex system may only be observable when the parts are combined (patterns). 82, 83

Complicated System A system that cannot (or easily) be broken down into simpler parts. Complicated is a term indicative of problematic, convoluted, torturous, difficult or inconsistent features. 83

Component Formally, the set of vertices that a vertex is connected (adjacent) to by its edges. Also known as a *neighbourhood*. Informally, a component simply refers to any unit or part of a large system. 85, 88

In Component The set of vertices (component) that can reach a vertex (via in-coming edges). 88

Out Component The set of vertices (component) that a vertex can reach (via out-going edges). 88

Critical An edge, vertex or component whose removal from a graph will result in a major change in graph topology (such as *disconnected*) or processes. 89

Cumulative Advantage See Preferential Attachment for a graph specific meaning. 117

Cycle A path in a directed graph (directed edges) with the same initial and terminal vertex. Also known as a *loop*. 87–89

Euler C. A cycle that passes through all graph edges. 89

Hamiltonian C. A cycle that passes through each graph vertex once. 89

Degree The number of edges connected to a vertex. Note that, because there may be more than one edge between two vertices, the degree may be greater than the number of adjacent vertices. Also known as the *local degree* or *valence* of a vertex.. 87, 88, 90, 96, 97, 111

D. Correlation A way of describing that the probability $P(k, k')$ of degree for one vertex k is correlated to the number of edges k' that neighbours have. A random graph has no such correlation, but many complex networks do. Also known as *degree-degree correlation*. 96

D. Distribution The distribution of vertex degrees in a graph, and characterised by a distribution function $P(k)$. 97

In-degree The number of "in-coming" edges to a vertex v with v as their *terminal* vertex. 87, 88

Out-degree The number of "out-going" edges from a vertex v with v as their *initial* vertex. 87, 88

D. Distribution Function A function that gives the probability that a randomly selected node has exactly k edges (degree). Different graph models have characteristic distribution functions. 97

Mean D. The mean degree $\langle k \rangle$ of all vertex degrees k_i in a graph. An overall measure of edge density in a graph. 97

Power-Law A relationship between variables such that one is proportional to a power of the other. A power-law degree distribution, when plotted on a log-log graph, present as straight lines. Power-law relationships exhibit the property scale invariance, and hence systems with power-law relationships are known as "scale-free". 96, 98

Diameter See Graph Diameter. 90, 102

Distance See Path Length. 89

Distance Matrix A matrix representation of a weighted graph where cells represent direct connection weights or minimum path weights. Useful for shortest path lookup between vertices. 96

Edge Connection between two vertices. The vertices are called the endpoints of the edge. An edge can be *directed* or *undirected*, and *weighted*. Also called a *link* (computer science), *bond* (physics), *tie* (sociology), *line* or an *arc*. The term *arc* is often reserved for directed edges. 81, 86

- **Directed E.** An edge that includes direction (initial and terminal vertices). 90
- **Weighted E.** An edge with an associated weight "value" (or values) used to represent information other than topology. 90, 92

Effective The capability to produce a desired result. For a network, the ability of a topology to support desired processes such as communication or movement. 82

Efficient The minimisation of cost. A measure of the realised cost in comparison to a worst case outcome. For a topology, the ability to effectively support the desired processes while minimising the cost of the topology. 82, 85

- **Global Efficiency** The normalised sum of the inverse shortest path lengths between all pairs of vertices; the harmonic mean. Can also be applied to weighted graphs. Unlike the characteristic path length measure it is well-defined for disconnected graphs. 96, 103
- **Local Efficiency** A localised form of the global efficiency measure, indicating the mean normalised ratio of local clustering for each vertex in a graph. 96, 103

Emergent Property A property that cannot be observed locally (at a component level) but only as a global structure (patterns) or dynamic abilities. 82

Entangled Graph A type of graph that exhibits a highly "interwoven" topology, with highly homogeneous structure with respect to degree, node distance, betweenness and loop distance which are all very narrow. 105

Free Tree An unrooted tree graph. (A normal tree without a single special root node). 94

Girth See Graph Girth. 90

Graph A finite set of *vertices* connected by *edges*, also called a *network*. A description of vertices, edges and connections (excluding geometry or weight) is known simply as the *topology*. 81, 86

- **G. Betweenness** The vertex that participates the most number of times in the shortest path between all vertex pairs in the graph. This is a measure of shortest path utility. 102, 103
- **G. Centre** The vertex with the lowest mean path length to all other nodes in the graph. However this does not indicate that it will be used (utility) for other shortest paths in the graph. 102, 103

G. Cost A measure of the relative cost of a network compared to a fully-connected graph with all possible connection. 85, 96

G. Degree The maximum degree of any vertex present within a graph. See Degree. 90

G. Diameter The number of edges in the longest geodesic path between two vertices *or* the average geodesic path length. 90

G. Girth The length of the shortest cycle in a graph. See Cycle . 90

G. Order The number of vertices in a graph. 90

G. Size The number of edges in a graph. 90

Graph Layout A visual representation type or method for graphs based on the organisation of vertex placement and the drawing of edges. 93

Force-directed Placement A graph layout algorithm using an iterative gradient-descent process to minimisation an energy function based on a physical (mechanical) model of forces (springs, attraction, repulsion). 94

General A description of graph drawing methods that are "general" in nature (and most applicable to complex topology representation). 93

Grid Graph layout using a regular grid pattern. 92

Hierarchical Graph layout using a regular hierarchical organisational pattern. 92

Lattice Graph layout using a regular lattice pattern. 92

Linear Graph layout using a regular linear pattern. 92

Ring Graph layout using a regular ring pattern. (A linear pattern that repeats). 92

Graph Models Methods of defining or creating graph topology through either a deterministic description or a randomised graph generator (*growth model* or *game*). 108

BA A growth model using preferential attachment proposed by Barabási and Albert. It is similar to Price's model except that is uses undirected edges. Preferential attachment is based only on vertex degree. 109, 118

Configuration Random graph model using a specified (configuration) degree distribution (or function) and applying it to an otherwise random graph. 109, 114

ER Random graph models proposed by Erdös and Rényi, using either a fixed number of vertices $G(n, p)$ or a fixed number of vertices and edges $G(n, m)$. 83, 109, 113

Hierarchical Graph models based on repetition (growth) of simple components creating a hierarchical structure. 112, 123

Merge-Regenerate A graph model whose central principle is that groups of nodes are selected, merged and then new nodes added with random edges. The initial graph is simply random. Also known as "merge and create" or "aggregation and injection" or simply "merging". 109, 121, 124

Price A preferential growth model used by Price in the research of paper citation networks. As new nodes are added, the probability that an existing node will be connected to is directly proportional to the number of edges it already has. 117–119

Maximal See Complete Graph. 90

Mean Path Length See Characteristic Path Length. 85, 115

Motif A pattern of interconnection within a graph that occurs at a frequency higher than expected for a similar randomised graph with the same degree characteristics. Motifs can be used to identify and classify network types and functions. 96, 100, 101

Neighbourhood See Component. 88, 111

Network Density Typically a reference to the degree of vertices in a graph. See: Degree, Mean; Degree Distribution; and Clustering Coefficient. 99

Node See Vertex. 81

Order See Graph Order. 90

Path A sequence of consecutive edges in a graph connected by edges. Also known as a *chain*. 87–89

- **Euler P.** A path that passes through all graph edges. 89
- **Geodesic P.** The shortest path in a graph between two vertices. There may be more that one equivalent geodesic path between vertices. 88, 89
- **P. Length** The number of edges traversed in a *path* (chain). Also known as *distance*. 89

Preferential Attachment Used to describe a graph growth process where new edges (attachments) are allocated with preference to an existing vertex quality such as the number of edges a vertex has. Also by many names including the "Yule Process", "Gibrat's Law" and "cumulative advantage". 109, 118

Resilient The ability to recover from adversity or other challenges. Applies to networks and processes that can adapt to topology changes (vertex removal). 82, 85

Ring Lattice A regular graph model formed in a ring. Used as the basis for a famous small-world graph model by Watts and Strogatz. 84

Robustness A measure of how stable graph qualities (such as connectedness) and properties (such as mean path length) are in response to vertex removal. 90

Scale-Free A scale free network is one that has been observed to contain scale-invariant relationship in properties, namely degree distribution. Scale-free properties have been strongly associated with complex systems. See Power-Law. 85, 93, 98, 124

Separation Characteristic The properties and qualities that describe the path between two vertices in a graph. In social networks, the separation degree (steps) between two people. 84

Simulated Annealing An optimisation method that simulates the physical process of annealing by allowing the occasional acceptance of less attractive solutions or values. 94

Six Degrees of separation. Refers to the idea that in the world-wide network of people, any two people will be separated by an average of six steps (involving seven people). This small average number of steps gives the impression that we live in a "small world". 84, 87

Size See Graph Size. 90

Small-World Phenomena The idea that people are separated, on average, by a small number of intermediate steps, and that this gives people the impression that we relatively close to everyone even though there are many people in the world. Networks with similar characteristics (low mean path length and high clustering) as said to exhibit the same "small-world" phenomena. See Six Degrees. 84

Spring Embedder See Force-directed Placement. 94

Topology (Graph specific) A description of vertices, edges and connections in a graph (excluding geometry or weight properties). 86

Transitivity See Clustering Coefficient. 99, 100

Travelling Salesman Problem (TSP) A classic tour optimisation problem in which a hypothetical salesperson must find the most efficient sequence (order) of destinations to visit in their territory. See Hamiltonian Cycle. 89

Tree A connected acyclic graph. A *rooted* tree has a single special identified root node. Graph layout using a regular, tree based pattern with a root, parent, children, siblings and leaf concepts. 92, 93

- **Children** Tree nodes that are connected to a higher-level parent node. 94
- **Parent** A tree node that has children node(s) of a lower level connected to it. 94
- **Root** A single root node at the first level of a tree. 94
- **Sibling** A tree node that shares the same parent node with another node. 94

Valence See Degree. 87, 88

Vertex (Plural: *vertices*). A single point or 'dot' in a graph. Also called a *node* in computer science, a *site* in physics or an *actor* in sociology. 86

- **Isolated V.** A vertex of zero degree. (No edges in or out.) See Vertex Degree. 88
- **Initial V.** For a directed edge, the vertex the edge starts from. 89
- **Terminal V.** For a directed edge, the vertex the edge ends at. 89

Appendix B

Benchmark Problems

B.1 Domain Qualities

There are many traditional and classic benchmark problems that have been used to evaluate search algorithms, including the different algorithms of evolutionary computation (EC). Within EC, some benchmarks are particular to specific dialects such as GAs or EP. Although the generality of results based on benchmarks must be limited, they are a useful means of developing qualitative insight into the effectiveness of EA based search. So the role of most benchmark suites is to collect examples of different domain qualities. These can include:

- **dimensionality**, given that most problems increase in difficulty with increases in input and output dimensionality;
- **range** or resolution of parameter variables;
- **constraints** that may be imposed upon the feasible space;
- **objectives** which may be singular or multiple, including maximisation or minimisation. It is possible to argue that these are also constraints;
- **modality**, either unimodal or multimodal, and may include the concave or convex form;
- **regularity** of modality (if present) or otherwise random features; and
- **separability**. If the domain can be broken into a summation of independent functions the domain is likely to be easier than a domain with complex inter-dependent variables.

It is a practical requirement that benchmark problems include high-dimensional instances, as it is arguable that to be of real-world value a trivial one or two dimensional test domain is insufficient.

The notation used is x_i for each value of a solution vector $\mathbf{x}$, which is applied to a function $f(\mathbf{x})$. A solution vector is of length n and typically maps to the dimensions of the problem. The exception being where the solution vector maps multiple values to a single dimension of the problem domain. An optimal solution vector is denoted as $\mathbf{x}^*$ such that $f(\mathbf{x}^*)$ gives the optimal possible output for the domain.

Many functions have a traditional minimisation or maximisation goal. For example, benchmark problems from the domain of continuous "function optimisation" are often minimisation domains, while benchmark functions used to investigate the role of niching mechanisms are often maximisation in nature. However, in almost all cases it is a trivial

matter to create "inverted" forms of the initial function by changing the overall sign and optionally adding an "offset" value to elevate (or lower) the surface as needed. The examples listed in this appendix have been presented in their most common form.

All of the charts shown in this appendix have been created using data generated directly from the ESEC framework software as part of testing and validation. Although not all of these functions are of importance to the key research experiments of the thesis, they are important useful additions to the EC framework software for classic comparisons. There are also additional domains within the framework that are not specifically mentioned in this appendix.

B.2 Classic Binary Problems

B.2.1 Introduction

The following binary problems are not considered examples of "real-world" domains. Rather, each of the following binary problems have been used by researchers as part of investigations into the processes of evolutionary search. For example, the OneMax function is a basic binary search domain, while the Royal Road function is specifically designed to match the "building block" mechanisms that have been theorised at work in GAs. The deceptive 3-bit and 4-bit functions are tough domains for any comparative hill-climbing function.

These problems have been included as part of a good coverage and reference for those interested in this area. Of most relevance to this thesis are the binary (represented) problems in Section B.4 such as MMDP, P-PEAKS, L-SAT, and NK.

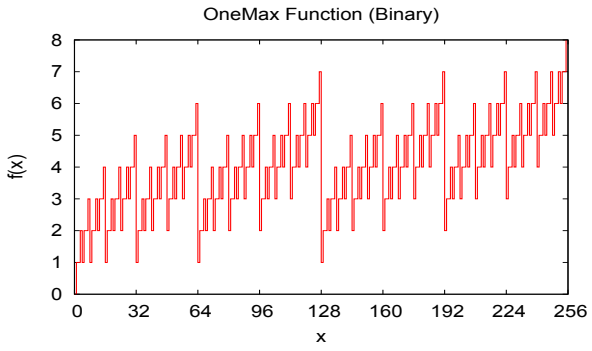
B.2.2 OneMax Function

The "OneMax" function is a simple maximisation problem and commonly used as a GA binary genome benchmark test. The search objective for this domain is to maximise the number of '1' bits in a bit string [304]. For this reason the problem is also known as "Bit Counting" or quite simply the "Max" problem.

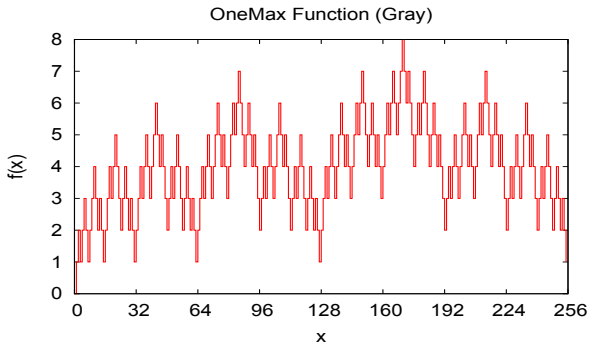
Stated in a more formal manner, the search objective is to find a vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ where the variables of $\mathbf{x}$ are in the range of (0,1). That is, $x_i \in [0, 1]$. The function is defined by the following equation:

$$F(\mathbf{x}) = \sum_{i=1}^{n} x_i \tag{B.1}$$

The optimum maximised vector is $\mathbf{x}^* = (1, 1, \ldots, 1)$ resulting in an output of $f(\mathbf{x}^*) = n$, where n is the length of the bit string.



(a) Standard binary encoding



(b) Gray encoding

Figure B.1: The OneMax function on a simple 8-bit binary string representing integer values from 0 to 255. The output of the function is the total number of 1's that occur in the binary string. In (a) standard binary encoded integers have been used and results in some large step transitions between adjacent values, while (b) uses Gray encoding and transition from one adjacent input value to the next only require a single bit change.

B.2.3 Royal Road Function

The Royal Road function has been used as part of the investigation and analysis of GAs, in particular with respect to comparisons with hill-climbing algorithms. Mitchell and Forrest performed a number of experiments during the 1990s to gain insight into what makes a "hard" problem for GAs, and how GAs perform when hierarchical building-block (BB) schema are deliberately made part of the search domain. See [239, 238] and related work.

The Royal Road function is a discrete, non-deceptive, unimodal problem space. In principle, BB combinations should allow a GA to find the optimal solutions easily. Comparatively, a hill-climbing algorithm, using a single bit-change-per-step style of adaptation, should find the domain difficult. The component evaluation uses an *all-or-none* style of reward. A GA should try to recombine components that have been "rewarded" by evaluation.

As a general case, the minimising Royal Road function can be described as:

$$f(\mathbf{x}) = cn - \sum_{i=i}^{n} \sigma(x_i) \tag{B.2}$$

where c is the number of bits used to represent each variable and n is the number of dimensions. The function σ provides reward for the presence of schema or building-blocks. For example, we can define σ as:

$$\sigma(x_i) = \begin{cases} c & \text{if}(x_i^1 \wedge x_i^2 \wedge \ldots \wedge x_i^c = 1) \\ 0 & \text{otherwise} \end{cases}$$

So for a binary string genome, divided into say $c = 3$ bits per dimension, a global optima at $\mathbf{x}^* = ((1,1,1),(1,1,1),\ldots,(1,1,1))$ will give the minimum of $f(\mathbf{x}^*) = 0$. For clarity and comparison a sample of five vectors and evaluations is shown in Table B.1.

$\mathbf{x} = (x_1, x_2, x_3, x_4)$	σ	$f(\mathbf{x})$
000 000 000 000	0 0 0 0	$12 - 0 = 12$
000 111 000 000	0 3 0 0	$12 - 3 = 9$
010 011 111 000	0 0 3 0	$12 - 3 = 9$
111 000 000 111	3 0 0 3	$12 - 6 = 6$
111 111 111 111	3 3 3 3	$12 - 12 = 0$

Table B.1: Royal Road evaluations for five example vectors (rows) of length $n = 4$. The final vector is the optimum. Each component of the vector contains $c = 3$ bits and is represented by a space divider. The σ values show the individual component evaluation. It can been seen from the example vectors the all-or-none nature of the reward of this domain. Useful building blocks (entire components) are strongly rewarded.

B.2.4 Goldberg's Deceptive 3-bit Function

This deliberately deceptive function was conceived for the analysis of binary string GAs. It uses three bits to represent values for each dimension of the function space, and has also been called "Goldberg's order-3 minimal deceptive problem". (See [133, 135] with [134] for more details of the design and analysis.)

The function is described by:

$$f(\mathbf{x}) = \sum_{i=1}^{n} m(x_i) \tag{B.3}$$

where x_i is a 3-bit binary string, and m is a mapped table of values for either the minimisation or maximisation case, shown in Table B.2.

Global optima are located at $\mathbf{x}^* = ((1,1,1),(1,1,1),\ldots,(1,1,1))$ and will yield either $f(\mathbf{x}^*) = 0$ for minimisation or $f(\mathbf{x}^*) = n \cdot 8$ for the maximisation case.

Note that the deceptive nature of the function operates at the genotype level, so that operators most affected are those that utilise binary level transitions. Figure B.2 shows a hypercube representation of a single 3-bit component with associated payoff values for the maximising case.

i	String	Max. Value	Min. Value
0	000	7	1
1	001	5	3
2	010	5	3
3	011	0	8
4	100	3	5
5	101	0	8
6	110	0	8
7*	111	8	0

Table B.2: Goldberg's deceptive 3-bit function. The * indicates the optimal string. Values for both the maximisation (max.) and minimisation (min.) domains are listed.

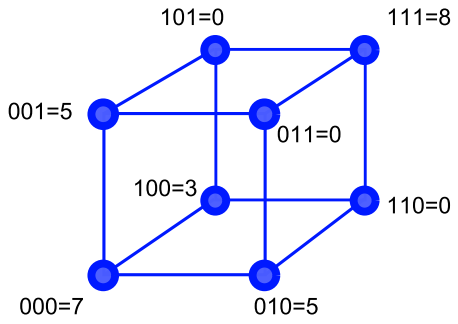


Figure B.2: Hypercube representation of Goldberg's 3-bit deceptive function. Each dimension of the problem is encoded as 3-bit value. Bit-wise transitions are deceptive both away from the optima and toward the poorer regions.

B.2.5 Whitley's Deceptive 4-bit Function

Similar in concept and motivation to Goldberg's 3-bit deceptive function, Whitley's 4-bit deceptive function includes misleading component values. However, the extended range of values allows deception across the entire range of 4-bit values, not just the extremes [365]. It is considered harder than the 3-bit deceptive function.

The definition of the function $\mathbf{x}$ is the same as Equation (B.3), however 4-bits per dimension are used, and the table m of coded values (shown in Table B.3) is different.

i	String	Max. Value	Min. Value
0	0000	28	2
1	0001	26	4
2	0010	24	6
3	0011	18	12
4	0100	22	8
5	0101	16	14
6	0110	14	16
7	0111	0	20
8	1000	20	10
9	1001	12	18
10	1010	10	20
11	1011	2	28
12	1100	8	22
13	1101	4	26
14	1110	6	24
15*	1111	30	0

Table B.3: Whitley's deceptive 4-bit function. The * indicates the optimal string value. Values for both the maximisation (max.) and minimisation (min.) cases are listed.

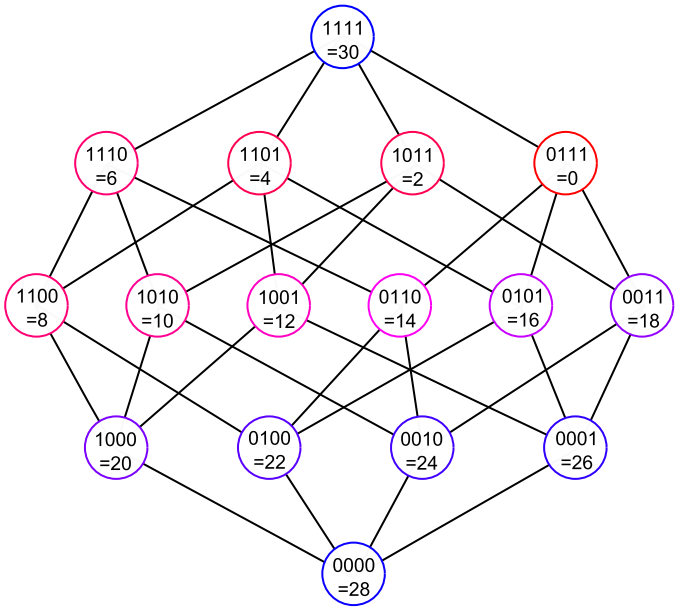


Figure B.3: Graph representation of the encoded values used in Whitley's 4-bit deceptive function. Both the binary string values and the maximisation values have been shown, with colour added to help represent the deceptive nature of the domain.

B.3 Classic Continuous Optimisation Problems

B.3.1 Introduction

As already discussed in the body of this thesis, EAs are not intrinsically function optimisers despite the fact that they can be applied to this type of domain and many others. Historically, though, EAs have been heavily tested and validated against a number of continuous optimisation functions.

In this section all of the functions contain a single optima. Similar functions with multiple optima are presented in Section B.4. All functions in this section except two, Easom and FSM, are generalised for an n dimensional case, although some were historically proposed as two-dimensional functions (as in Rosenbrock's function). As a guide, a value in the order of $n = 30$ is often used as the "high dimensional test" value, although this is arbitrary.

Table B.4 show a list of the functions and a summary of the dimensionality, if the domain is stated with limited input constraints, if the function is separable (into additive components), and if the domain is unimodal or multimodal with regular or irregular features. Other qualities such as concave or convex nature could also be listed.

Function	n	Constraints	Separable	Modality
Sphere	$\mathbb{N}$	No	Additive	Unimodal
Hyperellipsoid	$\mathbb{N}$	No	Additive	Unimodal
Quadric	$\mathbb{N}$	No	No	Unimodal
Noisy Quartic	$\mathbb{N}$	No	Additive	Unimodal
Easom	2	Yes	No	Unimodal
Rosenbrock(2D)	2	No	No	Unimodal
Rosenbrock(N)	> 2	No	No	Irregular
Rastrigin	$\mathbb{N}$	No	Additive	Regular
Griewangk	$\mathbb{N}^*$	No	No	Regular
Ackley	$\mathbb{N}$	No	No	Regular
Schwefel	$\mathbb{N}$	Yes	Additive	Irregular
Michalewicz	$\mathbb{N}$	Yes	Additive	Irregular
FSM	6	Yes	No	Irregular

Table B.4: Comparison of classic continuous optimisation function. Modality is noted as "Regular" for regular multimodal and "Irregular" for irregular multimodal features. $\mathbb{N}$ is used to denote the set of positive integer values. Note: The Griewangk function does not generalise well to high n and is so indicated with $\mathbb{N}^*$.

B.3.2 Sphere

The Sphere function is a simple, continuous, strongly convex and unimodal surface. Traditionally it is a minimisation problem, but can easily be inverted and offset to create a maximisation problem.

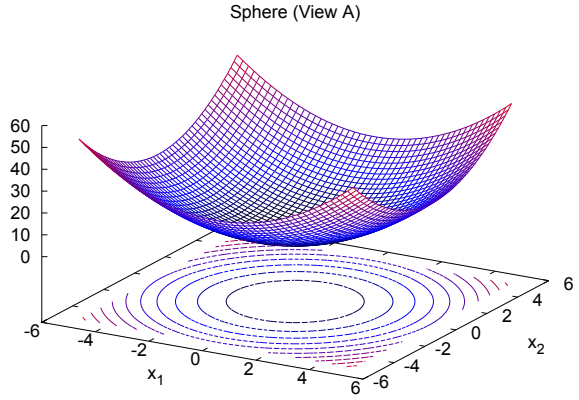
This classic problem is also known as De Jong's Function 1 (F1) due to its early and referenced use in GA research as a baseline test problem [74], and similarly for ES research [289]. The Sphere function was originally defined for two dimensions and then subsequently generalised for N dimensions [16].

The objective is to find a vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ of range $x_i \in [-32.768, 32.768]$ (although other ranges have been defined in literature) but is not considered constrained in general. The function is defined by the following equation:

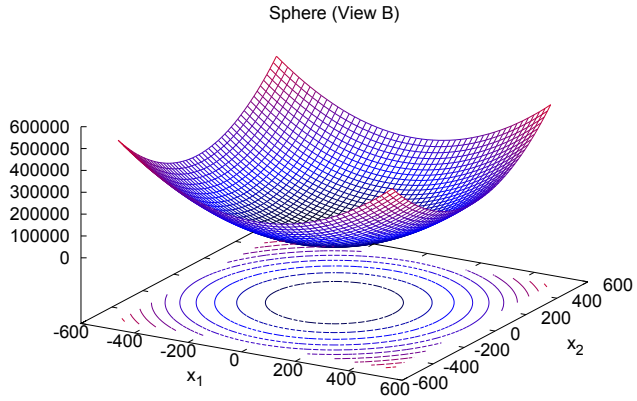
$$f(\mathbf{x}) = \sum_{i=1}^{n} x_i^2 \tag{B.4}$$

where the optimum vector $\mathbf{x}^* = (0, 0, \ldots, 0)$ will result in $f(\mathbf{x}^*) = 0$.

In Figure B.4 are two examples of the Sphere function at two different value ranges to emphasise the scale invariant nature of this test domain.



(a) $x_i \in [-6, 6]$



(b) $x_i \in [-600, 600]$

Figure B.4: Two examples of the Sphere function (De Jong F1) illustrating the scale invariant nature of this domain. Even though (a) and (b) are different (by two orders of magnitude) the resulting graphs look identical in shape and proportion.

B.3.3 Hyperellipsoid

The hyperellipsoid function is similar to the Sphere function, and is a simple, convex unimodal surface. It is also known as an "axis parallel hyperellipsoid function". The form of this function is essentially that of the Sphere function "stretched" along each additional dimensional axis i of n.

The hyperellipsoid function is included mainly as a reference for the Quadric function. The objective is to search for a vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ of range $x_i \in [-5.12, 5.12]$. The function is defined by the following equation:

$$f(\mathbf{x}) = \sum_{i=1}^{n} i^2 \cdot x_i^2 \tag{B.5}$$

where the optimum vector $\mathbf{x}^* = (0, 0, \ldots, 0)$ results in $f(\mathbf{x}^*) = 0$.

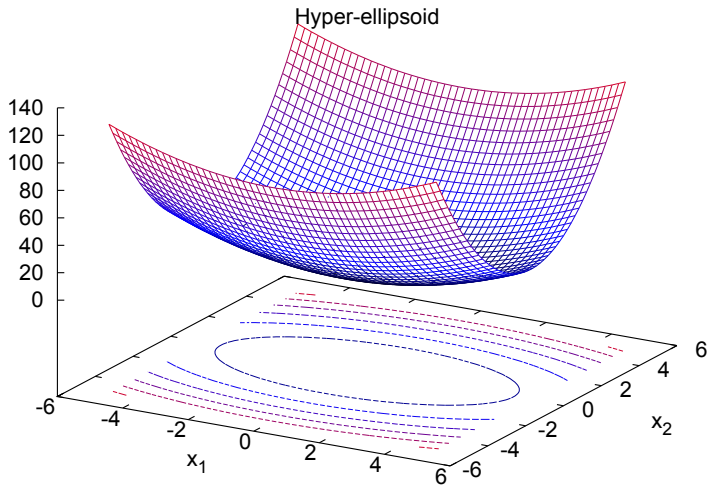


Figure B.5: Hyperellipsoid function for $n = 2$ dimensions.

B.3.4 Quadric

The Quadric minimisation problem is continuous, convex and unimodal. It is similar to the hyperellipsoid function (a stretched Sphere), however it is also rotated. Due to its use in cited benchmark suites it has also been given the names of "Schwefel's function 1.2", "Schwefel's Double Sum" [311] and the "rotated hyperellipsoid function".

The objective is to search for a vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ of range $x_i \in [-65.536, 65.536]$. The function is defined by the following equation:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \left[\sum_{j=1}^{i} x_j \right]^2 \tag{B.6}$$

where the optimum vector $\mathbf{x}^* = (0, 0, \ldots, 0)$ will result in $f(\mathbf{x}^*) = 0$.

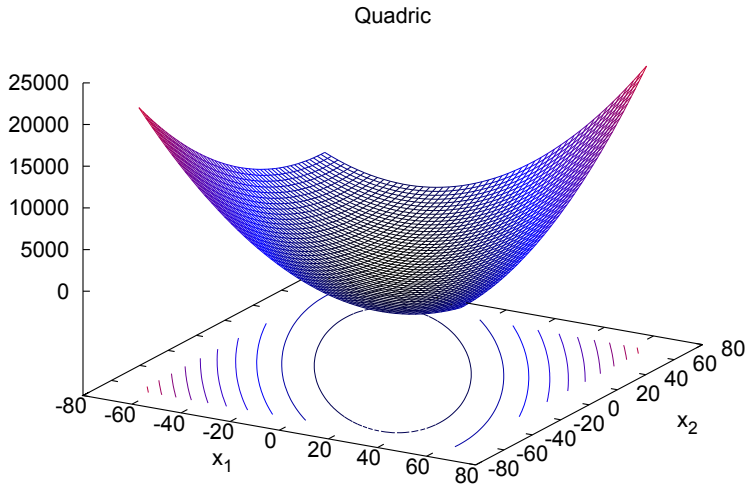


Figure B.6: Quadric function for $n = 2$ dimensions. The standard range of $x_i \in (-65.536, 65.536)$ has been used.

There are other variations to this function that displace (offset) the optimum region in proportion to the magnitude of each axis dimension value, and thus create a non-symmetrical version. Such variations are applied when there are specific research questions begin investigated.

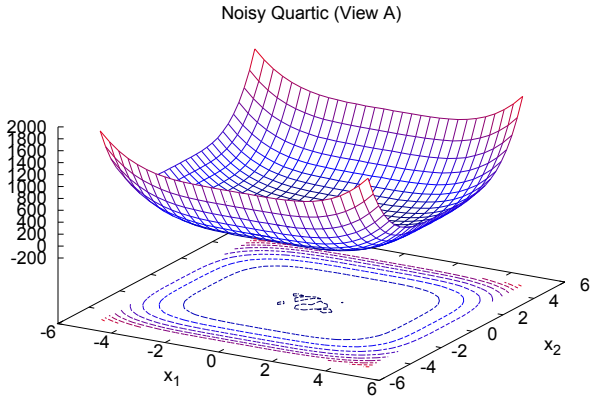
B.3.5 Noisy Quartic Function

Search algorithms can be tested for their "robust" qualities with the use of noisy test problems. The Noisy Quartic function is a summation of simple quartic terms with an added noise term.

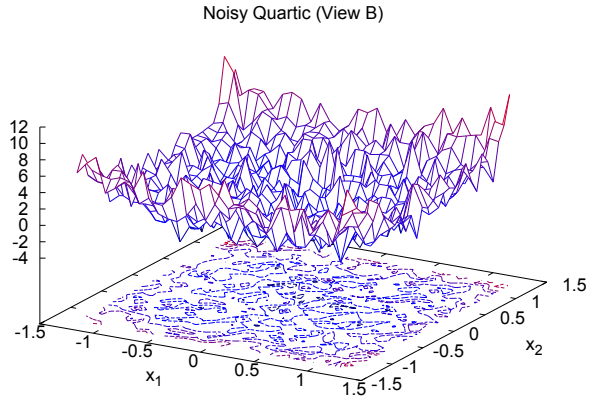
$$f(\mathbf{x}) = \sum_{i=1}^{n} i \cdot (x_i)^4 + random \tag{B.7}$$

In the case shown here the noise term is a random Gaussian variable (with a mean of 0 and standard deviation of 1). Some references appear to use a uniform random value in the range of $[0, 1)$ or similar. As the importance of the noise term is simply to *disrupt* the search space the general quality of any noise term is similar. See Figure B.7 which shows the influence of noise on the familiar quartic function.

It is not possible to define the minimum location or expected value due to the noise term. Despite this, a robust search should be able to locate a solution vector in the area of $\mathbf{x}^* \approx (0, 0, \ldots, 0)$ resulting in $f(\mathbf{x}^*) \approx 0$.



(a) $x_i \in [-5, 5]$



(b) $x_i \in [-1.28, 1.28]$

Figure B.7: The noisy quartic function from a (a) macro view of the entire domain (showing the overall quartic function), and (b) the micro view near the average global optima with obvious noise perturbations.

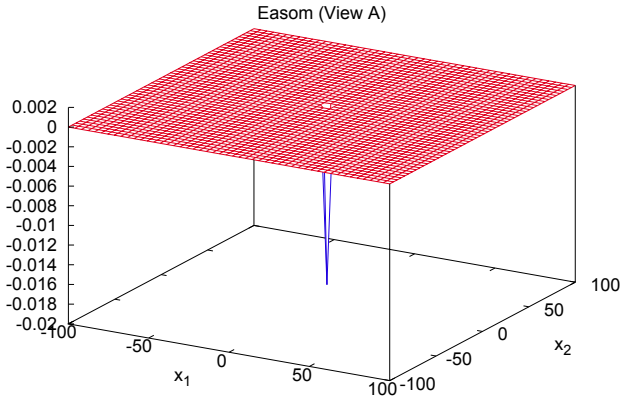
B.3.6 Easom Function

The Easom function is a very specific unimodal two dimensional maximisation function [95]. It can be inverted for use in minimisation (as shown here). The most striking feature of the search landscape is the large global plateau area with almost no gradient "hint" information to guide a gradient-based search to the small region surrounding the global optima. Thus the Easom function is a difficult domain for methods reliant on global gradient information and can allow other techniques (such as EC) to perform comparatively well in this spartan domain.

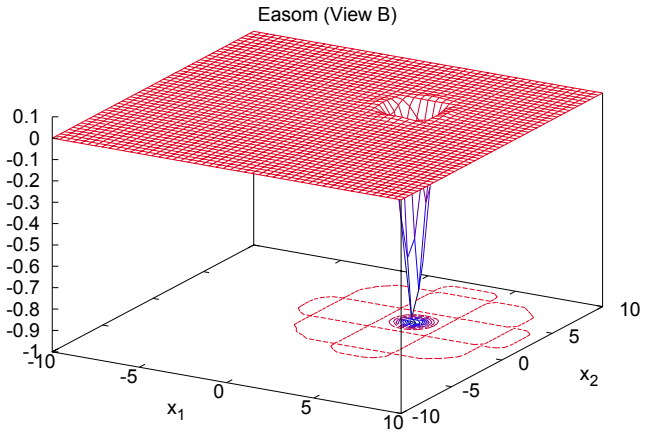
The objective is to search for a vector $\mathbf{x} = (x_1, x_2)$ where both dimensions are in the range $[-100, 100]$. The function is defined by the following equation:

$$f(\mathbf{x}) = f(x_1, x_2) = -\cos(x_1) \cdot \cos(x_2) \cdot e^{-(x_1-\pi)^2-(x_2-\pi)^2} \tag{B.8}$$

where the optimum vector of $\mathbf{x}^* = (\pi, \pi)$ will result in $f(\mathbf{x}^*) = -1.0$.



(a) $x_i \in [-100, 100]$



(b) $x_i \in [-10, 10]$

Figure B.8: The Easom function at both (a) the full range of values (-100,100), and (b) a closer inspection of the surface (-10,10) which emphasises the small region of landscape information to assist search.

B.3.7 Rosenbrock's valley

Rosenbrock's valley is a very well known classic optimisation problem [294], with many alternative titles. For example, it is also called De Jong's Function 2 (F2), Rosenbrock's "saddle" and the "Banana" function.

The function is continuous and unimodal, though the non-convex surface gradient can be deceptive to some search methods. In Figure B.9 the domain is presented with both a linear and log output scale to emphasise the subtle and deceptive nature of the global optimal area.

A two dimensional case, where the objective is to search for a vector $\mathbf{x} = (x_1, x_2)$ of range $x_i \in [-2.048, 2.048]$, is given as:

$$f(\mathbf{x}) = (1 - x_1)^2 + 100 \cdot (x_2 - x_1^2)^2 \tag{B.9}$$

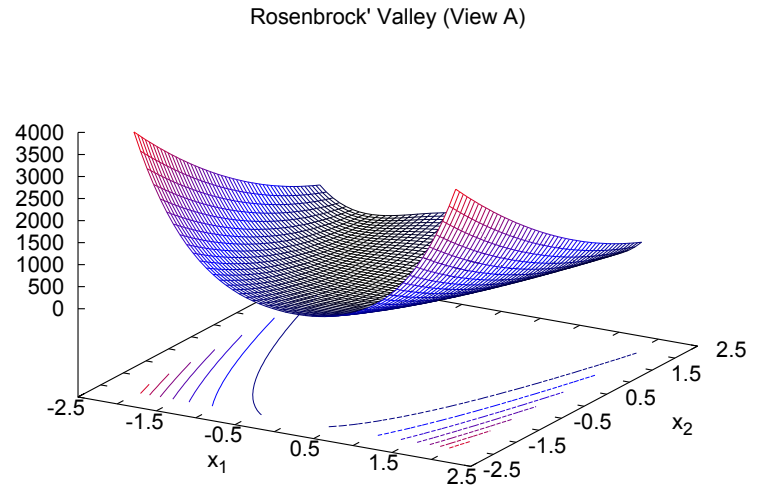
Convergence to the "valley" region is considered a trivial task due to the large dominant parabolic feature, however convergence to the global optimum is quite difficult for purely gradient based techniques.

For a more general n-dimensional case, the objective function is $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ also in the range $x_i \in (-2.048, 2.048)$. One version of the generalised n-dimensional case is shown below.

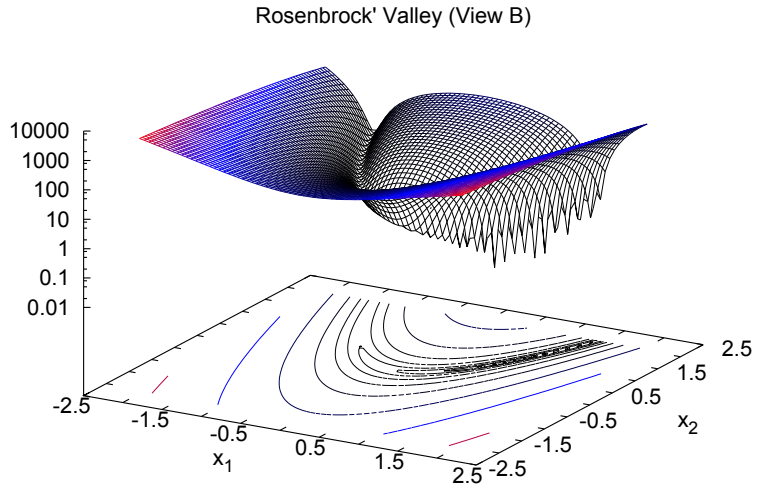
$$f(\vec{x}) = \sum_{i=1}^{n} 100 \cdot \left(x_{i+1} - x_i^2\right)^2 + (1 - x_i)^2 \tag{B.10}$$

where the optimum vector $\mathbf{x}^* = (1, 1, \ldots, 1)$ results in $f(\mathbf{x}^*) = 0$.

Note that there are other n dimensional generalisations that maintain a summation of separate independent 2D cases $(n/2)$ rather than allowing the more difficult overlapping interdependence of the case shown. It is not always clear how practitioners have implemented a general n dimensional case as the visually presented 2D cases are identical.



(a) Linear output scale



(b) Log output scale

Figure B.9: The 2D Rosenbrock function. Both (a) and (b) use the same range of values for $\mathbf{x}$, however (a) uses a linear output scale to give a sense of the overall global gradient, while (b) uses a log scale to emphasise the surface details in the global optimal region. Note the graph artefacts in the region near the global optima $\mathbf{x} = (1, 1)$.

B.3.8 Rastrigin Function

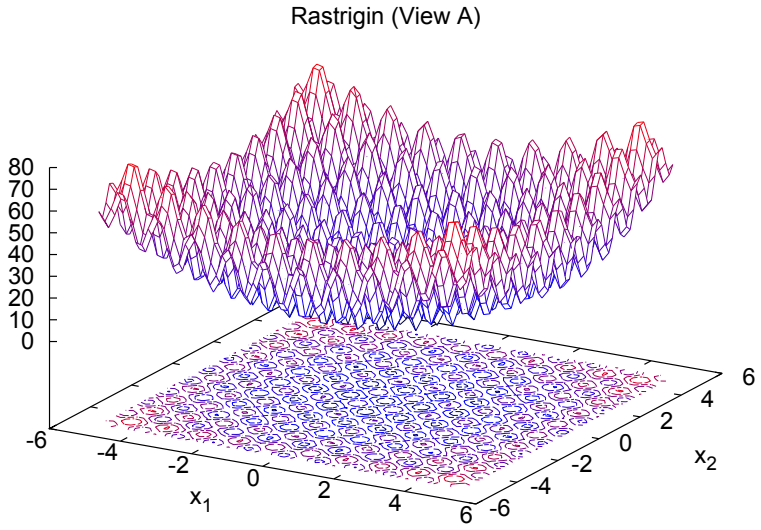
Rastrigin's minimisation problem is also similar to the Sphere (De Jong's F1) function, however the surface is modulated with an additional cosine term to induce multiple local minima and hence create a highly multimodal and deceptive surface. The minima are, however, regularly distributed. Regular modal features are typically easier for a search method to deal with and the function is also separable.

This function was first proposed by Rastrigin for two dimensions [342] and later generalised for n dimensions [247, 246].

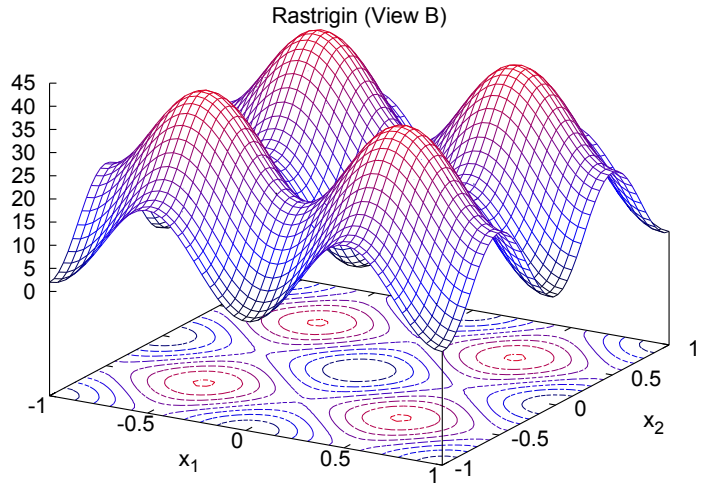
The objective is to search for a vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ in the range of $x_i \in [-5.12, 5.12]$. The function is defined by the following equation:

$$f(\mathbf{x}) = A \cdot n + \sum_{i=1}^{n} \left(x_i^2 - A \cdot \cos(\omega \cdot x_i) \right) \tag{B.11}$$
$$A = 10 \; ; \; \omega = 2 \cdot \pi \; ; \; x_i \in [-5.12, 5.12]$$

where A controls the amplitude and ω sets the frequency modulation. The optimum vector for the case shown is $\mathbf{x}^* = (0, 0, \ldots, 0)$ resulting in $f(\mathbf{x}^*) = 0$.



(a) $x_i \in [-5.12, 5.12]$



(b) $x_i \in [-1.0, 1.0]$

Figure B.10: The Rastrigin function has an overall Sphere quality with a modulated cosine. In (a) the entire global region is shown and (b) gives a closer view of the region near the global optima $(0, 0)$.

B.3.9 Griewangk Function

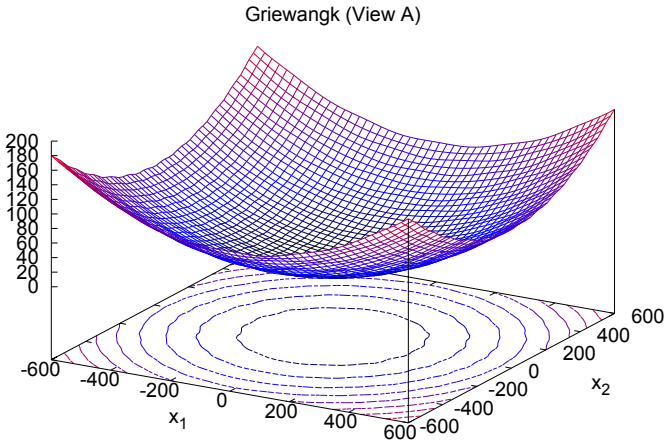
Griewangk's function is a minimisation problem similar to Rastrigin's function. The domain is also similar to an n-dimensional Sphere function, and modulated by a cosine based term [342, 246]. The function contains many local optima regularly distributed. Of interest are the distinct levels of scale (see Figure B.11). At the macro-scale level (largest value range) this function is somewhat easier to navigate than the Rastrigin function, however it contains deceptive qualities at medium and micro-scale ranges of values.

The search objective is for a vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ in the range of $x_i \in [-600, 600]$. The function is defined by the following equation:

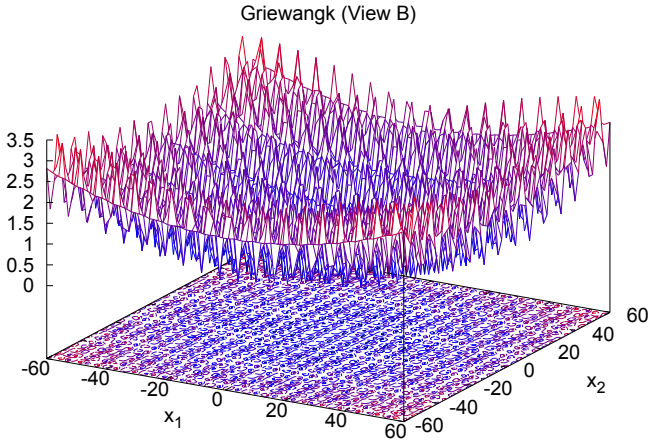
$$f(\mathbf{x}) = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} cos\left(\frac{x_i}{\sqrt{i}}\right) \tag{B.12}$$

where the optimum vector $\mathbf{x}^* = (0, 0, \ldots, 0)$ results in $f(\mathbf{x}^*) = 0$.

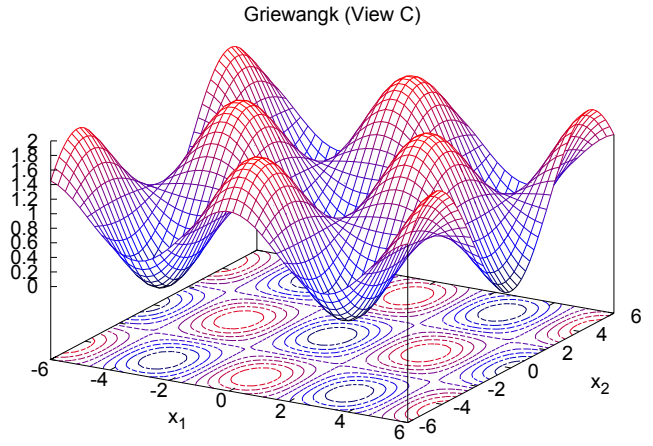
Although at first consideration this function appears to be a good scalable, non-separable and non-linear test function it has been shown by Whitely et al. [366] that it does not scale well and actually becomes relatively easier (smoother and so less deceptive) as n increases. With this understanding it is not recommended as a high-dimensional benchmark for comparing algorithm performance.



(a) $x_i \in [-600, 600]$



(b) $x_i \in [-60, 60]$



(c) $x_i \in [-6, 6]$

Figure B.11: The Griewangk function at (a) a macro-level view where it resembles the Sphere (De Jong F1) function, (b) an intermediate level where it resembles the Rastrigin function, and (c) the micro-level view of the global optima region.

B.3.10 Ackley Function

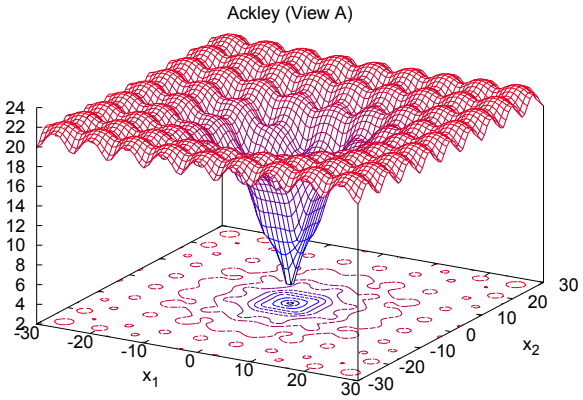
Ackley's function or "Ackley's Path" [1, 2] is a minimisation problem widely used as a multimodal benchmark. Originally defined for two dimensions it was later generalised for n dimensions by Bäck [21, 16]. The basic two-dimensional form is an overall exponential "well" that is modulated by a cosine term providing the familiar macro and micro level challenges of many benchmark functions. However as a distinction to the Rastrigin function, Ackley's function is not separable despite the appearance of regular local optima.

The search objective is for a vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ limited to the range of $x_i \in [-32.768, 32.768]$ (although some references state $x_i \in [-30, 30]$). The function is defined by the following equation:

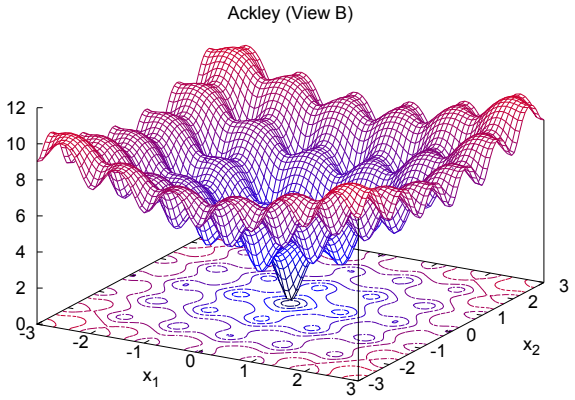
$$f(\mathbf{x}) = -20 \cdot e^{-0.2\sqrt{\frac{1}{n}\cdot\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n}\cdot\sum_{i=1}^{n}\cos(2\pi x_i)} + 20 + e^1 \tag{B.13}$$

where the optimum vector $\mathbf{x}^* = (0, 0, \ldots, 0)$ results in $f(\mathbf{x}^*) = 0$.

The constant terms of the function are listed separately in some references, but rarely altered in practice from those shown.



(a) $x_i \in [-30, 30]$



(b) $x_i \in [-3, 3]$

Figure B.12: The Ackley function at both (a) a macro view of the entire domain (showing the overall exponential well), and (b) the micro view near the global optima with obvious periodic modulation.

B.3.11 Schwefel Function

Schwefel's multimodal minimisation problem [310] is deceptive because the single global minimum is geometrically distant from the best local minima within the search space, making it likely for search algorithms to converge to non-optimal locations.

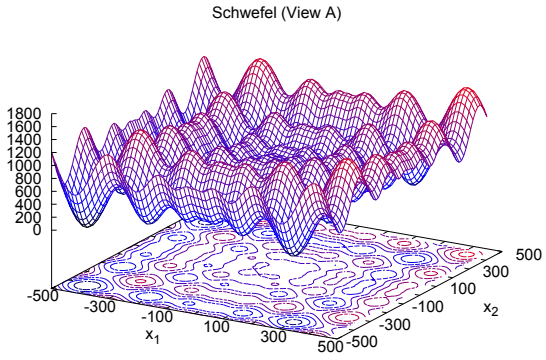
The search objective is for a vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ in the range of $x_i \in [-500, 500]$. The function is defined by the following equation:

$$f(\mathbf{x}) = \sum_{i=1}^{n} x_i \cdot \sin\left(\sqrt{|x_i|}\right) + 418.9829 \cdot n \tag{B.14}$$

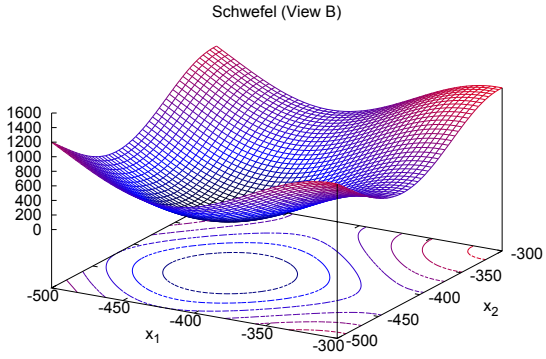
where the single optimum vector $\mathbf{x}^* = (-420.9687, -420.9687, \ldots, -420.9687)$ will result in $f(\mathbf{x}^*) = 0$.

The offset term is used to make the optimum value of $f(\mathbf{x}^*) = 0$. Some implementations remove this offset term, negate the sum, and require the search to find the optimum value of $f(\mathbf{x}^*) = -n \cdot 418.9829$ instead.[1]

Figure B.13 shows the entire domain and the region near the single global optimum. Other sub-optima are a significant distance from the global best.



(a) $x_i \in [-500, 500]$



(b) $x_i \in [-500, -300]$

Figure B.13: Schwefel function at (a) the entire domain view and (b) a micro scale view near the global optimal (-420.9687,-420.9687).

[1] This can affect search algorithms in regard to scaling and performance measures, but does not change the multimodal qualities.

B.3.12 Michalewicz's Function

Michalewicz's multimodal minimisation problem creates a domain with $n!$ local optima [230]. A parameter m is used to define the "steepness" of valley features. When m is large, the search space become very difficult as there are many narrow valleys and little global information (gradients) to guide a search process. (See Figure B.14.) When m is large this domain has been likened to searching for the proverbial "needle in a haystack".

The search objective is for a vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ in the range of $x_i \in [0, \pi]$. The function is defined by the following equation:

$$f(\mathbf{x}) = -\sum_{i=1}^{n} sin(x_i) \cdot \left(sin\left(\frac{i \cdot x_i^2}{\pi} \right) \right)^{2m} \tag{B.15}$$

where, for example, in the case of $m = 10$ and $n = 10$ the global minimum value is $f(\mathbf{x}^*) = -9.66$.

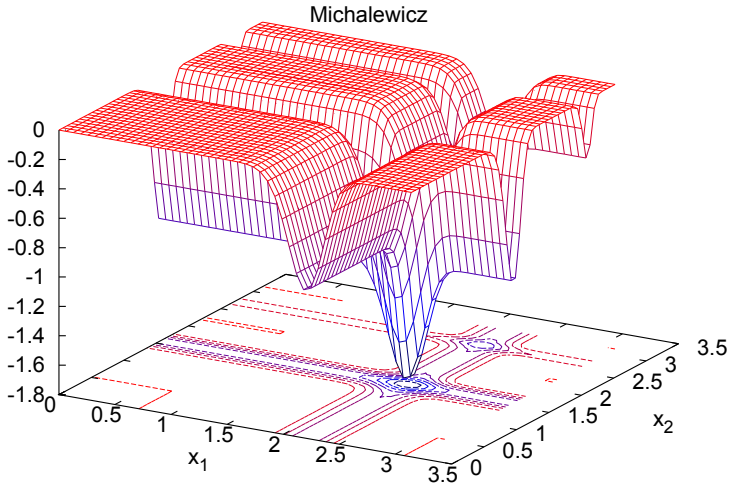


Figure B.14: An example of Michalewicz's function for a simple 2D ($n = 2$) case. Note that although the domain features can be separated, there is little gradient information that might guide a search process.

B.3.13 Frequency Modulation Sounds Problem

The Frequency Modulated Sounds problem (FMS) [343, 344] is a parameter identification problem in a highly complex multimodal function with strong epistasis. It consists of determining a set of six real value parameters $\mathbf{x} = (a_1, \omega_1, a_2, \omega_2, a_3, \omega_3)$ in the range of $[-6.4, 6.35]$ used in a FM sound model represented by:

$$y(t) = a_1 \cdot \sin(\omega_1 \cdot t \cdot \theta + a_2 \cdot \sin(\omega_2 \cdot t \cdot \theta + a_3 \cdot \sin(\omega_3 \cdot t \cdot \theta))) \tag{B.16}$$

where $\theta = (2 \cdot \pi/100)$.

For comparison a standard set of values is defined:

$$\mathbf{x}_0 = (1.0, 5.0, -1.5, 4.8, 2.0, 4.9)$$

which gives the standard function of:

$$y_0(t) = 1.0 \cdot \sin(5.0 \cdot t \cdot \theta - 1.5 \cdot \sin(4.8 \cdot t \cdot \theta + 2.0 \cdot \sin(4.9 \cdot t \cdot \theta))) \tag{B.17}$$

The objective is to minimise the difference between a standard sound function $y_0(t)$ and the function $y(t)$ using the supplied values for $\mathbf{x}$. The two functions are sampled at 100 points during the period of $2 \cdot \pi$, the error squared (removing the sign) and summed.

$$f(\mathbf{x}) = \sum_{t=1}^{100} (y(t) - y_0(t))^2 \tag{B.18}$$

An optimum vector $\mathbf{x}^*$ indicates that no error exists between the standard and the data, in which case $f(\mathbf{x}^*) = 0.0$. Because of the complexity of this domain, algorithms without local search operators find it difficult to find solutions with high accuracy, and so it is reasonable to adjust the accuracy expectation to the algorithm qualities and investigation goals.

As a general note for algorithm comparison, that is especially important for this difficult domain, is that value representation (and resolution capability) in each algorithm must be equal.[2]

Other model standard instances for $y_0(t)$ can be defined which enables this FSM to be a problem generator. (See Section B.5 for other benchmark problem generators.)

[2]For example, both algorithms should use equivalent 32-bit binary encoded strings to ensure sufficient resolution and equal algorithm representation ability.

B.4 Multiple Niche Problems

B.4.1 Introduction

The functions in this section have been included because they have been specifically used to research search algorithms that exploit niches (using methods such as fitness sharing, crowding and explicit speciation). Most test functions selected for such research are *multiple-solution* domains that contain more than one equal best optima, as occupation of multiple equivalent niches demonstrates an algorithm's ability to distribute resources.

Although niching is often discussed in research as applied to maximisation domains, it is of course possible with minimisation. The first four simple functions presented are maximisation domains. The last two functions, *Himmelblau* and the *Six-hump Camel Back*, are presented in their more common minimisation form but are easily inverted.

B.4.2 One-dimensional Standards

The following set of four one-dimensional functions operate on a single variable x, constrained to $x \in [0, 1]$, and have an overall maximisation objective.

$$f(x) = \sin^6(5\pi x) \tag{B.19}$$

$$f(x) = \sin^6(5\pi(x^{3/4} - 0.05)) \tag{B.20}$$

$$f(x) = \left(e^{-2\log(2)\cdot\left(\frac{x-0.1}{0.8}\right)^2}\right) \cdot \sin^6(5\pi x) \tag{B.21}$$

$$F(x) = \left(e^{-2\log(2)\cdot\left(\frac{x-0.08}{0.854}\right)^2}\right) f \cdot \sin^6(5\pi(x^{3/4} - 0.05)) \tag{B.22}$$

For (B.19) and (B.21) solutions are located at:

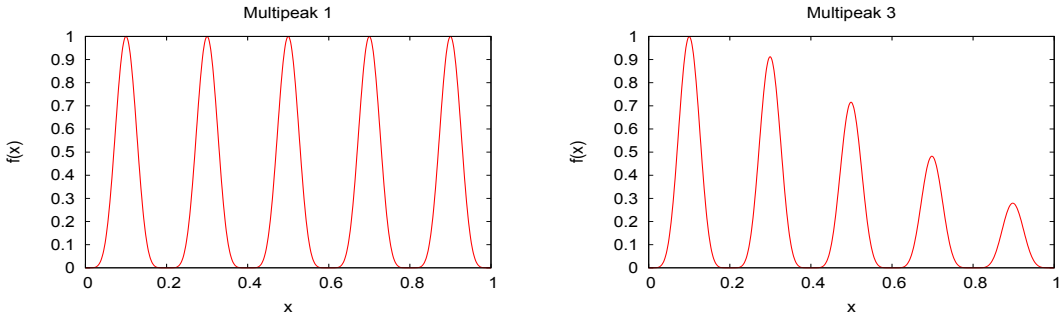
$$x = (0.1, 0.3, 0.5, 0.7, 0.9)$$

For (B.20) and (B.22) solutions are located at:

$$x = (0.08, 0.25, 0.45, 0.68, 0.93)$$

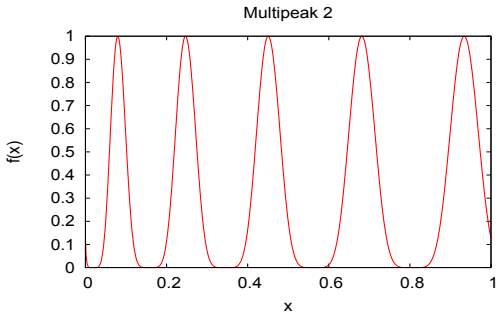
There are two *modal* varying qualities of these functions: *regularity* and *equality*. The first two functions (equations (B.19) and (B.20)) both contain equal value peaks, however in the first instance peaks are regularly (periodically) spaced, and in second peak spacing varies. The last two functions (equations (B.21) and (B.22)) both contain a set of peaks with different peak values, however in (B.21) the peaks are regularly spaced and in (B.22) the peaks are irregularly spaced. Graphs of these equations are shown in Figure B.15.

It is possible to extend all four functions to general n dimensional cases. Although this has been done it is a trivial exercise (using a basic summation with optional normalisation) and is not shown here. A more interesting variation is to both create an n dimensional and rotated form, so that the result is a non-separable multimodal domain for niches.

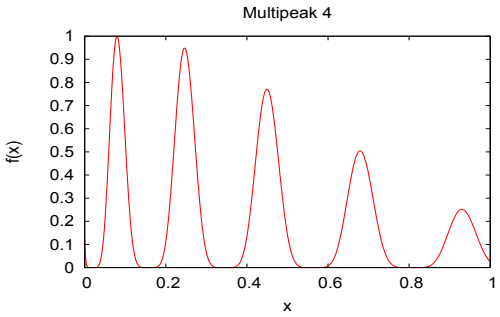


(a) Equal height and equal spacing

(b) Varying height and equal spacing



(c) Equal height and varying spacing



(d) Varying height and varying spacing

Figure B.15: Four standard one-dimensional multi-peak functions showing the different configurations of the varied modal qualities: *regularity* of peak spacing and *equality* of peak heights. (The order differs from the equation listing to enhance visual presentation.)

B.4.3 Himmelblau Function

The Himmelblau function is a standard multi-solution minimisation function of two dimensions. It is non-separable. It is formalised by the following equation:

$$F(\mathbf{x}) = (x_1^2 + x_2 - 11)^2 - (x_1 + x_2^2 - 7)^2 \tag{B.23}$$

where $\mathbf{x} = x_1 x_2$ and $x_1, x_2 \in [-5, 5]$. The set of standard known local optima are:

$$\{\mathbf{x}^*\} = \{(-2.81, 3.13), (3.00, 2.00), (3.58, -1.85), (-3.78, -3.28)\}$$

Each optima location gives an output of $f(\mathbf{x}^*) = 0.00$ however there are slight difference in the value of $f(\mathbf{x}^*)$ at higher resolution.

This function is commonly used in an inverted (and offset) form for niche-related research. Figure B.16 shows a graph of the entire domain space in its minimisation form.

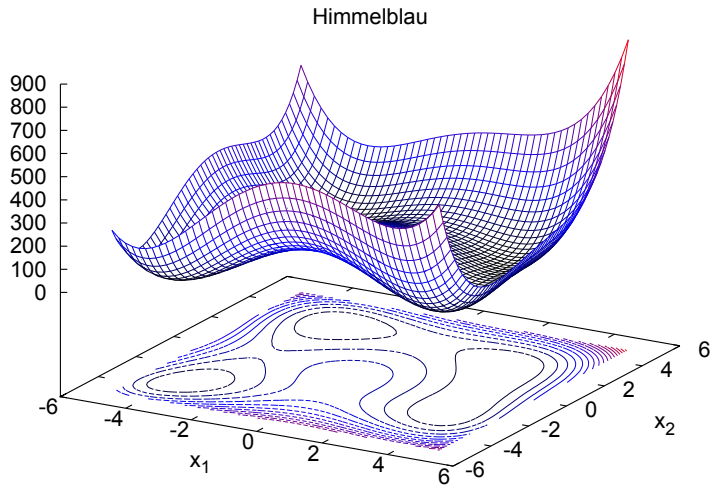


Figure B.16: Himmelblau function presented as a minimisation domain. The two-dimensional function contains four (approximately equal) optima.

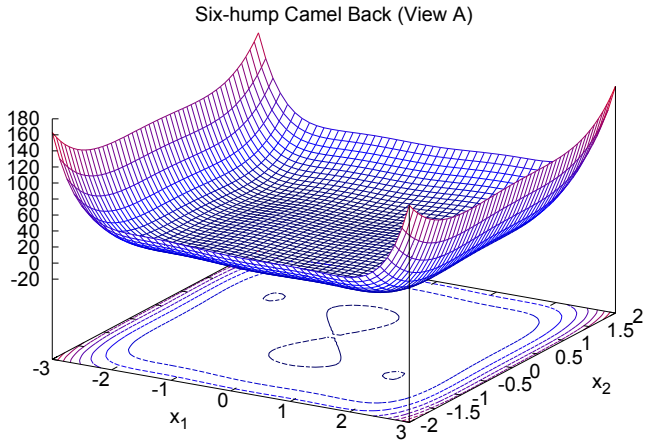
B.4.4 Six-hump Camel Back Function

The colourfully named "Six-hump Camel Back" function is a two dimensional non-separable multimodal minimisation problem with six minima features within the bounded domain [85]. Two of the minima are global (equal), making this a multi-solution problem. In Figure B.17 four of the optima features are clearly visible on the surface, while the remaining two resemble subtle plateau rather than clear optima.

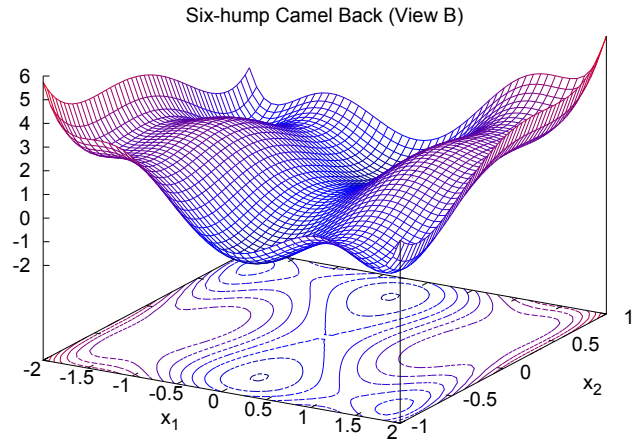
The search objective is a vector $\mathbf{x} = \{x_1, x_2\}$ in the range of $x_1 \in [-3, 3]$ and $x_2 \in [-2, 2]$. Asymmetric value range constraints are an unusual feature for a benchmark problem. The function is defined by the following equation:

$$F(\mathbf{x}) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4 \tag{B.24}$$

where the two optimum vectors are $\{\mathbf{x}^*\} = \{(-0.0898, 0.7126), (0.0898, -0.7126)\}$ with both resulting in $f(\mathbf{x}^*) = -1.0316$.



(a) $x_1 \in [-3, 3], x_2 \in [-2, 2]$



(b) $x_1 \in [-2, 2], x_2 \in [-1, 1]$

Figure B.17: Six-hump Camel Back Function. In (a) the entire asymmetric range of values is shown, while (b) is a closer view of the six "hump" areas (in this case minimisation "dips") of which four are distinctive.

B.5 Problem Generators

B.5.1 Introduction

Problem generators are used to create many unique instances of a particular problem class with specific common characteristics. Given a high number of random instances created by a generator, a search algorithm can be applied to each and the results provide a general and predictive view of the algorithms performance on the class of problem. Problem generators also help to support fairer comparisons of algorithms.

Table B.5 provides a summary of the generators described in this section, including the abbreviations used in this thesis and basic details. Most of the generators use binary representations but some can be altered to integer or real value forms. Problem descriptions in this section also try to highlight the benefit of these generator techniques for research.

Name	Long Name / Description	Values	Parameters
MMDP	**Massively Multimodal Deceptive Problem**	Binary	$k, n = 6k$
	Summation of 6-bit bipolar deceptive subcomponents. [139]		
P-PEAKS	**Multimodal Problem Generator**	Binary	n, P
	Nearest matching random peak. Strongly epistatic problem. [79]		
L-SAT	**Random Satisfiability Problem**	Binary	$C, L, V, n = V$
	Number (normalised) of True clauses in a Boolean CNF satisfiability expression. NP-hard and NP-complete when $L \geq 3$.		
NK	**NK Landscape (Kauffman's)**	Binary	n, k
	Fitness of n genes with k epistatic dependencies. [187]		
NKC	**NKC Landscape**	Binary	n, k, c, s
	Fitness of n genes, with k self epistatic dependencies and c interspecies links. Used as a co-evolutionary test. [187]		
SSP	**Subset Sum Problem**	Binary	n, C
	Determine if a subset S of elements from W can total the required sum C. NP-complete. [190]		
MAXCUT	**Maximum Cut Graph**	Binary	$G = (V, E), n, p$
	Find a cut of graph G that maximises the sum of weighted edges joining the two subgraphs V_1 and V_2. NP-complete. [190]		
ECC	**Error Correcting Code Design**	Binary	n, M, d
	Maximise the distance between codewords. [221]		
MTTP	**Minimum Tardy Task Problem**	Binary	n, tasks (l_i, d_i, w_i)
	Allocate tasks and minimise the penalty cost of un-allocated tasks. [190]		
MSG	**Max Set of Gaussians**	Real	$n, m, \mathbf{D}, p, r$
	Max value of a set of Gaussian components with parameterised peaks and distribution. Irregular and inseparable. [124]		

Table B.5: Summary of problem generators. See relevant sections for full details.

B.5.2 Massively Multimodal Deceptive Problem

The Massively Multimodal Deceptive Problem (MMDP) was specifically conceived by Goldberg [139] to be both deceptive and massively multimodal and hence help to "push the envelope" of GA research. The notion of a deceptive binary space has already been introduced in Section B.2 with Whitley's Deceptive 4-bit and Goldberg's Deceptive 3-bit functions.

A MMDP is composed of k deceptive bipolar binary substrings (s_i). Each binary substring (in this case composed of 6 bits) is evaluated based on the number of 1's it contains (*unitation*). The unitation value is then mapped to a *fitness* payoff value, with two equivalent global maxima at each unitation extreme (all 1's or 0's) and a wide deceptive local minima attractor based around the middle point (see Figure B.18). A summation of the substrings fitness values gives the overall MMDP fitness with the maximum payoff equal to k.

$$F(\mathbf{s}) = \sum_{i=1}^{k} fitness(s_i) \tag{B.25}$$

where $\mathbf{s}$ is the set of substrings (which together make up the vector $\mathbf{x}$) and *fitness* is the substring payoff function. In the case used here, the substrings are 6-bit and the fitness payoff mapping is shown in Figure B.18.

Unitation	fitness
0	1.000000
1	0.000000
2	0.360384
3	0.640576
4	0.360384
5	0.000000
6	1.000000

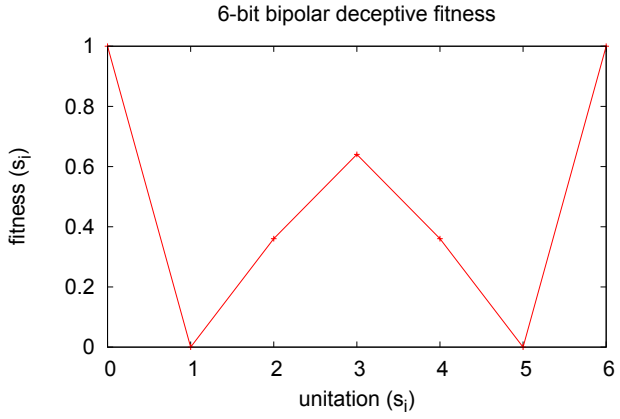


Figure B.18: A basic 6-bit bipolar deceptive payoff function used as a subfunction to create a Massively Multimodal Deceptive Problem. The fitness payoff values are shown as a table and as a representative figure.

For any substring length there are only 2^k global optima, although even for relatively short 6-bit substrings there are 22^k local optima (which certainly fulfils the criteria of "massively" deceptive). The k parameter controls the degree of modality.

B.5.3 Multimodal Problem Generator P-PEAKS

The P-PEAKS problem [79] was developed to help investigate epistasis by providing a tuneable parameterised problem space of selected size. It is also known as a "Bit-string Multimodality Generator"[3] or a random N-dimensional binary multimodal landscape.

Initialisation of the P-PEAKS problem creates P random Boolean n-bit strings (p_i) to represent "peaks" in the search space. The fitness of any n-bit string in the search space is the maximum number of bits is has in common with one of the peaks. By definition, the one winning peak is the "nearest" peak in the coded space to the search location $\mathbf{x}$. The fitness can be described by:

$$F(\mathbf{x}) = \frac{1}{n} \max_{1 \leq i \leq P} \{n - \mathit{Hamming}(\mathbf{x}, p_i)\} \tag{B.26}$$

where *Hamming* is a function to calculate the Hamming distance between two vectors. The fitness is normalised between 0.0 (no match) to 1.0 (an exact match).

By using a relatively small number of peaks in the landscape we can create a weak epistatic problem, while a relatively large number of peaks in the landscape creates a strongly epistatic problem. Note also that this problem is not effected by "flattening" of the landscape as epistasis increases (as observed in NK and L-SAT problems).

[3] See http://www.cs.uwyo.edu/~wspears/generators.html for implementation details of this and other generator test problems.

B.5.4 L-SAT Random Satisfiability Problem

A Boolean satisfiability (SAT) problem is one in which, for a given formula and a given set of values, if the expression is satisfied the result evaluates to True otherwise it is False. Or, more strictly from the Boolean perspective, is there some assignment of True and False values that will result in True? Many real and interesting problems can be expressed as formula (or rules) of this form.

If expressions are constructed using *literals* (variables) OR'd together to make up *clauses*, and multiple clauses are AND'd together to create *conjunctions*, we have the Conjunctive Normal Form (CNF). The NOT operator can be used only for single literals. In Disjunctive Normal Form (DNF) literals are AND'd together and clauses OR'd, and thus the domain is trivial to solve by comparison to CNF and is not considered here.

For example, the following Boolean logic expressions are all in CNF.

$$A \wedge B \wedge \neg C \tag{B.27}$$

$$A \wedge (B \vee \neg C) \wedge (A \vee \neg D \vee E) \wedge (C \vee F) \tag{B.28}$$

$$(A \vee B) \wedge (\neg B \vee C) \wedge (D \vee \neg E) \tag{B.29}$$

Expression (B.27) is a simple combination of literals, (B.28) shows a more complex form where clauses are of different lengths, and (B.29) is a regular set of clauses each with an equal number of literals.

By randomly creating a set of C clauses, each containing L literals selected from a set of V Boolean variables, we can create epistatic problems where the greater the number of clauses the greater the epistasis of the search domain. When $L \geq 3$ SAT problems are known to be *NP-hard*[4] [126].

In [236] it was shown that for SAT problems the "hardest area for satisfiability is near the point where 50% of the formulas are satisfiable", and further that this 0.5 satisfiable point is related to the ratio of clauses to variables. Their results suggest that for a fixed clause-length model a ratio of "roughly 4.3 times as many clauses as variables" will provide "computationally challenging instances" with a 50% chance of being satisfiable. For example, a random 3-SAT (L=3) problem of 430 clauses 100 variables will belong to this challenging set and is used in this work.

The method described below to create Random SAT problems in conjunctive normal form (CNF) for use with Binary search representation is based on an implementation by [236] and modified by William Spears[5].

To initialise the problem C clauses are created and filled with literals by choosing (with replacement) L variables uniformly from the set of all V variables, and negating with 50% probability to indicate NOT terms.

A solution attempt $\mathbf{x}$ is a Boolean string of length V. It assigns True or False values to each of the V variables.[6] The fitness of any given solution then is a ratio of the clauses satisfied using this allocation of Boolean values.

$$F(\mathbf{x}) = \frac{1}{C} \sum_{i=1}^{C} c_i(\mathbf{x}) \tag{B.30}$$

where $c_i(\mathbf{x})$ represents a clause of the set allocated with values from $\mathbf{x}$. The clause results in 1 for satisfied and 0 for not satisfied. The overall result $f(\mathbf{x})$ is normalised so that it will be 1.0 for all clauses satisfied and 0.0 for none satisfied.

[4] Informally *NP-hard* means that a problem is considered at least as hard as the hardest problems in NP (nondeterministic polynomial-time).

[5] See http://www.cs.uwyo.edu/~wspears/epist.html

[6] Note that not all variables many be used as literals to the random clauses created.

If the overall fitness is modified to be a linear weighting of clauses, the weights can be adapted to give more importance to the unsatisfied clauses.

$$F(\mathbf{x}) = \frac{1}{C}\sum_{i=1}^{C} w_i \cdot c_i(\mathbf{x}) \tag{B.31}$$

Eiben and van der Hauw [99] used Stepwise Adaptation of Weights (SAW) to update weight values based on the current best solution $\mathbf{x}^*$. This method uses integer weight values and will increment the value by 1 if a clause is not satisfied, and decrement the weight to 0 over time once satisfied.

$$w_i' = w_i + 1 - c_i(\mathbf{x}^*) \tag{B.32}$$

B.5.5 Kauffman's NK Landscape

Kauffman's NK fitness landscapes [187] were specifically designed as a rugged domain to test the effects of epistasis [17, 322, 98]. The two tuneable parameters are n for the problem length (number of binary genes in a haploid chromosome) and k for the number of linkages (epistatic interactions) between genes.

The landscape design uses two "principle" assumptions about the interactions of genes and the fitness of an individual.

- The fitness of a individual is the *additive sum of contributions* by each gene.
- The effects of *polygeny* and *pleiotropy* are effectively random.

To calculate the fitness for a chromosome $\mathbf{x}$ of n genes, the fitness of each *locus* (position) needs to be considered.

$$F(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} f(locus_i) \tag{B.33}$$

Simple versions of NK only consider k immediate neighbours to any gene, and although still interesting this is not a general model. Instead, a general model creates an arbitrary random mapping of k genes per gene – an *epistasis matrix* $\mathbf{E}$ of $n \times k$ random gene index values (excluding self index) – which can be predetermined and referred to as needed.[7] Listing B.1 shows a concise way of generating an epistasis matrix using Python.

Similarly, the fitness values for any single gene in combination with k other genes can be stored in a pre-determined *fitness matrix* of size $n \times 2^{k+1}$. (See Listing B.2.) This enables easy reference during the evaluation of any individual's genes but is quite memory expensive. Fitness values in F are from a normal random distribution of range $[0, 1)$.

The fitness for an individual then is the sum of each gene with its linked contribution of k other genes (Listing B.3) as stored in the fitness matrix F.

By allowing random walk models to repeatedly explore instances of the NK model, Kauffman noted that the best overall fitnesses were obtained when $k \approx 2$ and only increases slightly as n increases.

One problem with the NK model is that all genes have the same degree k of epistatic connections, which is likely not a "real-world" model. Similarly, it does not represent interspecies connections which is addressed by the NKC model.

[7] See `http://www.cs.uwyo.edu/~wspears/nk.c` for an implementation in C using a Hash map to save memory.

Listing B.1: Epistasis matrix initialisation code for an NK landscape in Python

```
def CreateE(n, k):
    ''' Create epistasis matrix E with N x K random index links '''
    E = [None] * n
    for i in range(n):
        links = range(n) # all possible links
        links.remove(i)  # no epistasis link to self
        shuffle(links) # random permutation
        E[i] = links[:k] # copy just what we need [0...(k-1)]
    return E
```

Listing B.2: Fitness matrix initialisation code for an NK landscape in Python

```
def CreateF(n, k):
    ''' Create fitness matrix F with n x 2^(k+1) random values '''
    F = [None] * n
    for i in range(n):
        # Use a list comprehension to create each list
        F[i] = [random() for col in range(2**(k+1))]
    return F
```

Listing B.3: Fitness evaluation code for binary genome with N-K dependencies.

```
def Evaluate(genes, n, k, E, F):
    ''' Calculate and return the fitness using n-to-k dependencies
        - The genome is a list of binary values named genes.
        - E (epistatis) and F (fitness) matrices pre-defined.
    '''
    fitness = 0.0
    for gene in range(n): # fitness for each locus
        fit_index = genes[gene]
        for i in range(n):
            multiplier = 2**(i+1) # 2^(i+1)
            epi_index = E[gene][i]
            fit_index += multiplier * genes[epi_index]
        fitness += F[gene][fit_index]
    return fitness / n
```

B.5.6 NKC Landscape

The NKC landscape model is an extension of NK model specifically used with co-evolving systems. Not only are there k linkage connections between between genes within an individual, but there are also c linkages to other species genes. From a coevolution point of view, this is a representation of the competitive and cooperative fitness influences between different species.

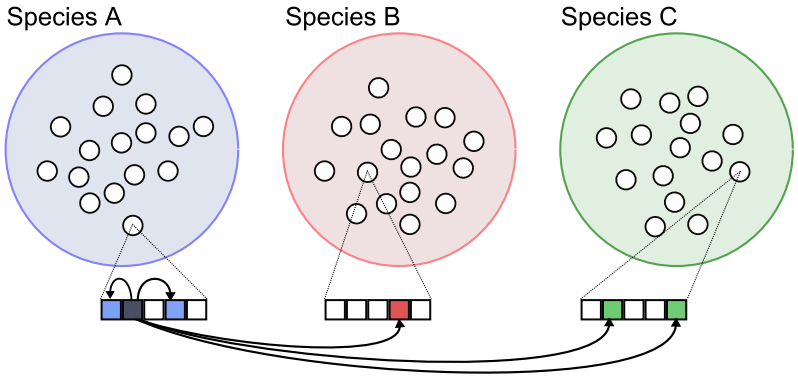


Figure B.19: A representation of the epistatic linkages for a single gene in the NKC model. There are $s = 3$ species, each individual has a genome length $n = 5$, with $k = 2$ intragenome linkages and $c = 3$ interspecies gene links. The linkages of the second gene in the first individual are shown.

An additional parameter is number of species s; if not specified this is typically $s = 2$. It is possible that rather than c representing interactions with a specific individual, c connections are representative of the species (ie. average or strongest traits) as this would be a more realistic representation of the biological population based (species level) interactions.

Unlike the NK model where best fitness values result with a low $k \approx 2$ value, for the NKC model a low k and high c lead to a chaotic regime while stasis occurs with high k and low c. It is interesting that the presence of c, although low, contributes to a stable system with higher k. The separation between chaotic and stable regions is roughly where $k = c$ and appears to be a natural region of attraction for systems that can adapt k.

An extension to the previous NK implementation for an epistasis matrix is shown in Listing B.4 and the fitness evaluation in Listing B.5. Note that as a simple way to evaluate a single individual interacting with other individuals, the genes from all interacting individuals are joined into a single string of genes. This makes the indices easy to store and reference during evaluation. A more flexible implementation is needed if n, k or c vary amongst species.

Listing B.4: Epistasis matrix for NKC problem.

```
def CreateE(n, k, c, s):
    ''' Create the epistasis matrix n x (k+c) with random links
        - also need group_size to calculate external gene indexes
    '''
    E = [None] * n
    for i in xrange(n):
        # Add the base k internal links first
        links = range(n)
        links.remove(i) # no epistasis link to self
        shuffle(links)
        E[i] = links[:k]
        # Now add c additional links (indices continue past n)
        links = range(n, n*s)
        shuffle(links)
        E[i].extend(links[:c]) # only what we need
    return E
```

Listing B.5: Fitness evaluation for binary genome with N-K-C dependencies.

```
def Evaluate(genes, n, k, c, E, F):
    ''' Calculate the fitness using n to (k+c) dependencies
        - genes is a join of this individual and any other
          individual needed for the c external dependencies
    '''
    fitness = 0.0
    for gene in range(n): # do this for first individual only
        fit_index = genes[gene]
        for i in range(k+c):
            multiplier = 2**(i+1)
            epi_index = E[gene][i]
            # note: epi_index may be for any individual
            fit_index += multiplier * genes[epi_index]
        fitness += F[gene][fit_index]
    return fitness / n
```

B.5.7 Subset Sum Problem Generator

The subset sum problem (SSP) consists of a set W of n integers, and considers if a subset S of W can be selected that equals an integer value C. Although the implementation suggested here allows for the optimisation of a function $f(\mathbf{x})$, the problem formally requires a *decision* ('yes' or 'no') rather than optimisation.

The subset sum problem is a challenging *NP-hard* and *NP-complete*[8] domain, and it is a special case of the well known "knapsack" problem [126]. The *partition problem* is a special case of the subset sum problem and can be polynomially transformed into it [185].

One method of implementation is to randomly select n values for the set W from a pool of positive integers in the range $\{1, \ldots, 1000\}$. This arbitrary limit is consistent with previous work by [190] and other authors. The value for C is created from the summation of a random subset selection S of W.

A subset S of W can be defined as string of Boolean values, such that $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ where $x_i \in \{0, 1\}$. If $x_i = 1$ the corresponding w_i is considered part of the set S and included in the subset summation $P(\mathbf{x})$.

$$P(\mathbf{x}) = \sum_{i=1}^{n} w_i \cdot x_i \tag{B.34}$$

The function to be minimised is the difference between C and the summation of the selected subset $P(\mathbf{x})$. A conditional form is used to penalise infeasible solutions so they will never have a value better than feasible solutions, and the function $f(\mathbf{x}$ is always ≥ 0.

$$F(\mathbf{x}) = \begin{cases} C - P(\mathbf{x}) & \text{if } C - P(\mathbf{x}) \geq 0 \\ P(\mathbf{x}) & \text{otherwise} \end{cases} \tag{B.35}$$

[8] *NP-complete* problems (also known as NP-C or NPC) is a complexity theory class of problem. They are the most difficult problems in NP (non-deterministic polynomial time).

B.5.8 MAXCUT Maximum Cut Graph Problem

Consider an undirected graph $G = (V, E)$ where V denotes the set of n vertices and E the set of edges. Let the edges be weighted and there be no self-connected vertices. The maximum cut problem is to partition the set of vertices V into two disjoint sets V_0 and V_1 such that it maximises the sum of the weights edges from E that span V_0 and V_1.

This problem is also known to be *NP-complete*, and that the Satisfiability problem (SAT) can be polynomially transformed into it [185].

The partitions V_0 and V_1 can be represented by a string of Boolean values of a length equal to the number of vertices in V. If $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ where $x_i \in \{0, 1\}$, a Boolean value of 0 indicates the corresponding vertex of V belongs to subset V_0 and a value of 1 to the set of V_1.

A weighted edge value between vertices i and j is denoted by w_{ij}. The function to be maximised is the sum of edges that span both V_0 and V_1 partitions [190].

$$F(\mathbf{x}) = \sum i = 1^{n-1} \sum_{j=i+1}^{n} w_{ij} \cdot [x_i(1 - x_j) + x_j(1 - x_i)] \tag{B.36}$$

Instances of the problem can be constructed using a random graph (ER style) of n vertices and with an edge probability p that any pair of vertices will have an edge. Two well-known cases are "cut20-0.1" and "cut20-0.9" which contain $n = 20$ vertices each, and with $p = 0.1$ for a sparse graph and $p = 0.9$ for a dense graph respectively. Weight values for edges are random value in the range of $[0, 1]$. For example, a case of "cut20-0.9" it has been reported [190] to have a maximum fitness of 56.74007.

As an implementation note, the connectivity and the weight values can be stored in a single adjacency matrix or list, where 0 indicates no connection and a none zero value as the weight value. In a matrix form values for w_{ij} should be reflected to w_{ji} to match the undirected nature of the graph (at the expense of memory).

B.5.9 Error Correcting Code Design

The challenge of designing codes has strong application to any digital communication or storage system. One essential feature identified early in the field of Error Correcting Codes (ECC) [221] is that a code with as much space as possible between codewords will be more robust to transcription errors. This in turn minimises the length of transmitted messages required to provide maximal correction of single uncorrelated bits. So, it is desirable to create codes with exactly this property of maximal distance between codewords.

Although formal design and incremental processes can be used to create error correcting codes, it is an interesting search space for discrete search algorithms. Many techniques have been applied including genetic algorithms [89], repulsion algorithms, particle swarm (using repulsion) and simulated annealing [125] and other hybrid forms [55].

The Binary code form considered here is created with parameters of (n, M, d) where n is the number of bits in each codeword, M the number of codewords required in the code, and d the minimum Hamming distance between any pair of codewords for the code to be valid. A good code contains a minimal n with a maximum possible d value for the required number of words M.

We can use a minimal-energy style measure of how well M words are placed in the corners of an n-dimensional code space. This is based on the summation of Hamming distances between all pairs of codewords.

$$F(C) = \frac{1}{\sum_{i=1}^{M} \sum_{j=1, i \neq j}^{M} d_{ij}^{-2}} \tag{B.37}$$

where d_{ij} is the Hamming distance between codewords i and j in code C.

One simplification is to search for $M/2$ codewords and create a complement[9] of each codeword for the full set of M codewords. This creates a considerably smaller search space and incorporates knowledge of the problem domain to reduce the search.

As an implementation consideration this means that for any $M/2$ set of codewords to be valid there must not only be the minimal separation distance d between all codewords but also that no complementary pairs exist in the $M/2$ set. Some search algorithms can utilise the invalid regions of search space, however a penalty would be needed if duplicate (invalid) codewords existed in a potential solution as the minimum energy formula is undefined (divide by zero) for any $d_{ij} = 0$ pair.

[9] For example with two binary strings the complement of 0110 is simply 1001.

B.5.10 Minimum Tardy Task Problem

The Minimum Tardy Task Problem (MTTP) is a single processor task scheduling problem. The description presented here for the problem is based on that presented by Khuri et al. in [190] which refers to the work of Stinson [329].

Consider a set T of n tasks where each task i contains three positive integer numbers for the task length l_i, task deadline d_i and task penalty (or weight) w_i. The objective is to select a subset S of T that represents a feasible schedule and which minimises the total penalty of all unallocated tasks.

This problem is very similar to a "maximum profit" schedule problem where the weight of allocated tasks represents profit which is to be maximised.

The subset S of T can be represented by a simple binary string. If $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ for each task in T with each $x_i \in \{0, 1\}$. a Boolean value of 1 for x_i indicates the task is in the set S and $x_i = 0$ that the task is absent from S.

Implementation style is divided into two groups; those that only evaluate valid (*feasible*) subsets, and those that allow invalid (not feasible or *infeasible*) subsets with penalty. If we evaluate a known valid solution, the sum of tardy tasks to be minimised is simply:

$$F(\mathbf{x}) = \sum_{i \in T-S} w_i \tag{B.38}$$

The inverse of this value can be used as a maximisation case.

Note that S does not explicitly indicate the order of tasks, and not all permutations would be feasible due to task deadline constraints. However, determining the feasibility of S is considerably easier than testing all $k!$ permutations of task order. It can be shown [329] that S is feasible *if and only if* tasks can be ordered by increasing task deadlines without violating any individual deadline. Other permutations may be valid, but this rule must still hold. So by pre-ordering all tasks in T by deadline, and maintaining this order for any subset S selected, the evaluation of feasibility is programatically a simple and quick sequence of task length additions and deadline constraint checks.

If infeasible solutions are also to be evaluated a penalty can be applied with a partial value for the valid parts of the solution [190]. A penalty value equal to the sum of all task penalties is used so that feasible solutions will always be better than infeasible one. Evaluation considers each task of S in turn, and rejects any that make the current schedule infeasible from the set S creating S'. The rejected tasks are counted as tardy tasks.

$$F(\mathbf{x}) = \begin{cases} \sum_{i \in T-S} w_i & \text{if } S \text{ is feasible} \\ \sum_{i \in T-S'} w_i + \sum_{i \in T} w_i & \text{otherwise} \end{cases} \tag{B.39}$$

where S' represents the feasible version of the infeasible S subset.

Task	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Length	2	4	1	7	4	3	5	2	4	7	2	9	8	6	1	4	9	7	8	2
Deadline	3	5	6	8	10	15	16	20	25	29	30	36	49	59	80	81	89	97	100	105
Weight	15	20	16	19	10	25	17	18	21	17	31	2	26	42	50	19	17	21	22	13

Table B.6: The "mttp20" standard minimum tardy task problem of $n = 20$ with a known global minima of $f(\mathbf{x}^*) = 41$ [190].

As a classic reference Table B.6 presents "mttp20" which contains 20 tasks and an allocation of task length, deadline and penalty values[190]. This is known to have a global minimum of $f(\mathbf{x}^*) = 41$ or the inverse maximum of $1/41 \approx 0.02439$.

B.5.11 Max Set of Gaussians Landscape Generator

The Max Set of Gaussians (MSG) landscape generator was specifically designed by Gallagher and Yuan [124] to "increase the research value of experimental studies" using a minimal but effective set of parameters. Two key features of this type of generator are that a large number of similar instances can be created (landscape classes), and that insights into algorithm behaviour can be gained by relating results to generated landscape structure.

Multivariate Gaussian functions are used as the basic building blocks of this landscape generator.

$$g(\mathbf{x}) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \cdot e^{\left(-\frac{1}{2}(\mathbf{x}-\mu)^T \cdot \Sigma^{-1} \cdot (\mathbf{x}-\mu)\right)} \tag{B.40}$$

To calculate the fitness of a vector $\mathbf{x} = \{x_1, x_2, \ldots, x_N\}$ for a set of m weighted components, the largest component value is used.

$$F(\mathbf{x}) = \max_i \omega_i \cdot g_i(\mathbf{x}) \tag{B.41}$$

The weight component ω_i can also include the constant normalisation terms for each Gaussian factor, so that the final form can be:

$$F(\mathbf{x}) = \max_i \omega_i \cdot e^{\left(-\frac{1}{2n}(\mathbf{x}-\mu)^T \cdot \Sigma^{-1}(\mathbf{x}-\mu)\right)} \tag{B.42}$$

where (by probability convention) σ is the covariance matrix and μ is the expected value for each Gaussian component.

n	dimensionality of landscape
m	number of Gaussian components
$\mathbf{D} = (l, u)$	lower and upper range for component locations
p	value for the global optimum peak
r	ratio of local optima to global value

Table B.7: Parameters used for a standard implementation of the Max Set of Gaussians landscape generator.

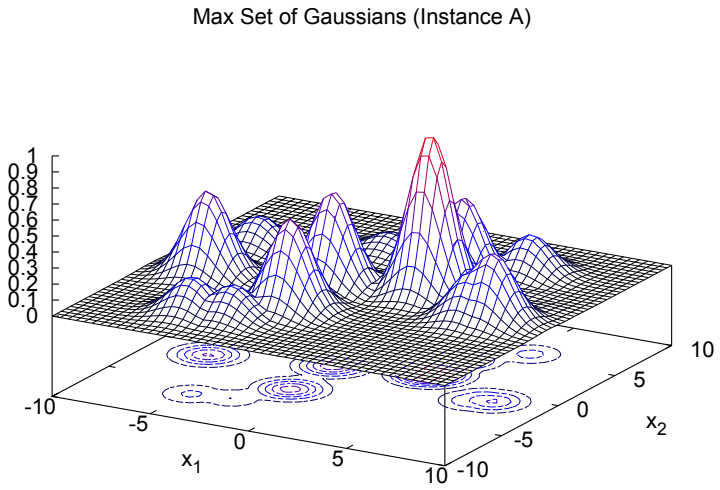
Table B.7 lists the parameters that can be modified in a standard implementation. Similarly, it is possible to set parameters for distribution of mean vectors and other aspects of the initialisation. However the parameters shown support a substantial amount of variation. It is also possible to "seed" the landscape with components in specific locations in order to create specific landscape features such as gaps.

In Figure B.20 are two random instances of the landscape each with different numbers of components and different ratio values of optimal peak to subpeak heights. Ridges can be formed due to the max function and this can be a problem for some gradient based techniques. Some Gaussian components are hidden by other larger or overlapping components.

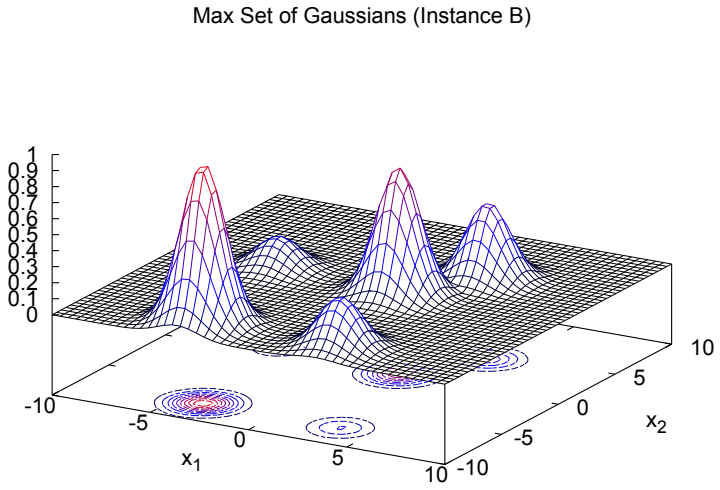
Implementation details for Matlab software have been provided by the originating authors.[10] This has been converted into Python code using the NumPy[11] numerical module to provide the matrix based operations needed. It would be challenging to implement this generator in a language without good support for matrix operations.

[10] See http://www.itee.uq.edu.au/~marcusg/msg.html for descriptions, examples, references and source code.

[11] See http://numpy.scipy.org/



(a) 20 components, ratio=0.6



(b) 5 components, ratio=0.8

Figure B.20: Two configurations of the Max Set of Gaussians landscape. Both are two dimensional instances with a peak global value of 1.0 and Gaussian component centres in the range of $[-8, 8]$. In a) there are 20 components and a ratio of 0.6, and b) has only 5 components and a ratio of 0.8.

Appendix C

Topology Survey

C.1 Introduction, Measures and Details

The intent of this appendix is to present a detailed survey of topologies that are central to, or components of, the investigations presented in the thesis body.

Graph properties and details are presented for a range of graphs types, organised in groups. The order begins with Full graphs, deterministic regular Lattices, Tree and Star graph models, and then moves on to stochastic random Erdös-Rényi (ER) models, the small-world growth model of Watts and Strogatz (WS), the Barabási-Albert (BA) scale free growth model, and the Merge-Regenerate (MR) model.

The title of each graph group or type includes characters used to abbreviate the topology type and features. For example, "L.hk4.b" represents a regular "Lattice" (L) graph with a hollow (h) "k4" neighbourhood size, and bounded (b) (non-circular) dimensions.

The potential complexity and variance of graph model parameters makes a complete survey of all possible graph instances impossible. It is hoped, however, that this survey provides a supportive and comparative explanation of the known and expected properties of the graph instances selected for use in experiments.

Basic notation of $G(n, m)$ is used for each graph topology G of n vertices and m edges. For each topology instance considered, there are three common assessment and presentation forms:

- Table of properties and statistics for the specified topology type for three (or more) standard graph sizes.
- Two dimensional layout image (or sample of images if presentation is variable) to visually represent the graph model topology.
- Sets of histogram plots for vertex degree and path length; one set for each standard scale.

Addition details, summaries and relevant comments are provided to assist in the comparison of topology instances.

The three standard scale values selected are $n = 100$, $n = 400$ and $n = 900$. If it is not possible for a topology model to exactly match these standard sizes, the nearest suitable value for n is selected. The selection of these values is based on the relationship in two dimensional lattices with equal dimensional sizes of 10×10, 20×20 and 30×30.

Each table of properties and statistics presents the following details which are all discussed in the thesis body and in the glossary:

- Number of vertices n.

- Number of edges m.
- Components of the graph (if a graph is disconnected).
- Diameter of the graph.
- Girth of the smallest cycle (if possible).
- Density of the graph, normalised.
- Mean vertex degree $\langle k \rangle$.
- Mean path length L.
- Clustering coefficient (transitivity) C.
- The normalised global efficiency of the graph E_{glob}.
- The normalised local efficiency of the graph E_{loc}.

For a single deterministic topology the values of a single graph instance are used. If a graph is stochastic, a sample (usually 30) of graph instances are created and the average values used where appropriate. All graph models are treated as undirected.

Page breaks have been intentionally added in order to present each new topology on a new page.

C.2 Full

Synopsis: A full graph topology in which every vertex is directly connected to all other vertices in the graph.

This is also referred to as a "panmictic" topology in the thesis. Basic properties and statistics are shown in Table C.1 for four different instances of increasing vertex number. Note the exponential increase in edges and the unity values for mean path length, clustering, efficiency and density. The graph is a single component, the mean degree $\langle k \rangle$ is defined as $n - 1$, triangles resulting in a minimal girth of 3, and the mean path length is quite simply 1 as there are direct paths to all other vertices.

A simple circular layout for a small $n = 20$ full graph is shown in Figure C.1. Even for a full graph of only $n = 100$ vertices, the edge density in an image makes edges indistinguishable.

The total number of edges m in a full undirected graph is $n(n-1)/2$, where n is the number or vertices. This exponential growth relationships means that any process or function that scales with respect to the number of edges will be exponentially impacted as the graph increases in vertex number. For example, the measure of local efficiency E_{loc} calculated empirically will take a very long time (although by definition it is simply 1 for a full graph), as every local subgraph considered is also a full graph of size $n' = n - 1$.

As a note on implementation, it makes little sense to explicitly define edges for a full graph if there are no variations in edge attributes (weights). A list of vertices is simpler and requires minimal computation and memory resources.

Property	n100	n200	n400	n900
n	100	200	400	900
m	4950	19900	79800	404550
Components	1	1	1	1
Diameter	1	1	1	1
Girth	3	3	3	3
Density	1.000	1.000	1.000	1.000
$\langle k \rangle$	99	199	399	899
L	1.000	1.000	1.000	1.000
C	1.000	1.000	1.000	1.000
E_{glob}	1.000	1.000	1.000	1.000
E_{loc}	1.000	1.000	1.000	1.000

Table C.1: Properties and statistics for full graph instances.

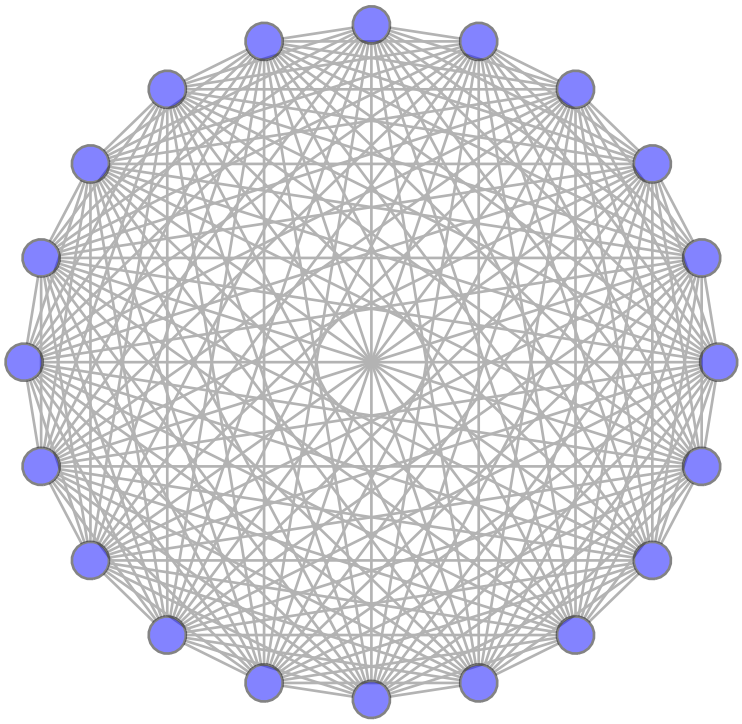


Figure C.1: Circle layout for a small $n = 20$ full graph.

C.3 Lattice (L)

C.3.1 Introduction

The presentation of the many lattice types deserves a quick introduction on both dimensional boundary issues, neighbourhood structures and the influence of rewiring.

Firstly, all the lattices selected in this survey are two dimensional (2D) in construction and presentation. It is common in for lattices graphs to be implemented with circular dimensional topology, in which case the "surface" of a lattices can be considered toroidal. Not only are such circular dimensional behaviour is rare in natural system, but bounded (non-circular) graphs present interesting changes in topology properties along or near bound edges, suggesting that they should be included and investigated rather than avoided.

Secondly, there are a number of different neighbourhood structures used, each denoted by the number of edges k a single vertex will have to other neighbours. As a reference the first three regular forms are presented in Figure C.2. Note that, in non-circular lattices, vertices located along boundaries will not contain the median vertex degree.

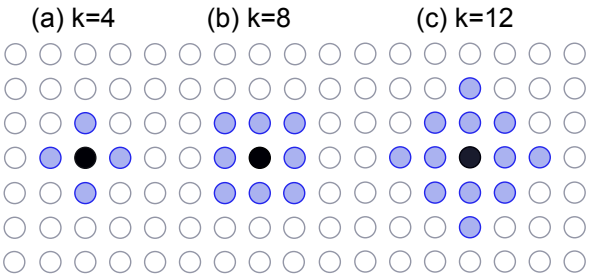


Figure C.2: Neighbourhoods for (a) $k = 4$, (b) $k = 8$ and (c) $k = 12$. Also known as the Von Neumann, Moore and Extended Moore neighbourhoods respectively.

Finally, as discussed in the body of the thesis, there are interesting transitions or "phase changes" in the properties of regular graphs as they are rewired by differing amounts. For each of the circular lattice forms selected, a sample of rewired instances is also evaluated for changes in normalised mean path length L/L_0 and clustering coefficient C, and the similar measures of global and local efficiency (denoted E_{glob} and E_{loc} respectively).

C.3.2 L.k4

Synopsis: A two-dimensional (2D) lattice where each vertex is connected to axes neighbours ($nei = 1$) resulting in a vertex degree of $k = 4$.

This type of vertex connection topology is also commonly known as a Von Neumann neighbourhood. The lattice axes in this topology are circular and create a toroidal space. See Section C.3.3 for the bound (non-circular) of this base topology type.

Property	n100	n400	n900
n	100	400	900
m	200	800	1800
Components	1	1	1
Diameter	10	20	30
Girth	4	4	4
Density	0.040	0.010	0.004
$\langle k \rangle$	4.000	4.000	4.000
L	5.051	10.025	15.017
C	0.000	0.000	0.000
E_{glob}	0.384	0.222	0.157
E_{loc}	0.000	0.000	0.000

Table C.2: Properties and statistics for L.k4 graph instances.

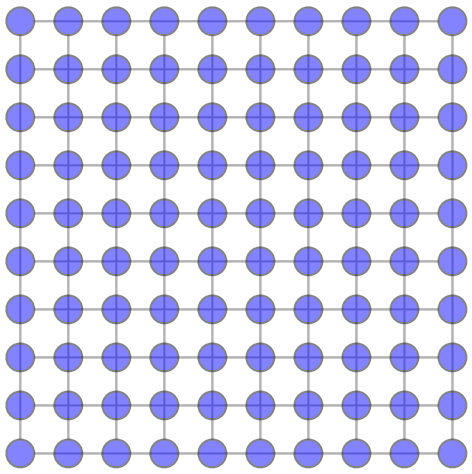


Figure C.3: 2D layout for a 10×10 L.k4 circular lattice.

Basic properties and statistics are shown in Table C.2 for three different instances selected for a linear increase in dimension size. They are 10×10 for $n = 100$, 20×20 for $n = 400$ and 30×30 for $n = 900$. The 10×10 ($n = 100$) circular layout is shown in Figure C.3.

Histograms for vertex degree and path length for each size are shown in Figure C.4, Figure C.5 and Figure C.6 respectively. Note that vertex degree is always $k = 4$ and also the triangular shape of the path length histogram. The most expensive paths are those that travel diagonally as they must take a Manhattan route.

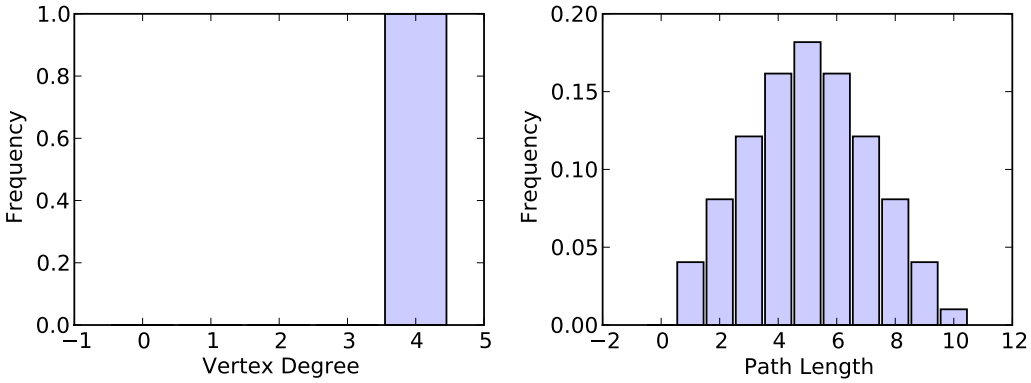


Figure C.4: Vertex degree and path length histograms for L.k4, $n = 100$.

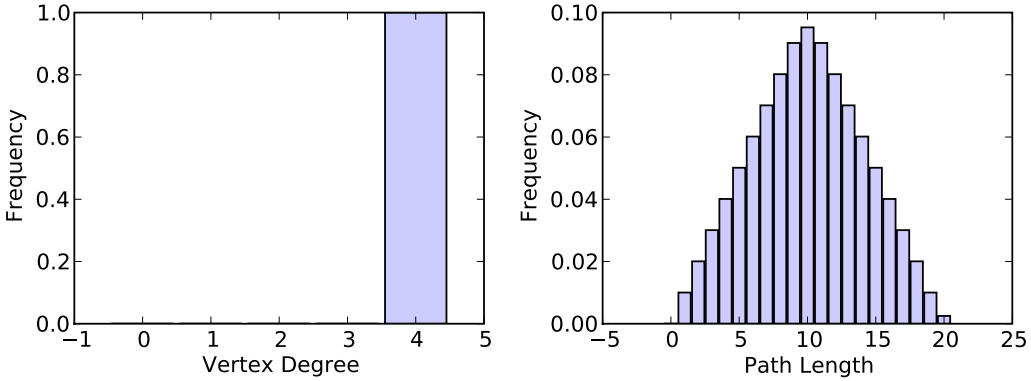


Figure C.5: Vertex degree and path length histograms for L.k4, $n = 400$.

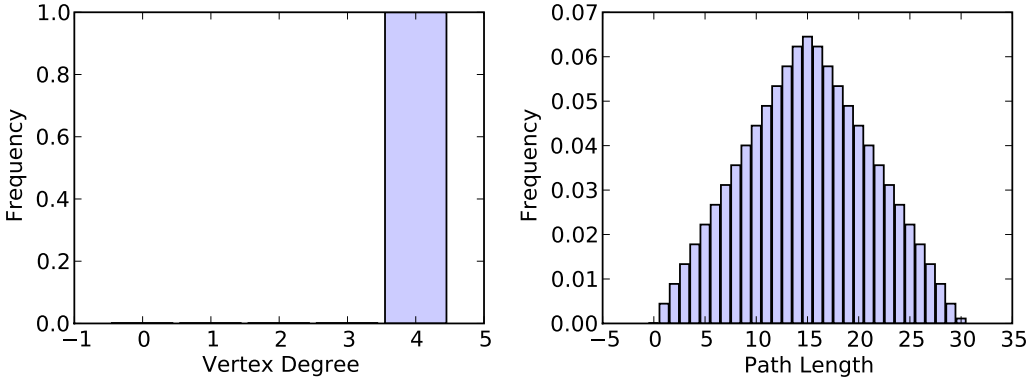


Figure C.6: Vertex degree and path length histograms for L.k4, $n = 900$.

C.3.3 L.k4.b

Synopsis: A version of the 2D L.k4 lattice (Section C.3.2) that has fixed boundaries, and so is not circular.

Note that unlike the L.k2 topology, the diameter of each L.k4.b lattice does not match the specified dimension size, and that the vertices located on the boundaries introduce degrees of $k = 3$ (sides) and $k = 2$ (corners) into the vertex degree histogram. The path length histogram is no longer the triangle shape of the circular case, but rather a skewed parabolic distribution.

Property	n100	n400	n900
n	100	400	900
m	180	760	1740
Components	1	1	1
Diameter	18	38	58
Girth	4	4	4
Density	0.036	0.010	0.004
$\langle k \rangle$	3.600	3.800	3.867
L	6.667	13.333	20.000
C	0.000	0.000	0.000
E_{glob}	0.329	0.189	0.133
E_{loc}	0.000	0.000	0.000

Table C.3: Properties and statistics for L.k4.b graph instances.

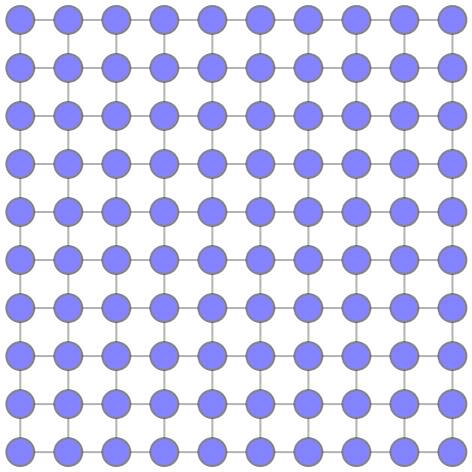


Figure C.7: 2D layout for a 10×10 L.k4.b non-circular lattice.

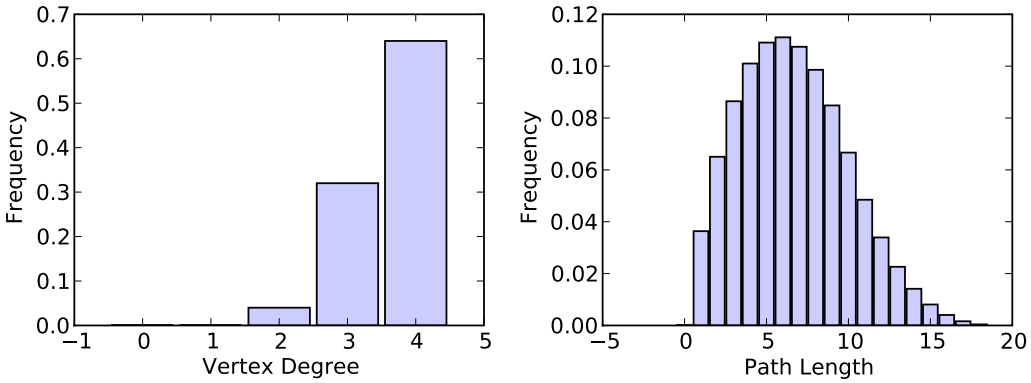


Figure C.8: Vertex degree and path length histograms for L.k4.b, $n100$.

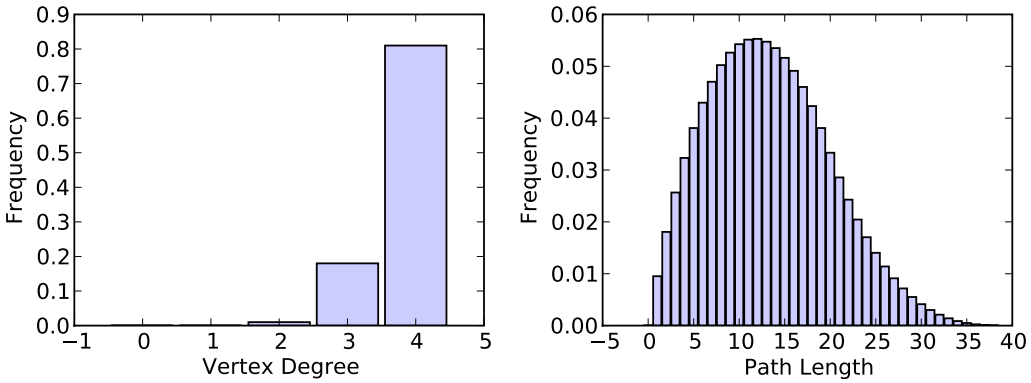


Figure C.9: Vertex degree and path length histograms for L.k4.b, $n = 400$.

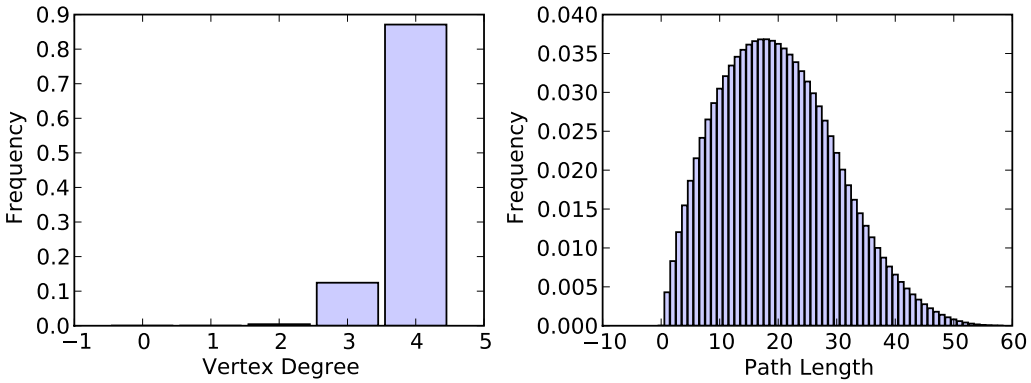


Figure C.10: Vertex degree and path length histograms for L.k4.b, $n = 900$.

C.3.4 Rewired L.k4

Synopsis: Rewired circular 2D L.k4 lattices of size $n = 400$ to present the influence of rewiring on graph properties.

For each rewiring probability $p > 0$ a sample of 30 instances is used. In all cases $n = 400$, $m = 800$, the density remains 0.010 and the mean degree $\langle k \rangle = 4$.

As the degree of rewiring is increased, it can be seen in Figure C.11 that that normalised mean path length L/L_0 drops from its expensive full lattice high. L_0 is the value of L for $p = 0.00$ shown in Table C.4. The use of a log scale for p in Figure C.11 shows an initial linear decrease in L/L_0 and a slight plateau as it approaches $p = 1.0$.

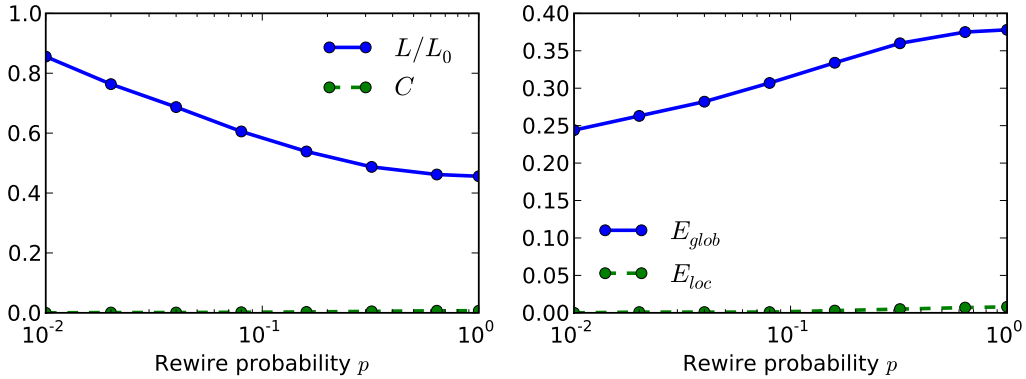
Clustering C is initially zero as a regular lattice of this type has no neighbourhood triangles, however when rewiring has a greater degree of influence the girth value decreases from its initial 4.0 down to the minimal value of 3.0 indicating that triangles are created.

Note that when the degree of rewiring is high ($p \geq 0.32$) the number of components begins to rise above 1.0 indicating isolated sub-graphs may sometimes be created.

The measures of global and local efficiency also reflect same changes as those indicated by L/L_0 and C, however without the need for normalisation or special-case handling for disconnected graphs.

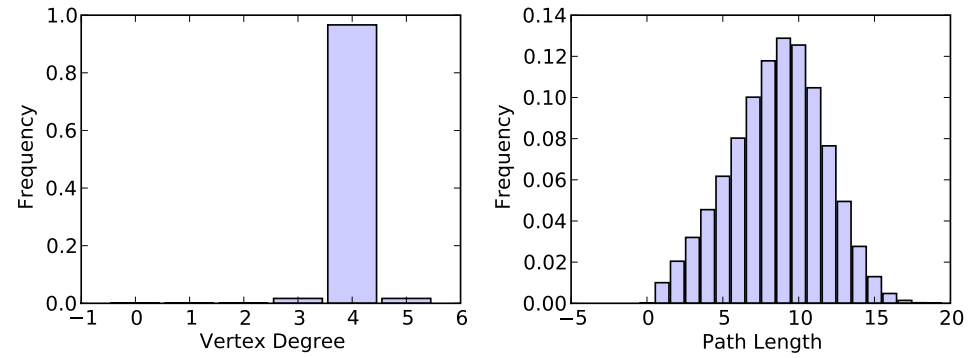
Property	k4.00	k4.01	k4.02	k4.04	k4.08	k4.16	k4.32	k4.64	k4all
p	0.00	0.01	0.02	0.04	0.08	0.16	0.32	0.64	1.00
Components	1.000	1.000	1.000	1.000	1.000	1.000	1.050	1.050	1.250
Diameter	20.000	17.400	14.950	12.850	11.150	9.700	8.900	8.250	8.550
Girth	4.000	3.950	3.750	3.600	3.400	3.100	3.000	3.000	3.000
L	10.025	8.577	7.655	6.887	6.071	5.401	4.885	4.630	4.572
C	0.000	0.000	0.001	0.001	0.002	0.003	0.005	0.007	0.007
E_{glob}	0.222	0.244	0.263	0.282	0.307	0.334	0.360	0.375	0.378
E_{loc}	0.000	0.000	0.001	0.001	0.001	0.003	0.005	0.007	0.008

Table C.4: Properties and statistics for rewired L.k4 lattice instances.

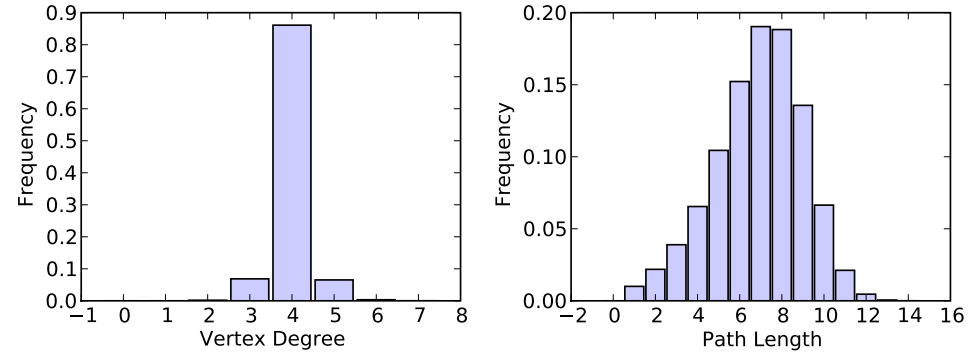


(a) (b)

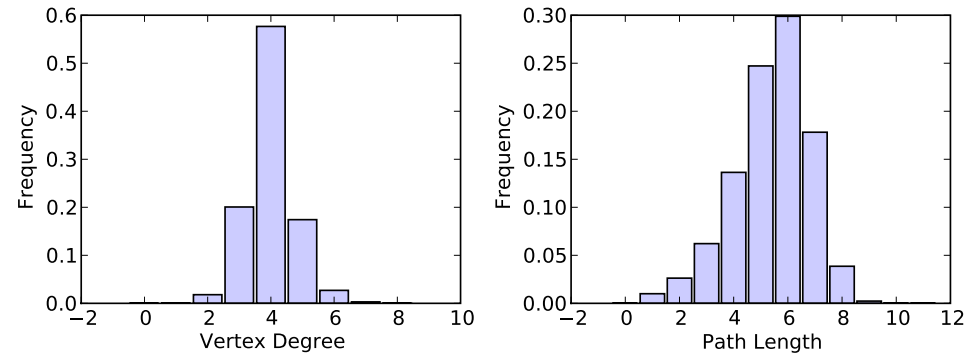
Figure C.11: Influence of p on (a) L/L_0 and C, and on (b) E_{glob} and E_{loc}.



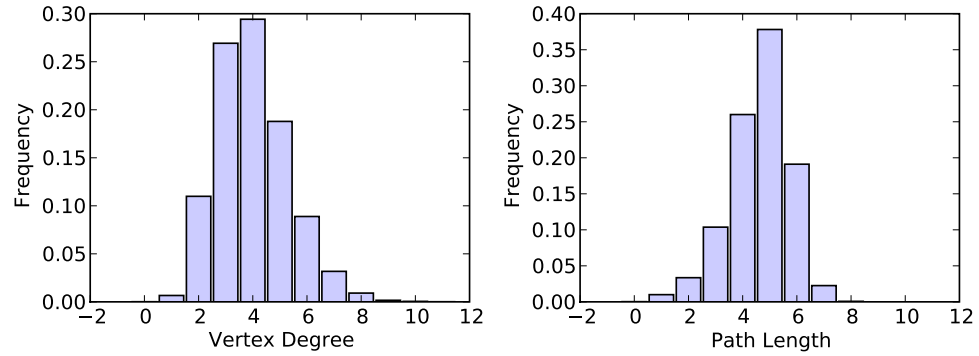
(a) $p = 0.01$



(b) $p = 0.04$



(c) $p = 0.16$



(d) $p = 0.64$

Figure C.12: Vertex degree and path length histograms for rewired L.k4 lattices.

C.3.5 L.k8

Synopsis: A two dimensional circular lattice with the same vertex edge profile as the L.k4 lattice (horizontal and vertical neighbours) and four additional edges to diagonal neighbours – a total of $k = 8$ edges per vertex.

This type of vertex connection topology is also known as a Moore neighbourhood. Not only are there more edges, but local vertex neighbourhoods overlap which is interesting with respect to processes and mixing of information in a graph.

Property	n100	n400	n900
n	100	400	900
m	400	1600	3600
Components	1	1	1
Diameter	5	10	15
Girth	3	3	3
Density	0.081	0.020	0.009
$\langle k \rangle$	8.000	8.000	8.000
L	3.384	6.692	10.017
C	0.429	0.429	0.429
E_{glob}	0.503	0.301	0.216
E_{loc}	0.798	0.798	0.798

Table C.5: Properties and statistics for L.k8 graph instances.

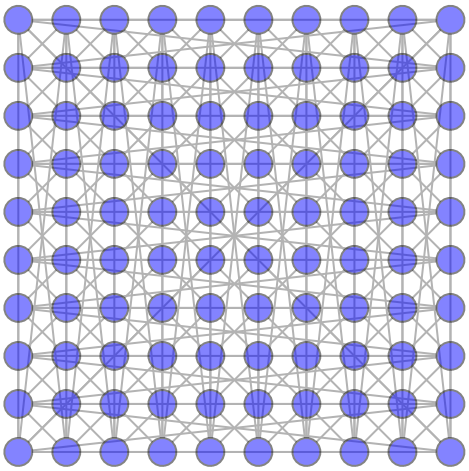


Figure C.13: 2D layout for L.k8 circular lattice.

In comparison to a L.k4 lattice, the L.k8 lattice has a higher density (cost), but also cycles of length = 3 (triangles, or a girth of 3). As a result, the clustering and local efficiency measures are no longer zero as in the L.k4 and L.k4.b cases. The diameter is halved (by the elimination of Manhattan routes for diagonal paths) and the mean path length $\langle k \rangle$ drops by $\approx 1/3$ and so the overall global efficiency increases. The path length histogram shape is essentially a half triangle.

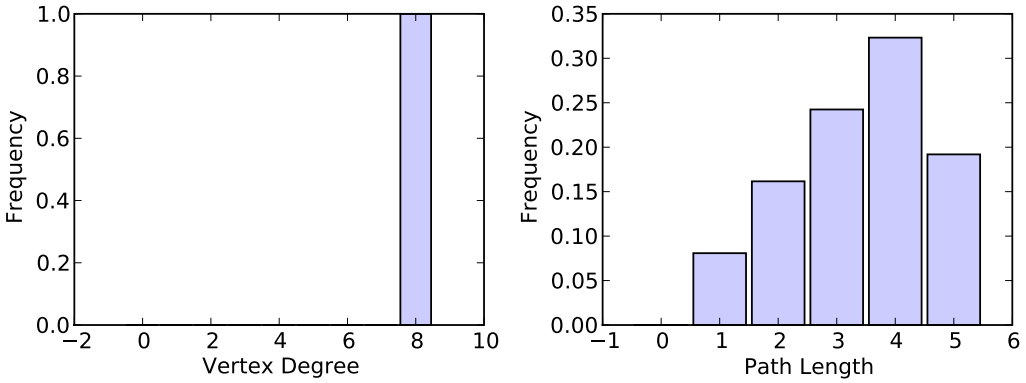


Figure C.14: Vertex degree and path length histograms for L.k8, $n = 100$.

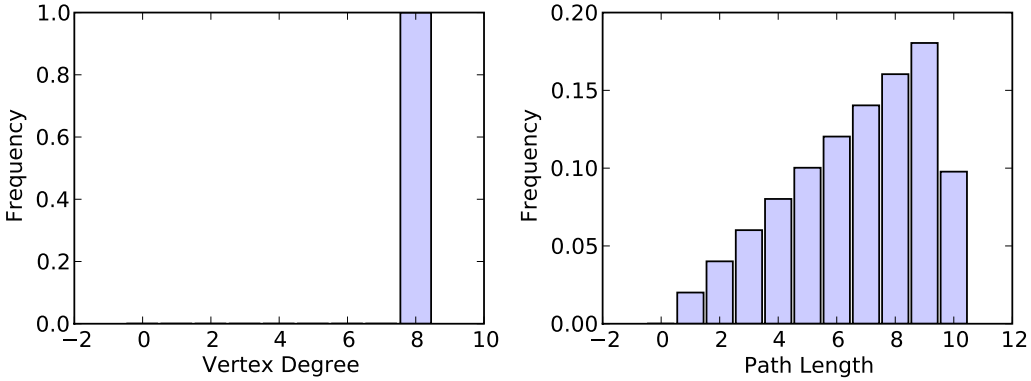


Figure C.15: Vertex degree and path length histograms for L.k8, $n = 400$.

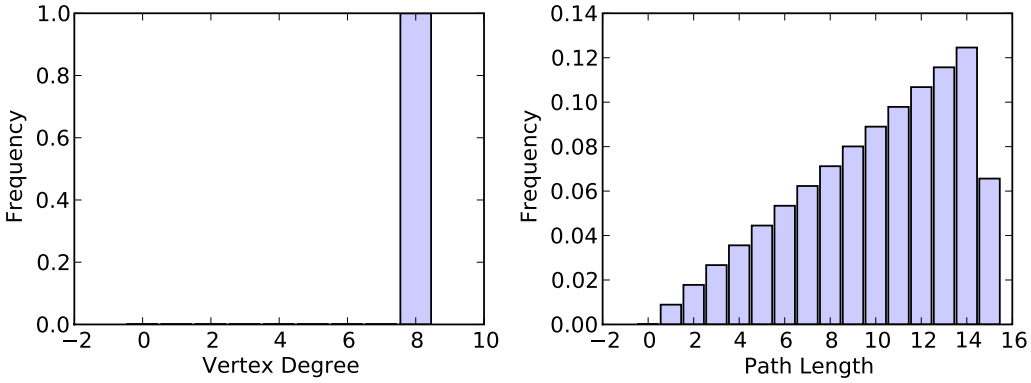


Figure C.16: Vertex degree and path length histograms for L.k8, $n = 900$.

C.3.6 L.k8.b

Synopsis: A non-circular (bound) version of the L.k8 2D lattice.

The degree of side and corner vertices is $k = 5$ and $k = 3$ respectively. The overall path length histogram, as in the non-circular L.k4.b lattice, resembles a slightly skewed parabolic distribution.

In comparison to the circular L.k8 lattice the density drops slightly, diameter nearly doubles and mean path length increases by $\approx 1/3$; the global efficiency decreases slightly. Clustering and local efficiency increase slightly as boundary side vertices are more "efficient" (with a higher ratio of the available triangles) then general vertices.

Property	n100	n400	n900
n	100	400	900
m	342	1482	3422
Components	1	1	1
Diameter	9	19	29
Girth	3	3	3
Density	0.069	0.019	0.008
$\langle k \rangle$	6.840	7.410	7.604
L	4.680	9.340	14.004
C	0.458	0.442	0.437
E_{glob}	0.423	0.251	0.179
E_{loc}	0.828	0.812	0.807

Table C.6: Properties and statistics for L.k8.b graph instances.

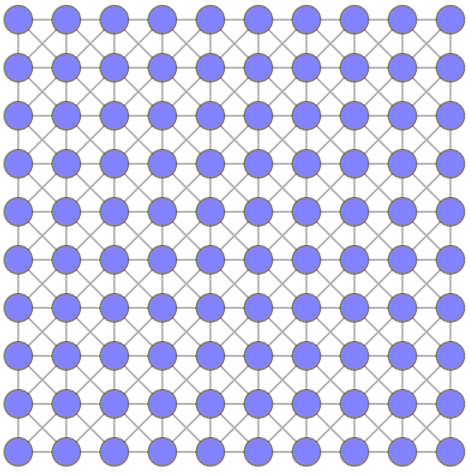


Figure C.17: 2D layout for L.k8 non-circular lattice.

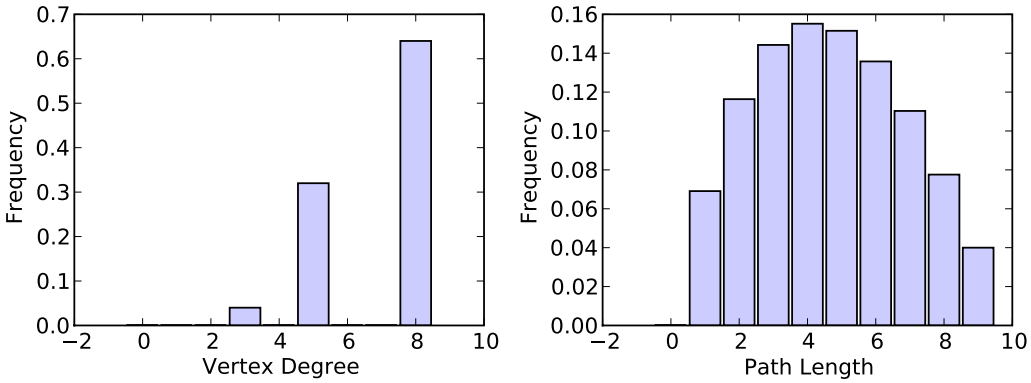


Figure C.18: Vertex degree and path length histograms for L.k8.b, $n = 100$.

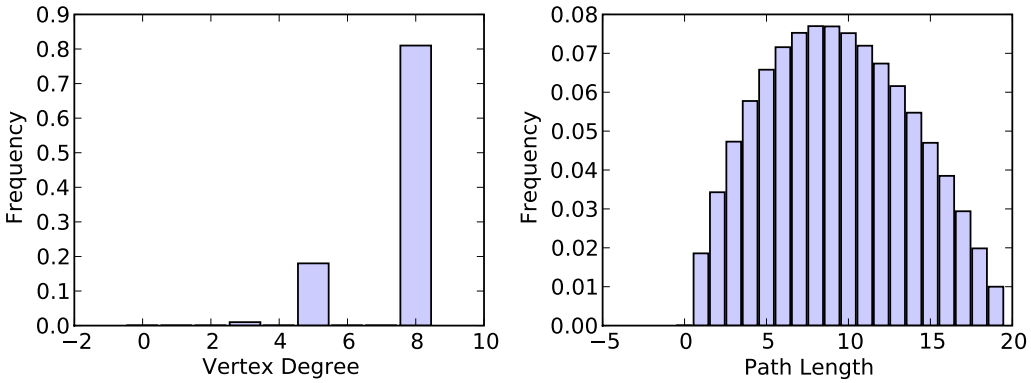


Figure C.19: Vertex degree and path length histograms for L.k8.b, $n = 400$.

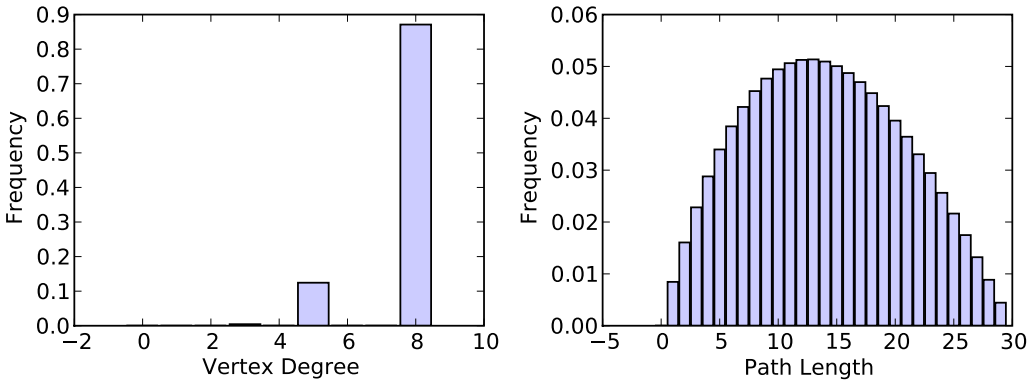


Figure C.20: Vertex degree and path length histograms for L.k8.b, $n = 900$.

C.3.7 Rewired L.k8

Synopsis: Rewired circular L.k8 lattices of size $n = 400$.

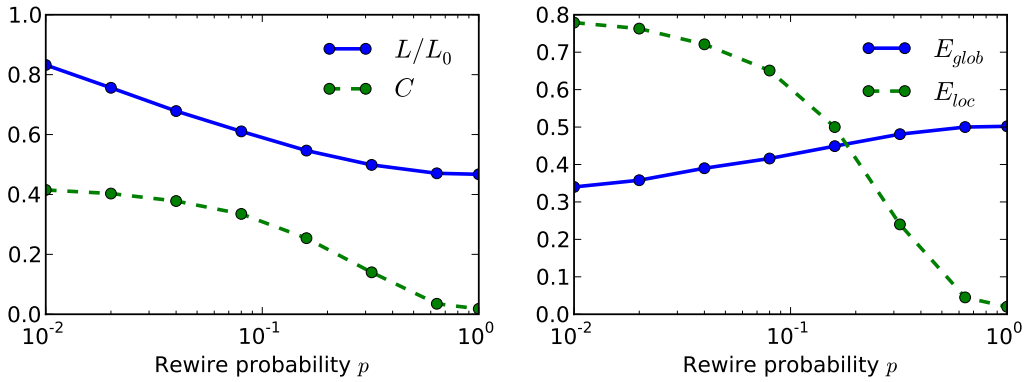
In all cases $n = 400$, $m = 1600$, the density is therefore 0.020 and the mean degree $\langle k \rangle = 8$. As in the rewired L.k4 study, for each rewiring probability $p > 0$ a sample of 30 instances is used.

With the presence of edges aligned along diagonal as well as horizontal and vertical axes, the Diameter is initially only 10 units and due to triangle structure the Girth is the minimal 3. As a result, there is a very high level of clustering C which translates to a strong local efficiency of 0.798.

With a greater density of edges the influence of rewiring is increased, as rewiring is the probability that each individual edge is rewired. Figure C.21 shows the dramatic changes in both mean path length and clustering coefficient C which, unlike the rewiring of the L.k4 lattice, is initially quite strong and reduces only after significant rewiring occurs.

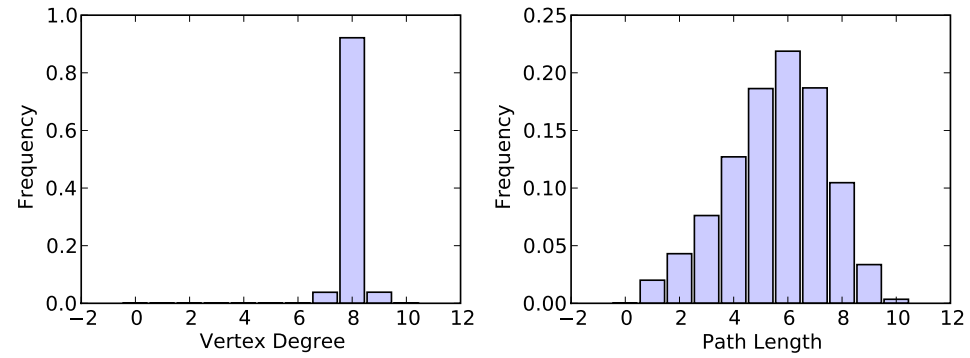
Property	k8.00	k8.01	k8.02	k8.04	k8.08	k8.16	k8.32	k8.64	k8all
p	0.00	0.01	0.02	0.04	0.08	0.16	0.32	0.64	1.00
Components	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Diameter	10.000	10.000	9.300	8.000	7.050	6.000	5.200	5.000	5.050
Girth	3.000	3.000	3.000	3.000	3.000	3.000	3.000	3.000	3.000
L	6.692	5.570	5.060	4.540	4.086	3.657	3.337	3.149	3.127
C	0.429	0.415	0.403	0.378	0.335	0.254	0.140	0.035	0.018
E_{glob}	0.301	0.340	0.358	0.390	0.416	0.449	0.481	0.500	0.502
E_{loc}	0.798	0.779	0.763	0.721	0.651	0.500	0.240	0.045	0.020

Table C.7: Properties and statistics for rewired L.k8 lattice instances.

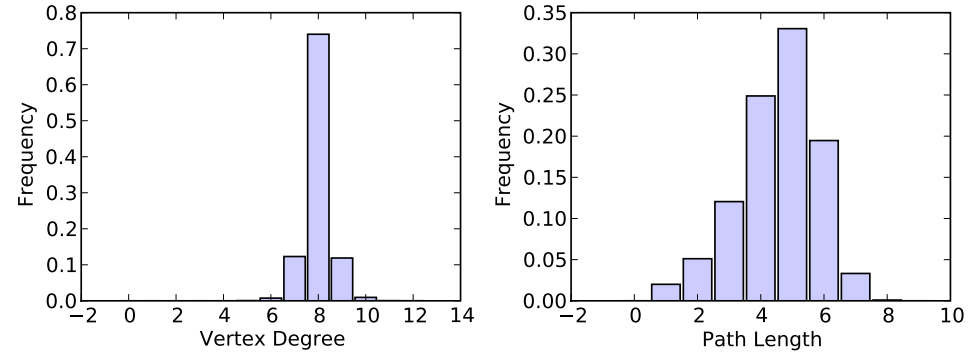


(a) (b)

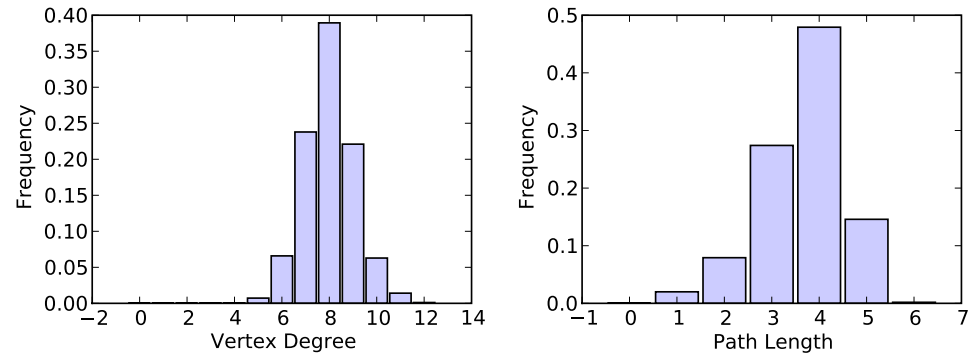
Figure C.21: Influence of p on (a) L/L_0 and C, and on (b) E_{glob} and E_{loc}.



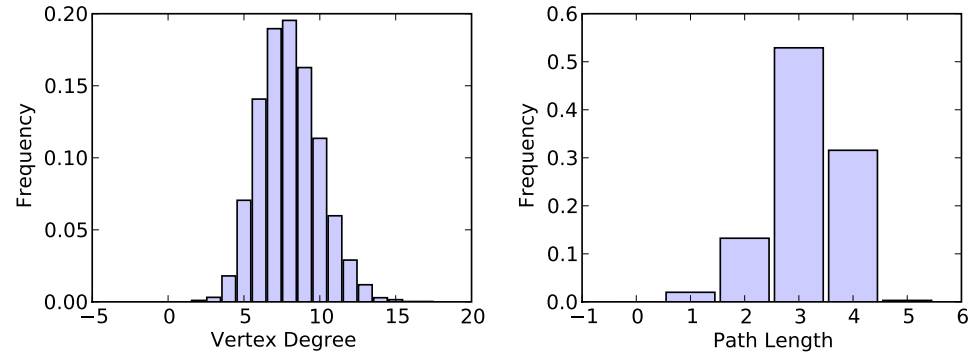
(a) $p = 0.01$



(b) $p = 0.04$



(c) $p = 0.16$



(d) $p = 0.64$

Figure C.22: Vertex degree and path length histograms for rewired L.k8 lattices.

C.3.8 L.k12

Synopsis: An extension of the L.k8 lattice, adding four additional edges to neighbours at a distance $nei = 2$ along the horizontal and vertical axes.

This type of vertex connection topology is also known as an Extended Moore neighbourhood. As in the progression from the L.k4 to L.k8 lattice, the additional edges create a greater amount of overlap with other vertex neighbourhoods, but as L.k12 extensions are only along the horizontal and vertical axes, the effective diameter (limited by diagonal diameter) is actually the same as the L.k8 topology.

In comparison to the L.k8 lattice, overall global efficiency is increased due to the increased density, lower mean path length and increased clustering. The path length histogram reflects the similar triangular shape.

Property	n100	n400	n900
n	100	400	900
m	600	2400	5400
Components	1	1	1
Diameter	5	10	15
Girth	3	3	3
Density	0.121	0.030	0.013
$\langle k \rangle$	12.000	12.000	12.000
L	2.778	5.263	7.759
C	0.455	0.455	0.455
E_{glob}	0.579	0.368	0.271
E_{loc}	0.813	0.813	0.813

Table C.8: Properties and statistics for L.k12 graph instances.

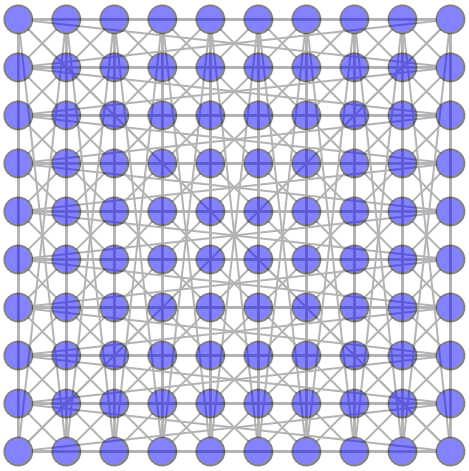


Figure C.23: 2D layout for L.k12 circular lattice. Although the appearance looks very similar to layout for a L.k8 lattice, note that the extended neighbourhood of each vertex overlaps to a greater degree.

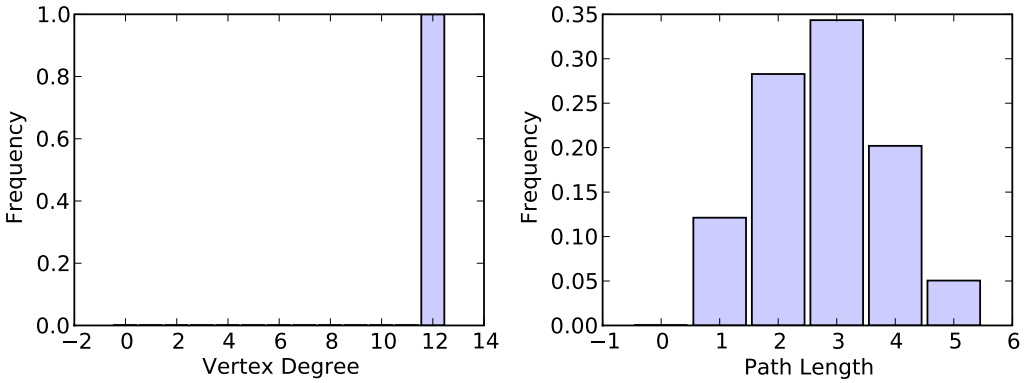


Figure C.24: Vertex degree and path length histograms for L.k12, $n = 100$.

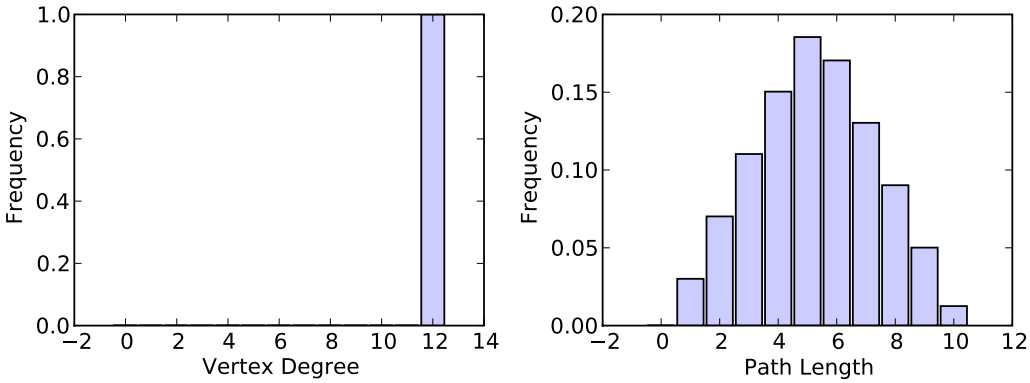


Figure C.25: Vertex degree and path length histograms for L.k12, $n = 400$.

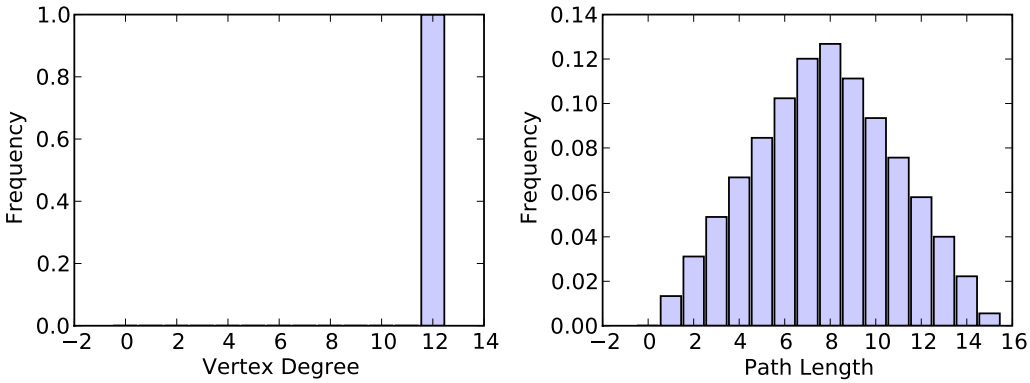


Figure C.26: Vertex degree and path length histograms for L.k12, $n = 900$.

C.3.9 L.k12.b

Synopsis: A non-circular (bound) version of the L.k12 lattice.

The differences of the bound L.k12.b lattice are similar in nature to the previous circular to non-circular lattice comparisons. However, because of the larger base vertex degree of $k = 12$, the side, corner and adjacent side/corner vertices add degrees of $k = [5, 7, 8, 10, 11]$ to the vertex degree histogram.

Global efficiency decreases as diameter approximately doubles, density drops and the mean path length increases. Local efficiency increases slightly due to higher cluster ratios of the side, corner and adjacent vertices (as seen in the L.k8.b profile).

The path length histogram has a basic parabolic form with an extended right tail due to the exotic range of vertex degrees present in side and diagonal paths.

Property	n100	n400	n900
n	100	400	900
m	502	2202	5102
Components	1	1	1
Diameter	9	19	29
Girth	3	3	3
Density	0.101	0.028	0.013
$\langle k \rangle$	10.040	11.010	11.338
L	3.586	6.917	10.250
C	0.487	0.469	0.464
E_{glob}	0.505	0.316	0.231
E_{loc}	0.832	0.822	0.819

Table C.9: Properties and statistics for L.k12.b graph instances.

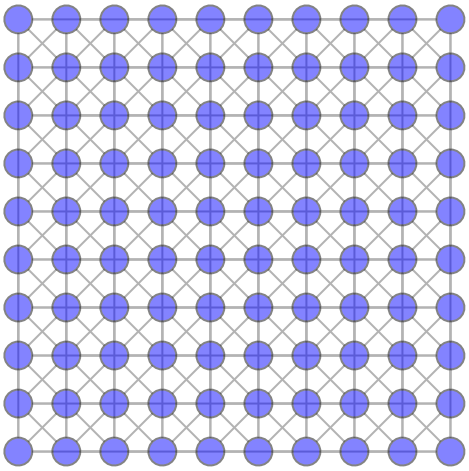


Figure C.27: 2D layout for L.k12 non-circular lattice.

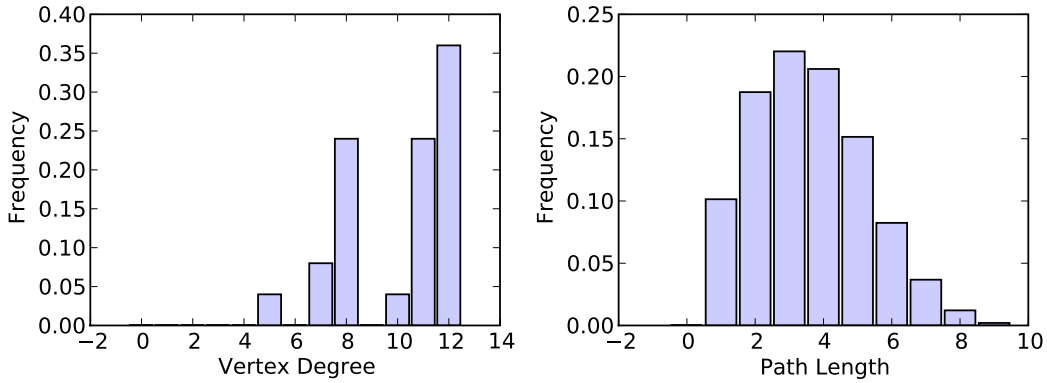


Figure C.28: Vertex degree and path length histograms for L.k12.b, $n = 100$.

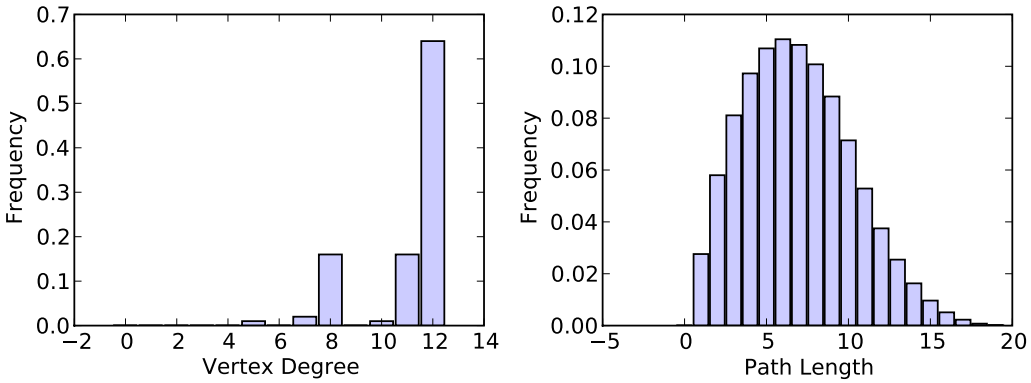


Figure C.29: Vertex degree and path length histograms for L.k12.b, $n = 400$.

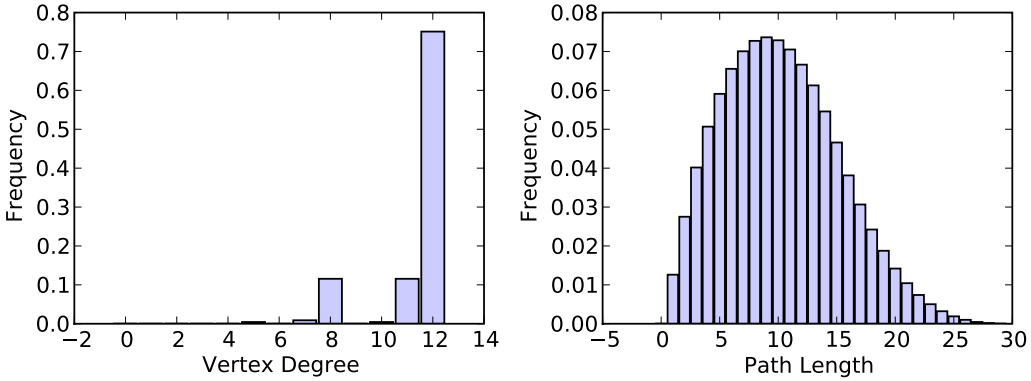


Figure C.30: Vertex degree and path length histograms for L.k12.b, $n = 900$.

C.3.10 Rewired L.k12

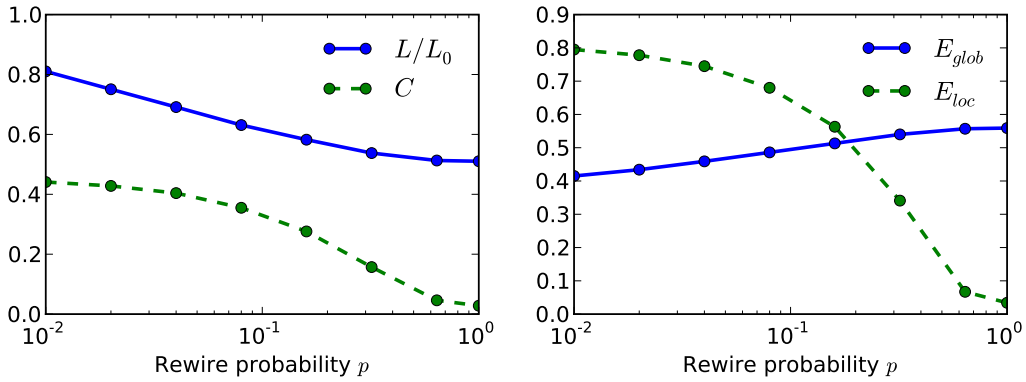
Synopsis: Rewired circular L.k12 lattices of size $n = 400$.

Of all the lattices considered, the k12 configuration presents the highest density. For all the rewiring cases presented here $n = 400$, $m = 2400$, the density is therefore 0.030 and the mean degree $\langle k \rangle = 12$. For each rewiring probability $p > 0$ a sample of 30 instances is used.

In comparison to the rewired lattices of L.k4 and L.k8, the increased density and the greater proportion of overlapping neighbourhoods in a L.k12 structure means that the initial graph has a large number of triangles and a strong level of local efficiency. This can be seen by comparing the local efficiency plots in Figure C.31 with those shown earlier for L.k8 in Figure C.21. Unlike the change in local efficiency, the comparative change in global efficiency is essentially the same as seen in the L.k8 survey.

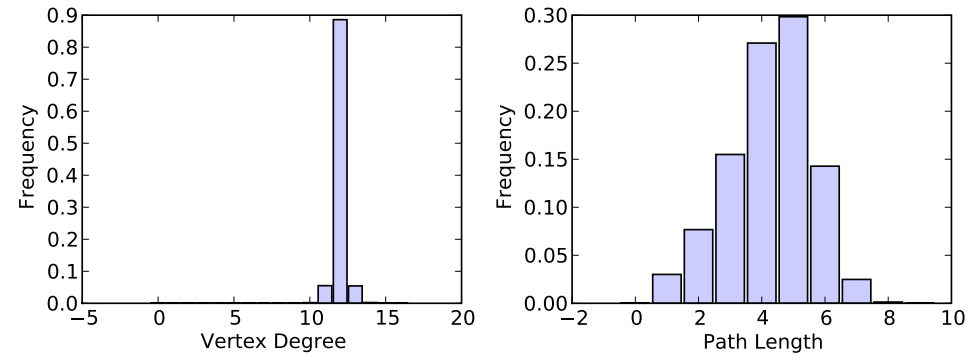
Property	k12.00	k12.01	k12.02	k12.04	k12.08	k12.16	k12.32	k12.64	k12all
p	0.00	0.01	0.02	0.04	0.08	0.16	0.32	0.64	1.00
Components	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Diameter	10.000	7.900	7.000	6.200	5.450	5.000	4.250	4.050	4.050
Girth	3.000	3.000	3.000	3.000	3.000	3.000	3.000	3.000	3.000
L	5.263	4.265	3.951	3.638	3.322	3.065	2.830	2.699	2.686
C	0.455	0.441	0.428	0.404	0.355	0.276	0.157	0.046	0.028
E_{glob}	0.368	0.415	0.434	0.459	0.486	0.513	0.540	0.557	0.559
E_{loc}	0.813	0.795	0.778	0.745	0.680	0.563	0.341	0.067	0.034

Table C.10: Properties and statistics for rewired L.k12 lattice instances.

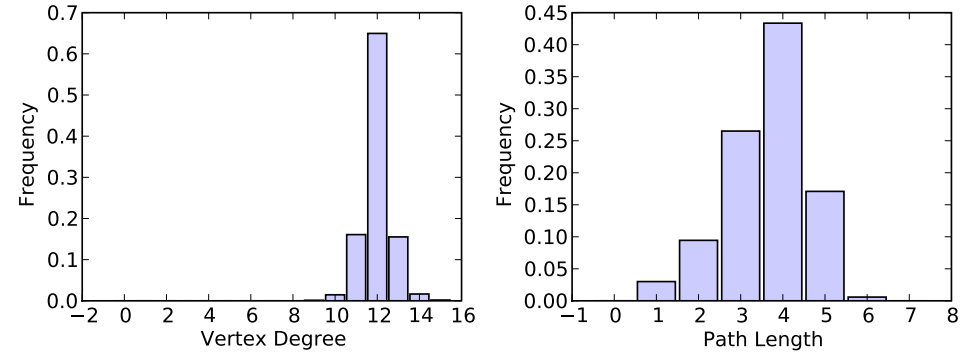


(a) (b)

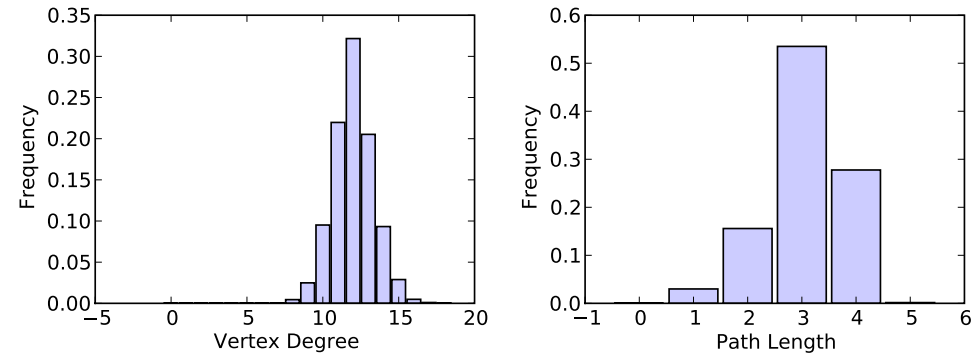
Figure C.31: Influence of p on (a) L/L_0 and C, and on (b) E_{glob} and E_{loc}.



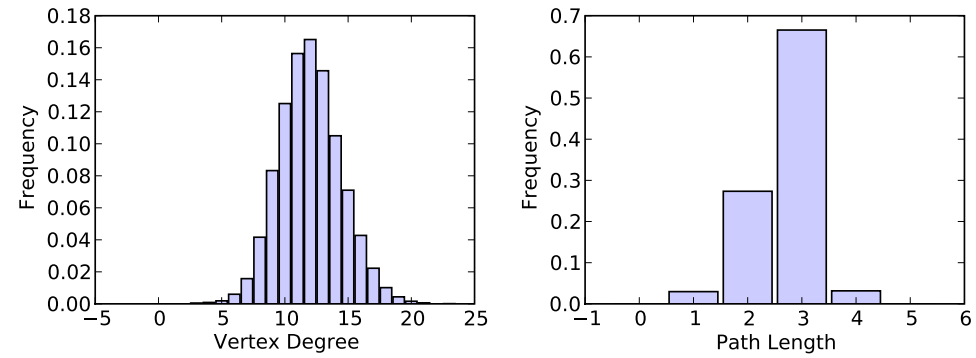
(a) $p = 0.01$



(b) $p = 0.04$



(c) $p = 0.16$



(d) $p = 0.64$

Figure C.32: Vertex degree and path length histograms for rewired L.k12 lattices.

C.3.11 L.hk4

Synopsis: An L.k4 lattice with vertices and edges removed at regular intervals to create hollow regions within the lattice topology.

The L.hk4 lattice is the first of several "hollow" topology profiles. By removing, at regular intervals, vertices and their edges lattices can exhibit a hybrid set of characteristics associated with both dense and sparse lattices.

Property	n100	n400	n900
n	96	408	901
m	142	574	1224
Components	1	1	1
Diameter	10	22	34
Girth	4	4	4
Density	0.031	0.007	0.003
$\langle k \rangle$	2.958	2.814	2.717
L	5.561	11.534	17.281
C	0.000	0.000	0.000
E_{glob}	0.354	0.196	0.138
E_{loc}	0.000	0.000	0.000

Table C.11: Properties and statistics for L.hk4 graph instances.

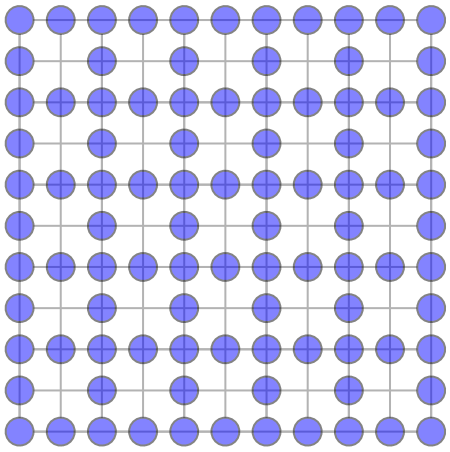


Figure C.33: 2D layout for L.k4 circular lattice with hollows.

In all of the standard 2D lattices presented here, hollows are created at regular non-overlapping intervals. (See Figure C.33.) This may create, as an artefact, exotic vertex degrees from side vertices in non-circular lattices, and unusual patterns from circular lattices. Alternative patterns or random removal schemes could also be easily used but are not considered here.

As vertices are removed to create hollow lattices, the dimensional size is increased slightly to compensate and create hollow lattices that are approximately equal in vertex count to the lattice scales profiled already.

The hollow lattice dimensions used to approximate the three existing lattice scales are 11×11 for $n = 96$, 23×23 for $n = 408$, and 34×35 for $n = 901$.

Consider the initial case of a circular L.hk4 lattice with hollows. The hybrid nature of the topology can be seen in the vertex degree and path length histograms. Although the path length histogram approximately resembles the standard circular L.k4 triangle distribution, the vertex degree histogram includes vertices of degree $k = 2$ and $k = 3$ and the base degree of $k = 4$. The degree $k = 2$ is easily observed in Figure C.33, however the $k = 3$ is less obvious in that they exist as an artefact of the circular connections. By selecting even dimension sizes it is possible to avoid these artefacts and create a graph with only vertices of degree $k = 2$ and $k = 4$.

Although the girth of all circular instances with an odd dimension size is four, it is only for a minority set of side boundary vertices; for the majority of vertices the natural girth is eight, as seen in the non-circular case where the are no circular boundary related artefacts.

In comparison to the circular L.k4 lattice, the hollow L.hk4 lattice diameters and mean path lengths are slightly larger, the densities are – importantly – lower, and global efficiency is only slightly lower. As in the L.k4 lattice, the lack of cycles of length 3 means that there are no clusters, and local efficiency is measured as zero.

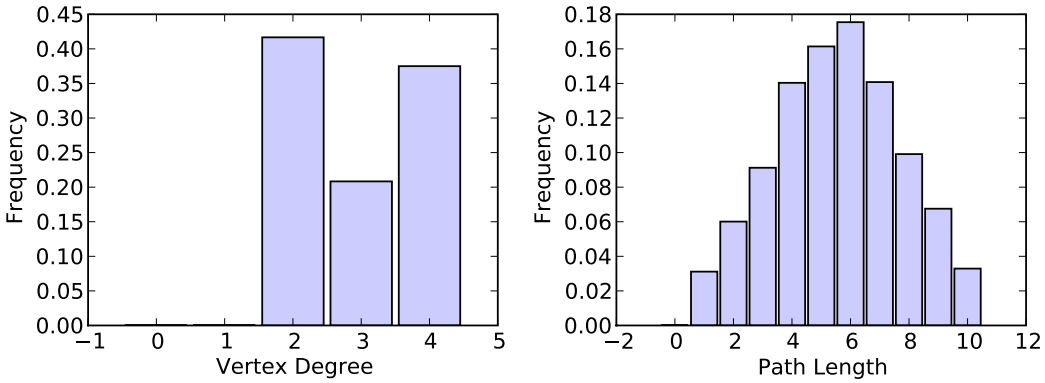


Figure C.34: Vertex degree and path length histograms for L.hk4, $n = 96$.

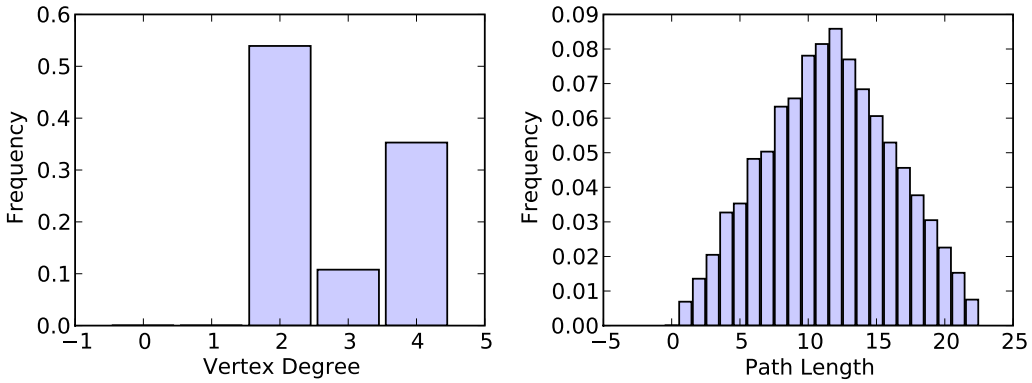


Figure C.35: Vertex degree and path length histograms for L.hk4, $n = 408$.

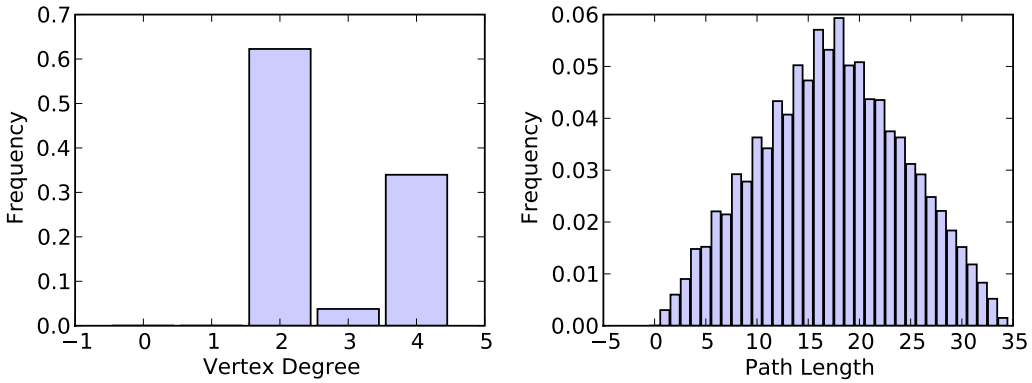


Figure C.36: Vertex degree and path length histograms for L.hk4, $n = 901$.

C.3.12 L.hk4.b

Synopsis: A non-circular (bound) version of the hollow L.k4 lattice.

In comparison to the circular L.hk4 lattice, the diameter of the non-circular L.hk4 lattice doubles. As discussed in the circular case, the natural girth of eight is present, and as a result the cluster and local efficiency measures are zero.

The removal of circular edges reduces the cases of $k = 3$ vertices as seen in the path length histogram, while the influence of two frequent and regular vertex degree types ($k = 2$ and $k = 4$) creates an obvious bifurcation of path length histograms which becomes more obvious as scale increases (and the influence of side vertex degree artefacts is reduced).

Property	n100	n400	n900
n	96	408	901
m	120	528	1172
Components	1	1	1
Diameter	20	44	67
Girth	8	8	8
Density	0.026	0.006	0.003
$\langle k \rangle$	2.500	2.588	2.602
L	7.596	15.578	23.135
C	0.000	0.000	0.000
E_{glob}	0.295	0.164	0.117
E_{loc}	0.000	0.000	0.000

Table C.12: Properties and statistics for L.hk4.b graph instances.

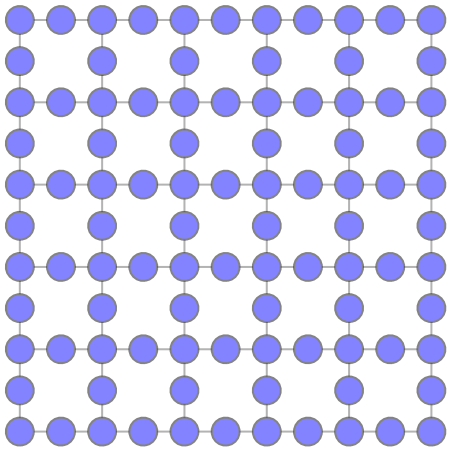


Figure C.37: 2D layout for L.hk4.b non-circular lattice with hollows.

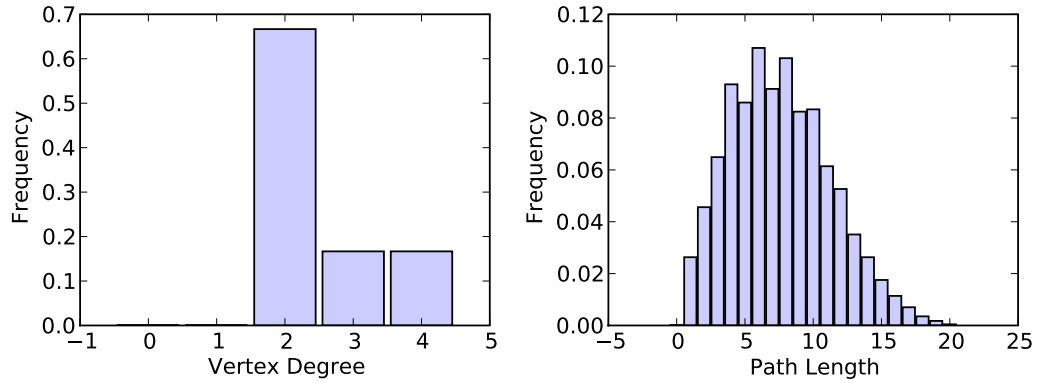


Figure C.38: Vertex degree and path length histograms for L.hk4.b, $n = 96$.

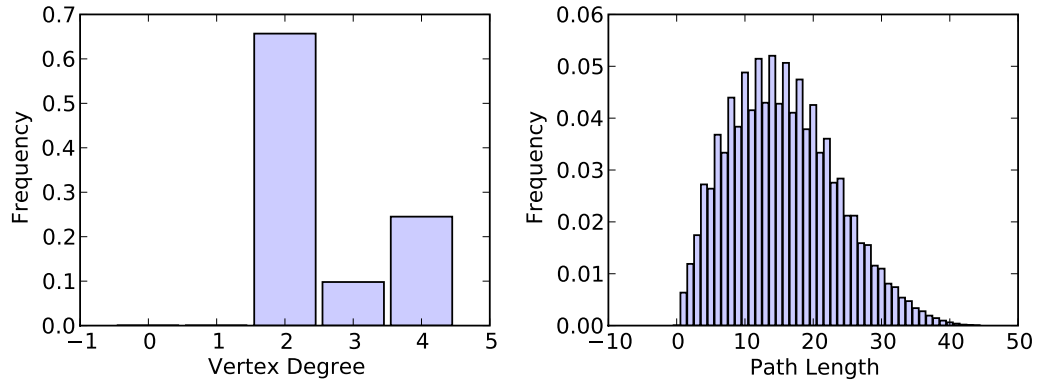


Figure C.39: Vertex degree and path length histograms for L.hk4.b, $n = 408$.

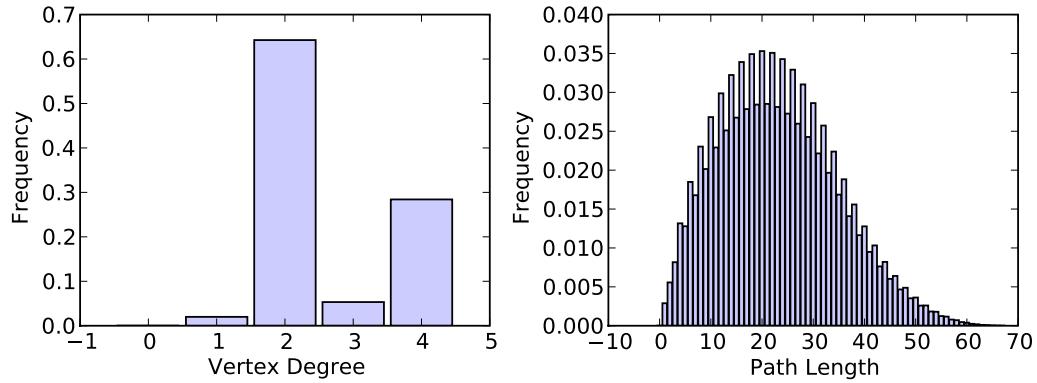


Figure C.40: Vertex degree and path length histograms for L.hk4.b, $n = 901$.

C.3.13 L.hk8

Synopsis: A hollow version of the L.k8 lattice.

The hollow L.hk8 lattice uses the same pattern of vertex removal as the L.hk4 lattice. With the addition of diagonal edges the girth reduces to 3 and the measures of clustering and strong local efficiency are present.

Of the three vertex degree types present (see Figure C.42 for example), the $k = 4$ and the $k = 6$ are the natural forms, with a side artefact of $k = 7$. The half triangle distribution of the path length histogram presents some slight bifurcation most noticeable in the $n = 901$ instance.

Property	n100	n400	n900
n	96	408	901
m	284	1148	2448
Components	1	1	1
Diameter	6	13	18
Girth	3	3	3
Density	0.062	0.014	0.006
$\langle k \rangle$	5.917	5.627	5.434
L	3.724	7.713	11.533
C	0.383	0.362	0.344
E_{glob}	0.467	0.268	0.191
E_{loc}	0.627	0.581	0.550

Table C.13: Properties and statistics for L.hk8 graph instances.

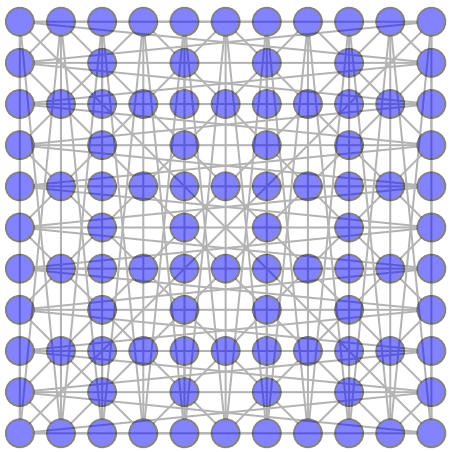


Figure C.41: 2D layout for L.hk8 circular lattice with hollows.

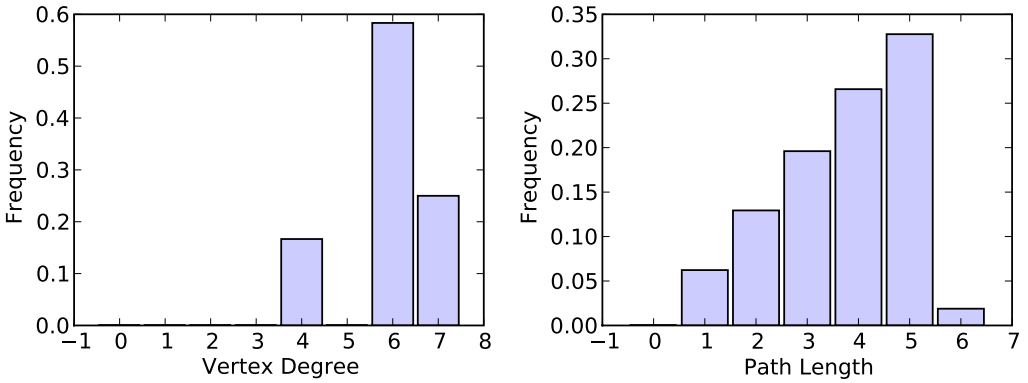


Figure C.42: Vertex degree and path length histograms for L.hk8, $n = 96$.

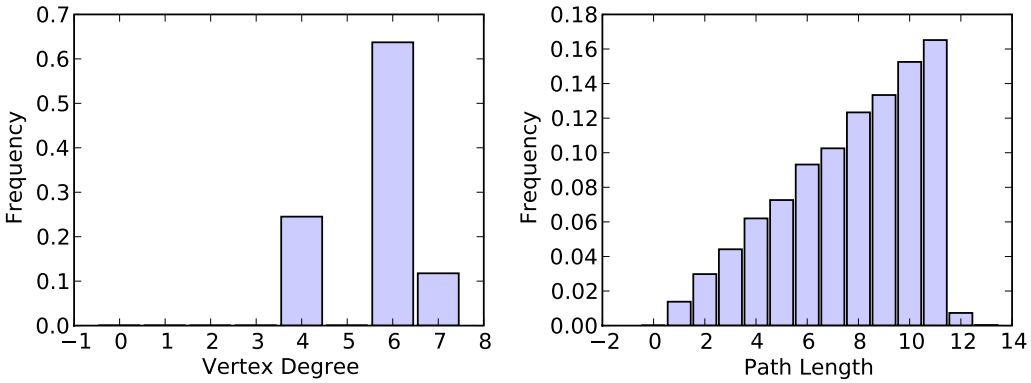


Figure C.43: Vertex degree and path length histograms for L.hk8, $n = 408$.

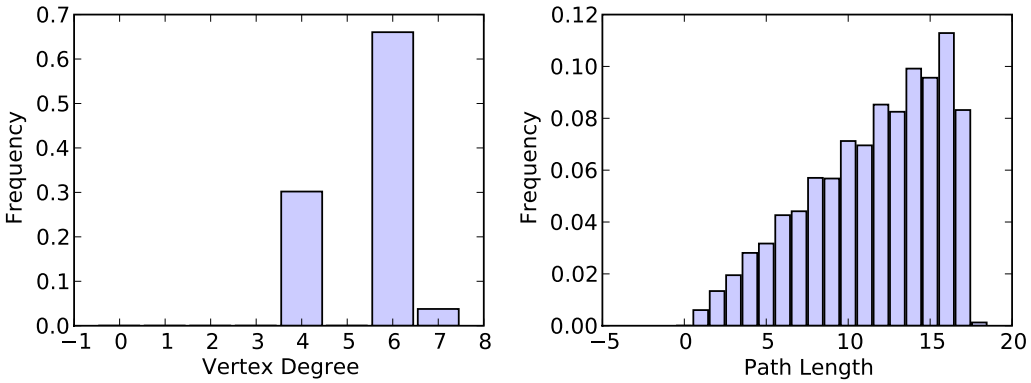


Figure C.44: Vertex degree and path length histograms for L.hk8, $n = 901$.

C.3.14 L.hk8.b

Synopsis: A non-circular (bound) version of the L.hk4 lattice.

Artefact vertices of degrees $k = 7$ are no longer present, but several new artefact species of $k = 2$ and $k = 3$ appear. Despite this, the bifurcation of the slightly skewed parabolic path length histogram is consistent with earlier results and expectations.

In comparison to the circular L.hk8 lattice, the global efficiency decreases in line with the reduced density, increased diameter and increased mean path length. Local efficiency also reduces, although the circular lattice values are elevated by the side artefacts (inconsistent neighbourhood pattern) already discussed.

Property	n100	n400	n900
n	96	408	901
m	220	1012	2294
Components	1	1	1
Diameter	11	23	34
Girth	3	3	3
Density	0.048	0.012	0.006
$\langle k \rangle$	4.583	4.961	5.092
L	5.314	10.901	16.194
C	0.346	0.339	0.338
E_{glob}	0.383	0.220	0.158
E_{loc}	0.556	0.542	0.547

Table C.14: Properties and statistics for L.hk8.b graph instances.

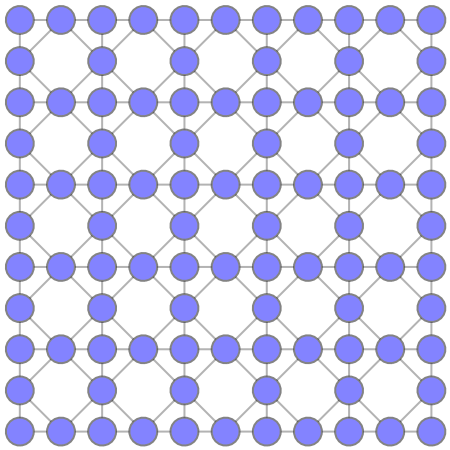


Figure C.45: 2D layout for L.hk8 non-circular lattice with hollows.

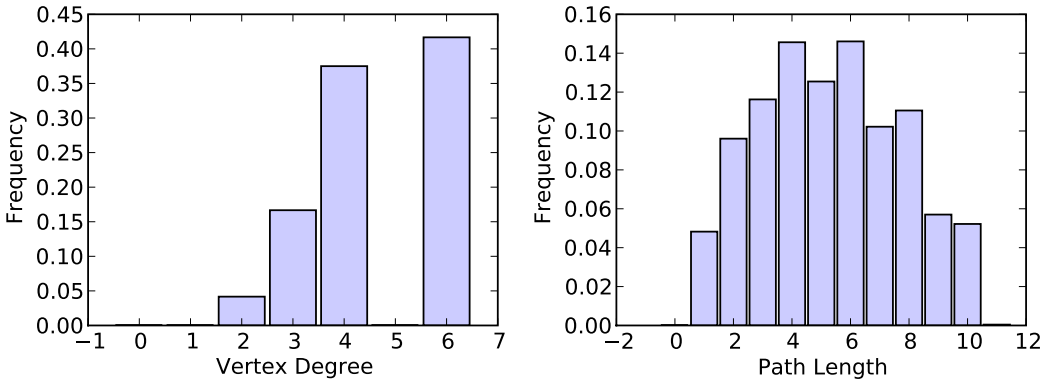


Figure C.46: Vertex degree and path length histograms for L.hk8.b, $n = 96$.

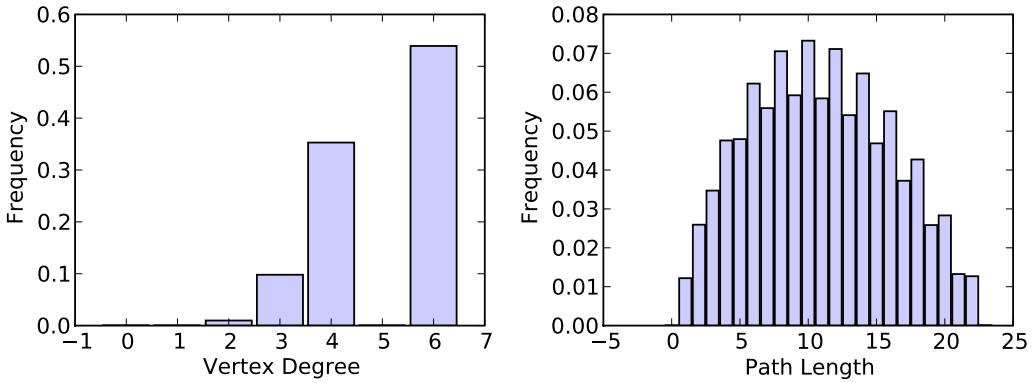


Figure C.47: Vertex degree and path length histograms for L.hk8.b, $n = 408$.

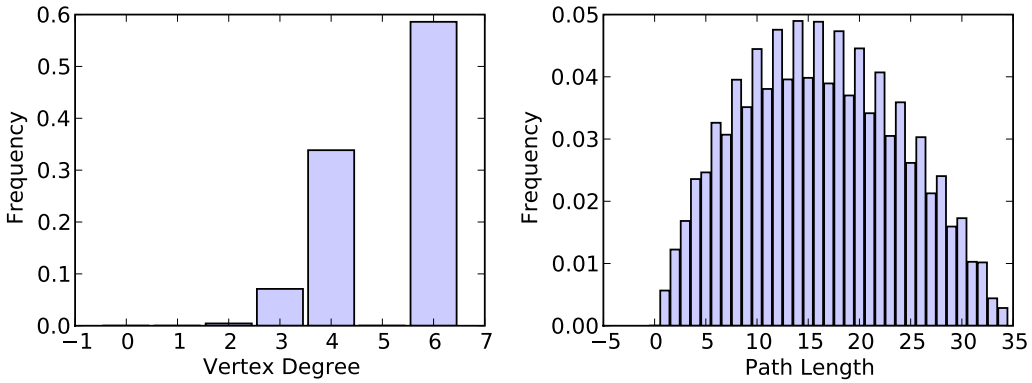


Figure C.48: Vertex degree and path length histograms for L.hk8.b, $n = 901$.

C.3.15 Rewired L.hk8

Synopsis: Rewired and hollow circular L.hk8 lattices of size $n = 408$.

The hollowed-out L.hk8 configuration presents an interesting base before rewiring begins. For all values of $p > 0$ considered 30 samples are taken, the size is $n = 408$ (≈ 400), $m = 1148$, the Density 0.014 and the mean degree $k = 5.627$ a hybrid of two natural forms ($k = 4$ and $k = 6$).

It is conceivable that the presence of heterogeneous neighbourhood structure might be advantageous or even essential to processes that require different local degrees. For example, strong or weak competition based on the number of neighbours. The rewiring may provide the additional small-world style adjustments needed to improve overall performance.

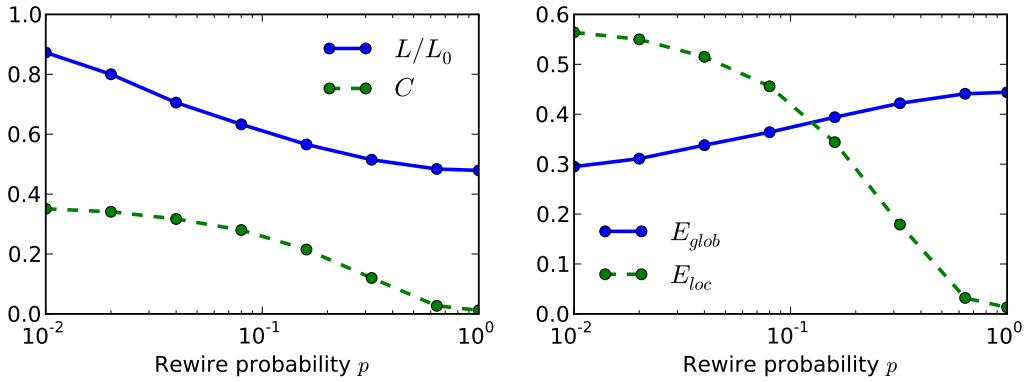
In many respects the L.hk8 is an intermediate between the L.k4 and L.k8 graphs, offering some of the advantages over the simpler and triangle free L.k4, while avoiding some of the cost of the L.k8 (and L.k12) structures.

The initial clustering coefficient is a strong $C = 0.362$, while the mean path length of $L = 7.71$ is only a little higher than that of L.k8 graphs ($L = 6.69$). The influence of rewiring is characteristically similar to that seen in other examples as p is increased; L rapidly reduces and the plateaus increasing global efficiency, while clustering is initially robust but eventually breaks down and local efficiency falls.

What is not clear from this standard review is the influence rewiring has on the vertex species (of degree $k = 4$ and $k = 6$) and how they might diverge. As there is a greater proportion of $k = 8$ vertices it would be expected that they are initially more sensitive the the rewiring process. However, it is the edges that are being rewired rather than vertices being move (or removed) and given the stochastic process involved, the importance of this may be negligible.

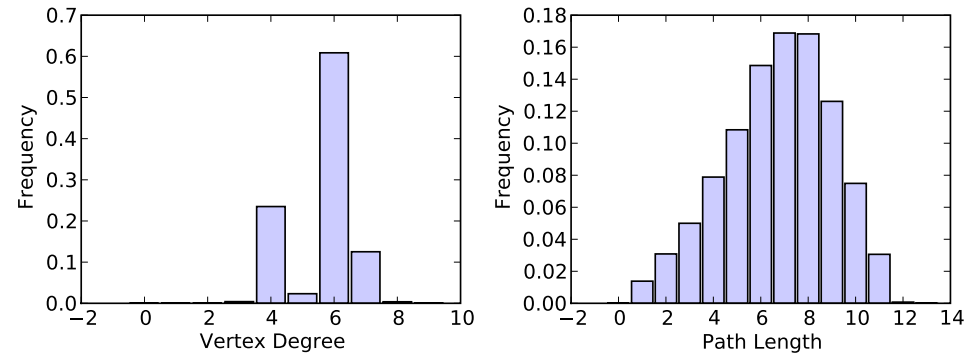
Property	hk8.00	hk8.01	hk8.02	hk8.04	hk8.08	hk8.16	hk8.32	hk8.64	hk8all
p	0.00	0.01	0.02	0.04	0.08	0.16	0.32	0.64	1.00
Components	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Diameter	13.000	12.100	11.250	9.950	8.800	7.450	6.950	6.350	6.600
Girth	3.000	3.000	3.000	3.000	3.000	3.000	3.000	3.000	3.000
L	7.713	6.732	6.171	5.441	4.882	4.364	3.970	3.731	3.696
C	0.362	0.351	0.341	0.317	0.280	0.215	0.120	0.027	0.012
E_{glob}	0.268	0.295	0.311	0.338	0.364	0.394	0.422	0.441	0.444
E_{loc}	0.581	0.564	0.550	0.515	0.456	0.344	0.179	0.032	0.013

Table C.15: Properties and statistics for rewired L.hk8 lattice instances.

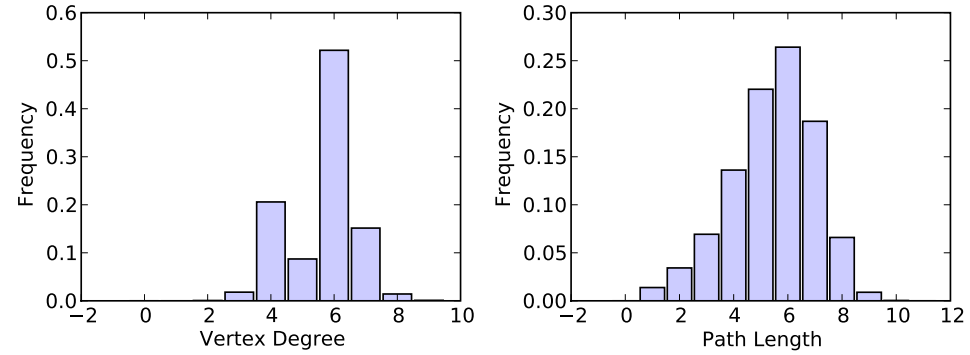


(a) (b)

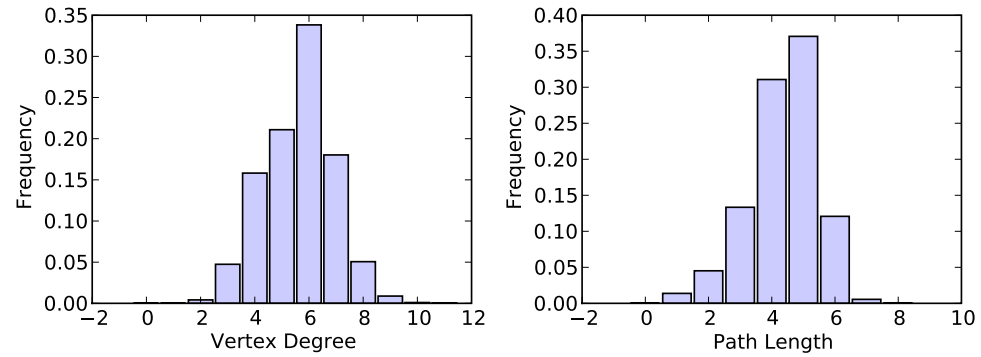
Figure C.49: Influence of p on (a) L/L_0 and C, and on (b) E_{glob} and E_{loc}.



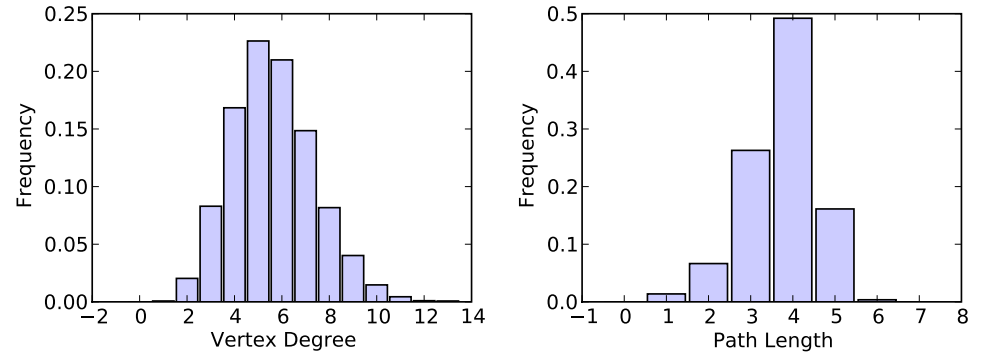
(a) $p = 0.01$



(b) $p = 0.04$



(c) $p = 0.16$



(d) $p = 0.64$

Figure C.50: Vertex degree and path length histograms for rewired L.hk8 lattices.

C.3.16 L.k6

Synopsis: A circular two dimensional (2D) lattice using three axes to create a topology where each vertex has a degree of $k = 6$.

A standard 2D regular lattice is used as a base, and every odd row is offset when considered for neighbourhood connection.

Property	n100	n400	n900
n	100	400	900
m	300	1200	2700
Components	1	1	1
Diameter	7	15	22
Girth	3	3	3
Density	0.061	0.015	0.007
$\langle k \rangle$	6.000	6.000	6.000
L	3.990	7.932	11.885
C	0.400	0.400	0.400
E_{glob}	0.451	0.266	0.189
E_{loc}	0.767	0.767	0.767

Table C.16: Properties and statistics for L.k6 graph instances.

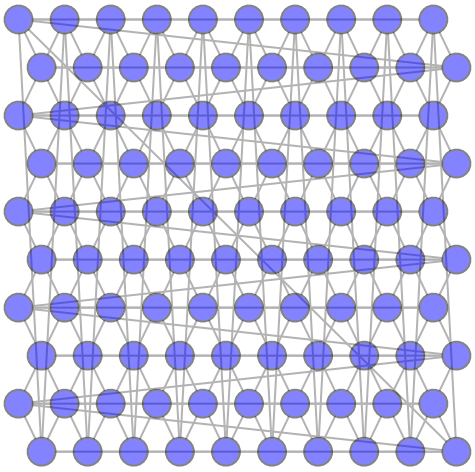


Figure C.51: 2D 3-axes layout for L.k6 circular lattice.

In comparison to the 2D two axes L.hk4 lattice, a L.k6 lattice has a high number of triangles and as a result, strong clustering and local efficiency. Also the diameter and mean path length are smaller and the overall density higher. Unlike the L.k4 lattice, this lattice does not suffer from the constraint of Manhattan routes for diagonal paths. Similarly, the L.k6 lattice is not as dense as the L.k8 lattice, which makes L.k6 a suitable intermediate model between the L.k4 and more dense L.k8 lattice.

The path length histogram for the L.k6 lattice is similar to the L.k4 and L.k8 lattice in that it has a somewhat triangular shape with an additional right skew shift. This is due to a three axes model being constrained by a square 2D connection boundary. (Ideally this boundary space should be aligned to the axes.)

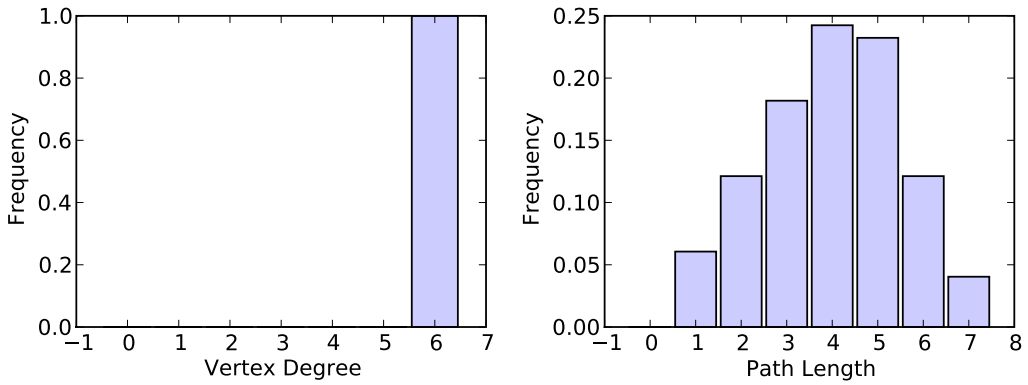


Figure C.52: Vertex degree and path length histograms for L.k6, $n = 100$.

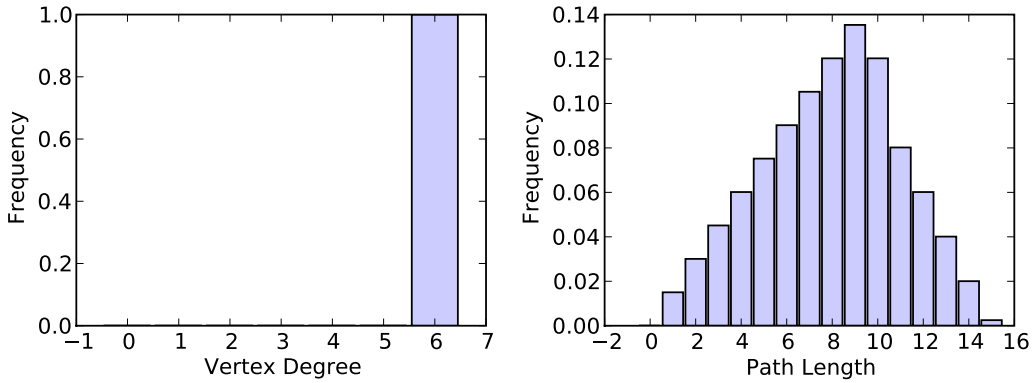


Figure C.53: Vertex degree and path length histograms for L.k6, $n = 400$.

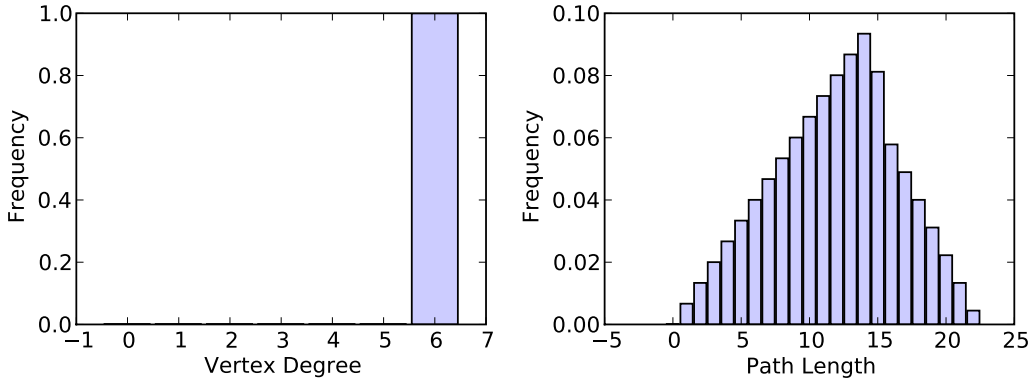


Figure C.54: Vertex degree and path length histograms for L.k6, $n = 900$.

C.3.17 L.k6.b

Synopsis: A non-circular (bound) version of the three axes L.k6 lattice.

This bound L.k6 lattice is similar in characteristics to previous lattice models and their bound/circular comparisons. In this bound form the diameter doubles, mean path length is higher, density lower resulting in an overall lower global efficiency. Locally the clustering and local efficiency all appear higher due to boundary (side) artefacts. All path length histograms take a parabolic distribution shape with a right tail.

Property	n100	n400	n900
n	100	400	900
m	261	1121	2581
Components	1	1	1
Diameter	14	29	44
Girth	3	3	3
Density	0.053	0.014	0.006
$\langle k \rangle$	5.220	5.605	5.736
L	5.391	10.758	16.130
C	0.416	0.407	0.405
E_{glob}	0.383	0.224	0.159
E_{loc}	0.788	0.776	0.773

Table C.17: Properties and statistics for L.k6.b graph instances.

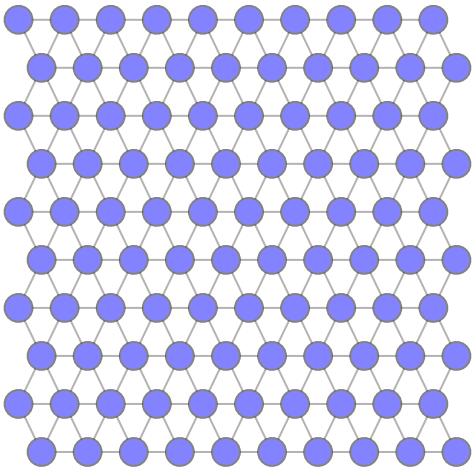


Figure C.55: 2D 3-axes layout for L.k6 non-circular lattice.

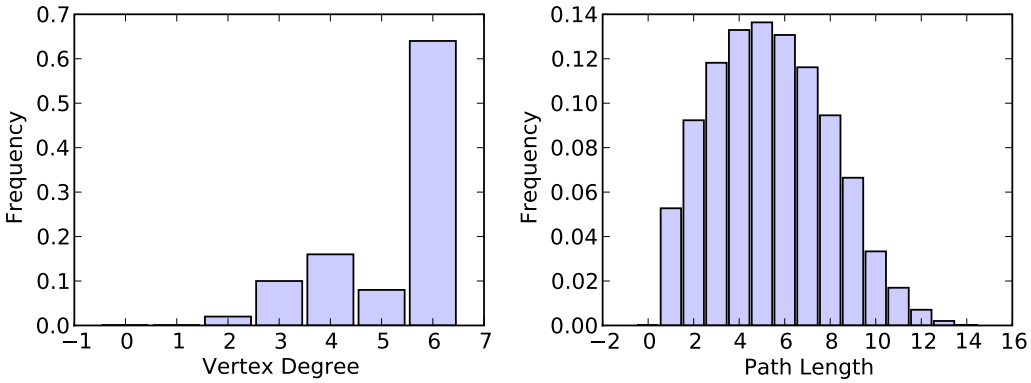


Figure C.56: Vertex degree and path length histograms for L.k6.b, $n = 100$.

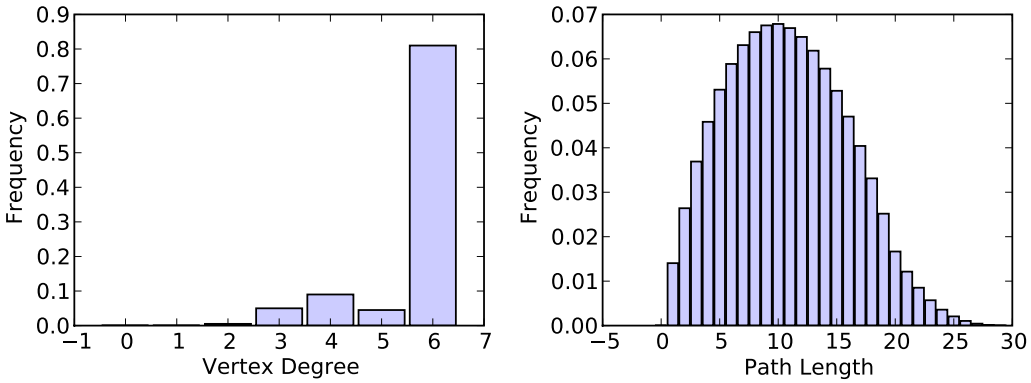


Figure C.57: Vertex degree and path length histograms for L.k6.b, $n = 400$.

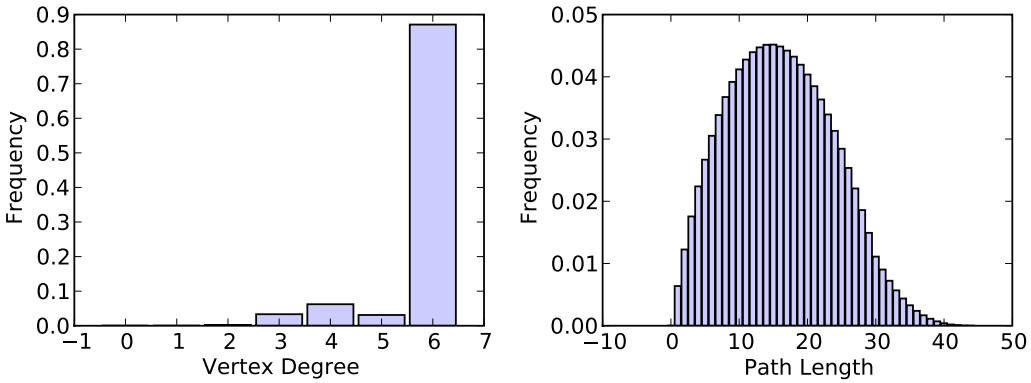


Figure C.58: Vertex degree and path length histograms for L.k6.b, $n = 900$.

C.3.18 Rewired L.k6

Synopsis: Rewired circular L.k6 lattices of size $n = 400$.

Using a planar three axes configuration the L.k6 structure creates a strong base of triangles and highly localised clustering with no overlapping neighbourhoods, while also avoiding the higher density and edge costs of the L.k8 and L.k12 configurations.

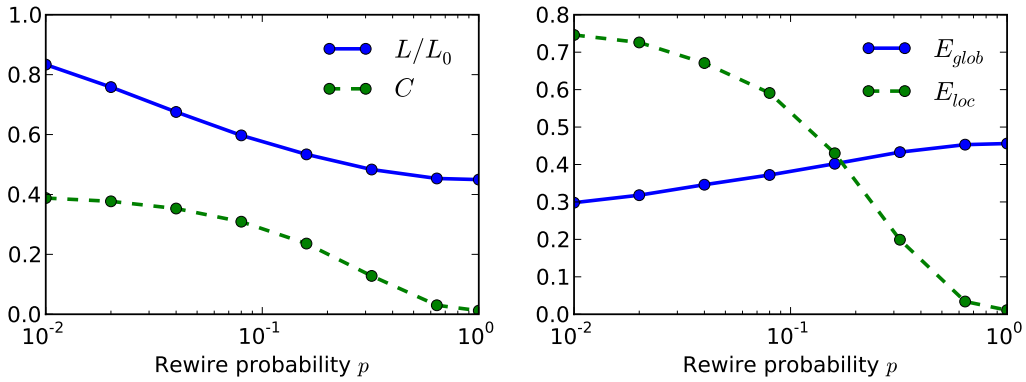
All instances are of size $n = 400$, $m = 1200$, Density 0.015 and by design a mean degree of $\langle k \rangle = 6$. Sample sizes of 30 are used for each value of $p > 0$ considered.

The initial diameter is 15, the mean path length $L = 7.932$, and the local clustering a high $C = 0.4$ (for a local efficiency of $E_{loc} = 0.767$). This is quite good in comparison to the expensive L.k12 graphs.

Again, the influence of increased rewiring levels reduces the mean path length increasing global efficiency, however greater levels of rewiring also eventually breaks down local clustering and reduces local efficiency.

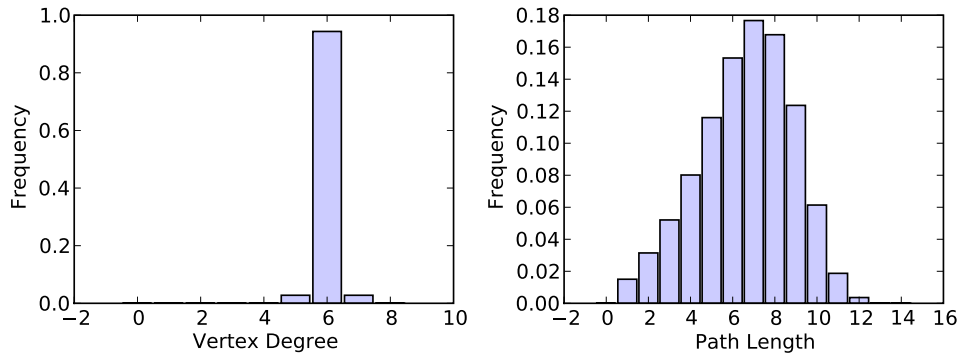
Property	k6.00	k6.01	k6.02	k6.04	k6.08	k6.16	k6.32	k6.64	k6all
p	0.00	0.01	0.02	0.04	0.08	0.16	0.32	0.64	1.00
Components	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.100	1.000
Diameter	15.000	12.750	11.250	9.850	8.450	7.150	6.250	6.100	6.050
Girth	3.000	3.000	3.000	3.000	3.000	3.000	3.000	3.000	3.000
L	7.932	6.612	6.016	5.357	4.737	4.234	3.833	3.596	3.566
C	0.400	0.388	0.377	0.353	0.309	0.236	0.128	0.030	0.012
E_{glob}	0.266	0.298	0.318	0.346	0.372	0.402	0.433	0.453	0.456
E_{loc}	0.767	0.746	0.726	0.671	0.591	0.430	0.199	0.034	0.011

Table C.18: Properties and statistics for rewired L.k6 lattice instances.

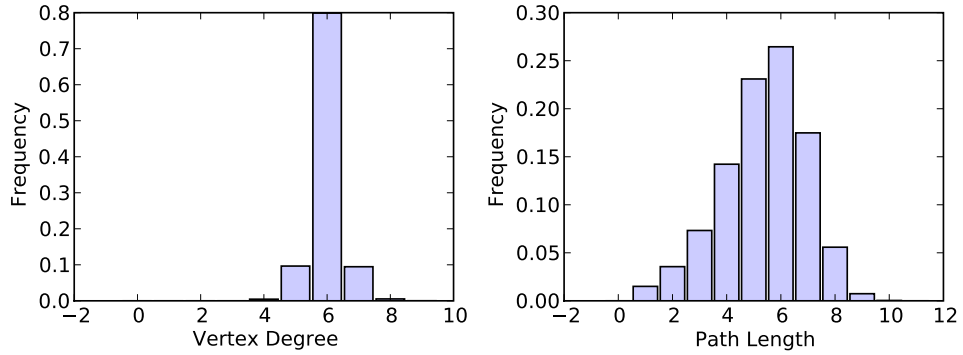


(a) (b)

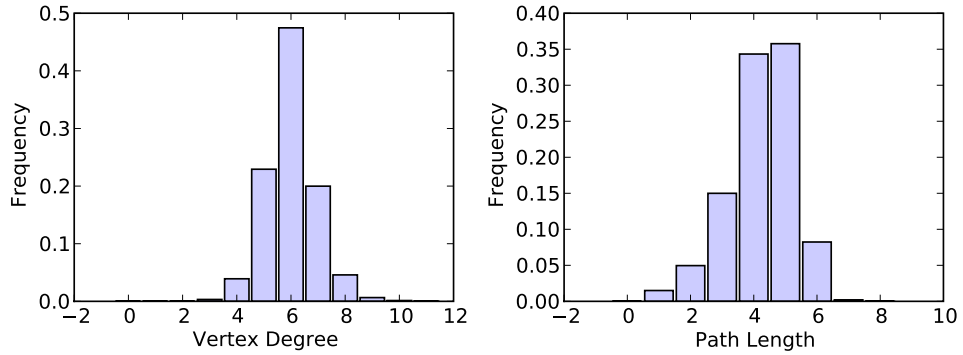
Figure C.59: Influence of p on (a) L/L_0 and C, and on (b) E_{glob} and E_{loc}.



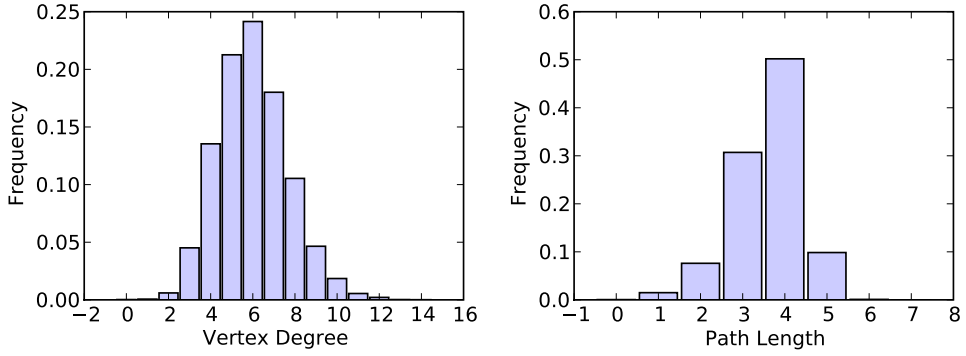
(a) $p = 0.01$



(b) $p = 0.04$



(c) $p = 0.16$



(d) $p = 0.64$

Figure C.60: Vertex degree and path length histograms for rewired L.k6 lattices.

C.3.19 L.hk3

Synopsis: A hollow form of the three axes L.k6 lattice, resulting in a natural "honeycomb" topology where each vertex has a degree of $k = 3$.

The L.k6 lattice lends itself elegantly to a hollow form. When a regular number of "centre" vertices are removed the resultant lattice is a familiar low density "honeycomb" structure. This also results in a rather large girth size of six.

Property	n100	n400	n900
n	96	416	912
m	144	624	1368
Components	1	1	1
Diameter	10	21	31
Girth	6	6	6
Density	0.032	0.007	0.003
$\langle k \rangle$	3.000	3.000	3.000
L	5.474	11.335	16.773
C	0.000	0.000	0.000
E_{glob}	0.356	0.196	0.139
E_{loc}	0.000	0.000	0.000

Table C.19: Properties and statistics for L.hk3 graph instances.

In order to ensure a minimal artefact model (as discussed in the L.hk hollow lattice models) it is best to require that the number of rows be a multiple of two and the columns a multiple of three. This ensures, assuming the lattice coordinates and axes as shown in Figure C.61, that in a circular case there are no artefact vertices of exotic degree. This can be seen in the vertex degree histograms (only degree $k = 3$ is present). Note also the skewed triangular distribution of the path length histogram confirming that it is not due to artefacts but rather the use of a square border and not the natural axes of the lattice.

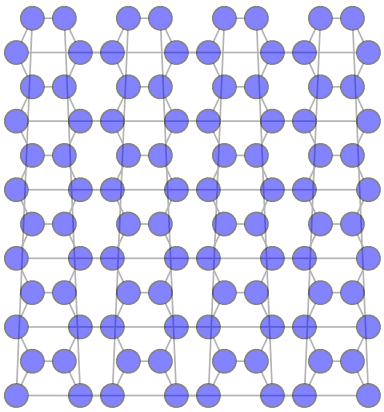


Figure C.61: 2D 3-axes layout for L.hk3 circular lattice with hollows.

With the additional constraints on axes size, the nearest suitable size is selected for each standard scale size. These are 12×12 for $n = 96$, 24×26 for $n = 418$, and 36×38 for $n = 912$.

Interestingly, in terms of density, diameter and global efficiency, this model is almost equivalent to the L.hk4 model, except that in this L.hk3 model the girth is a high 8 and

this model does not display the path length distribution bifurcation that results of a mix of vertex degree species in the L.hk4 model.

The large girth value creates a lot of separation and very low local connectivity properties. Based on triangles only, local efficiency is zero. In a related manner, there is no overlap of vertex neighbourhoods. This isolation, may be a useful feature for specific types of processes that need sparse distribution and degree structure.

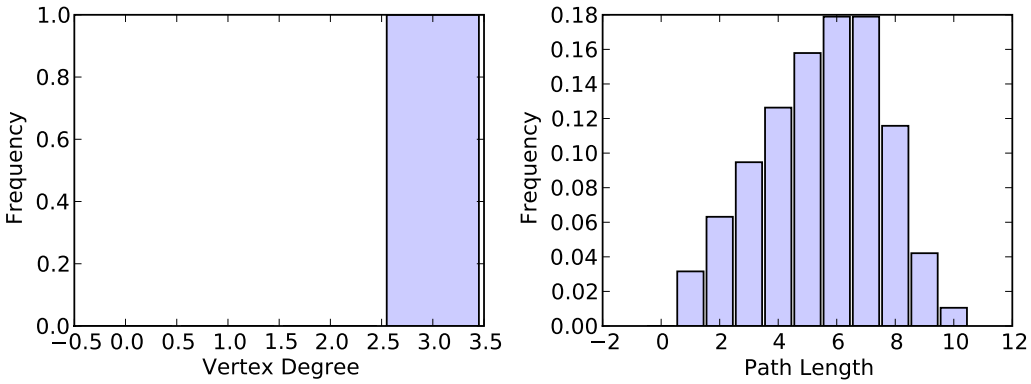


Figure C.62: Vertex degree and path length histograms for L.hk3, $n = 96$.

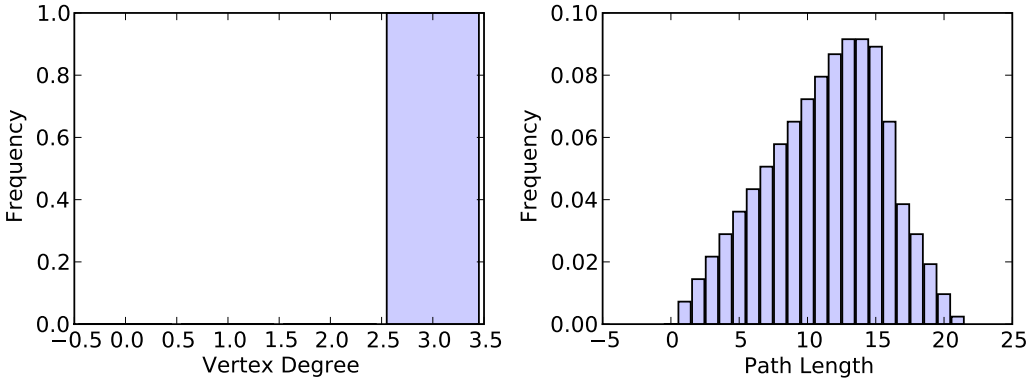


Figure C.63: Vertex degree and path length histograms for L.hk3, $n = 418$.

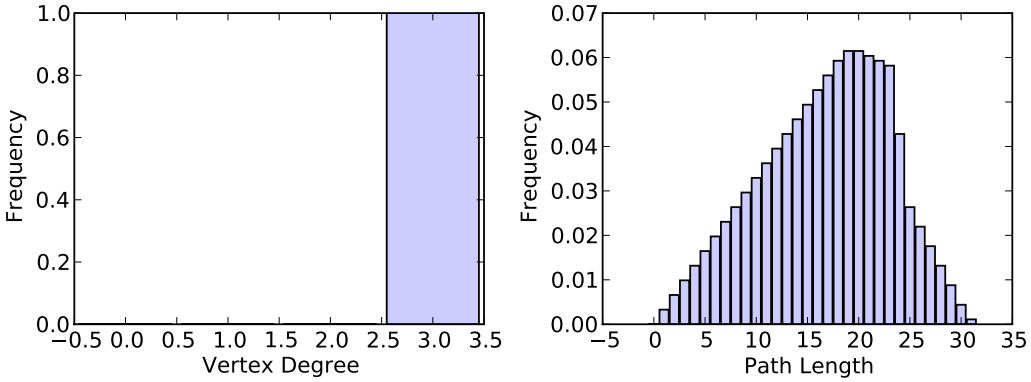


Figure C.64: Vertex degree and path length histograms for L.hk3, $n = 912$.

C.3.20 L.hk3.b

Synopsis: The non-circular (bound) form of the hollow L.hk3 lattice.

As expected from the result of other circular/bound lattice comparisons, in the bound case the diameter increases, mean path length increases, and the overall global efficiency decreases. Local efficiency measures remain zero. The side boundary vertices introduce additional vertex degree types which can be seen in the vertex degree histogram. The path length histogram is parabolic in shape with a small right tail. Note also the presence of "leaf" vertices with single edges ($k = 1$).

Property	n100	n400	n900
n	96	416	912
m	130	595	1325
Components	1	1	1
Diameter	18	40	60
Girth	6	6	6
Density	0.029	0.007	0.003
$\langle k \rangle$	2.708	2.861	2.906
L	7.440	15.440	22.875
C	0.000	0.000	0.000
E_{glob}	0.300	0.165	0.117
E_{loc}	0.000	0.000	0.000

Table C.20: Properties and statistics for L.hk3.b graph instances.

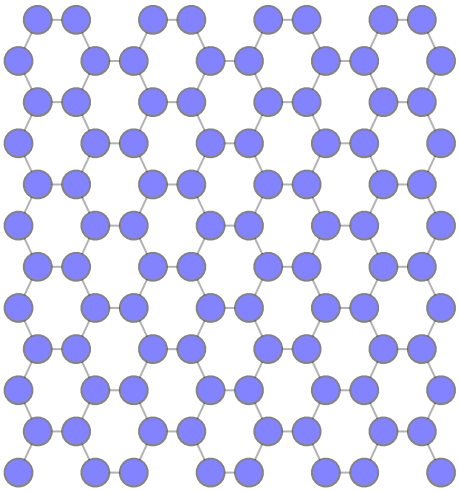


Figure C.65: 2D 3-axes layout for L.hk3 non-circular lattice with hollows.

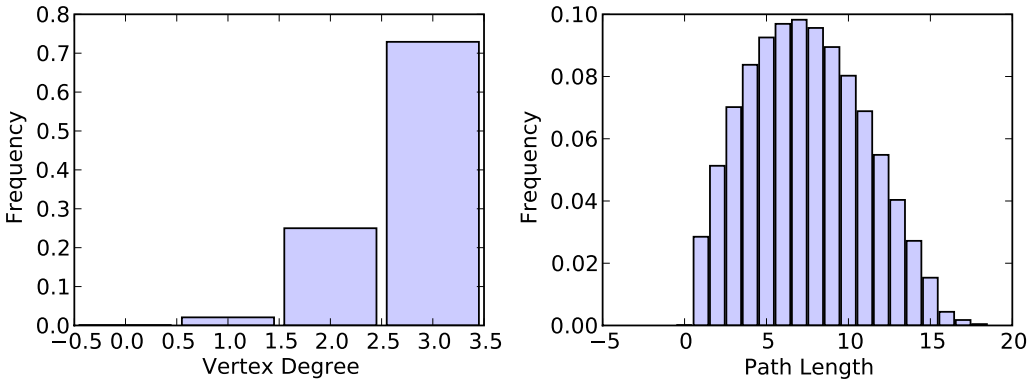


Figure C.66: Vertex degree and path length histograms for L.hk3.b, $n = 96$.

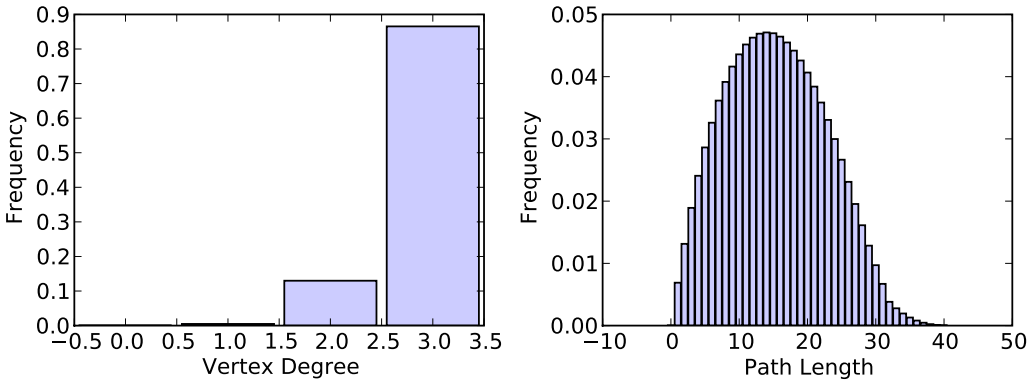


Figure C.67: Vertex degree and path length histograms for L.hk3.b, $n = 418$.

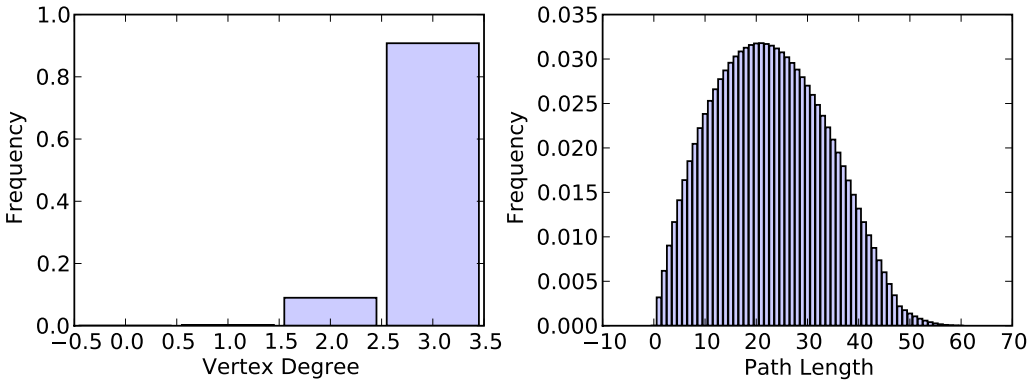


Figure C.68: Vertex degree and path length histograms for L.hk3.b, $n = 912$.

C.3.21 Rewired L.hk3

Synopsis: Rewired circular honeycomb L.hk3 lattices of size $n = 416$.

In this final lattice configuration we alter the classic honeycomb, denoted as L.hk3. For all values of $p > 0$ a sample of 30 graph instances are used. All L.hk3 graphs are of size $n = 416$ (≈ 400), $m = 612$, have a Density of 0.007 and a mean degree of $\langle k \rangle = 3$.

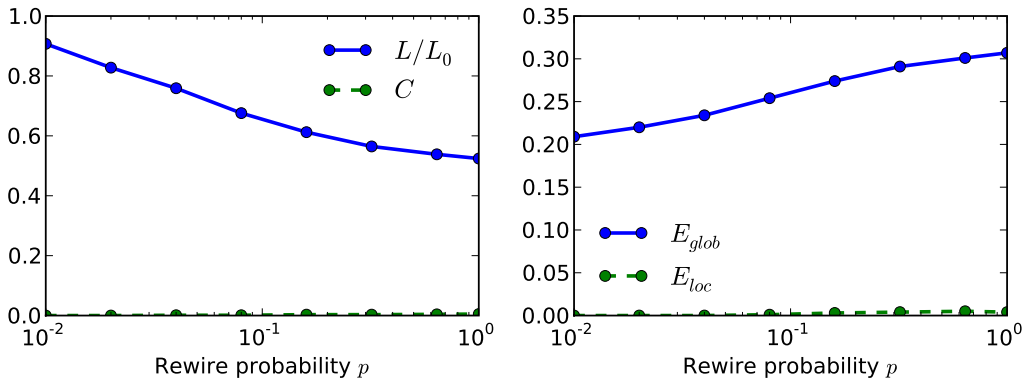
The initial honeycomb lattice is most like the L.k4 configuration, in that the absence of triangles means that clustering and local efficiency are zero. The mean path length is a little larger than a similarly sized L.k4 graph, however the L.hk3 structures has only 3/4 of edges.

With its isolated configuration of vertices, the Girth for the L.hk3 lattice is a large 6. If processes on a network need a degree of isolation, this is a good topology.

Rewiring has the expected result of reducing mean path length and thus improving global efficiency, while essentially having little influence on clustering until rather large degrees of rewiring are applied. It is interesting to note again that in sparse graphs, such as this one, the process of rewiring is more likely to create isolated components.

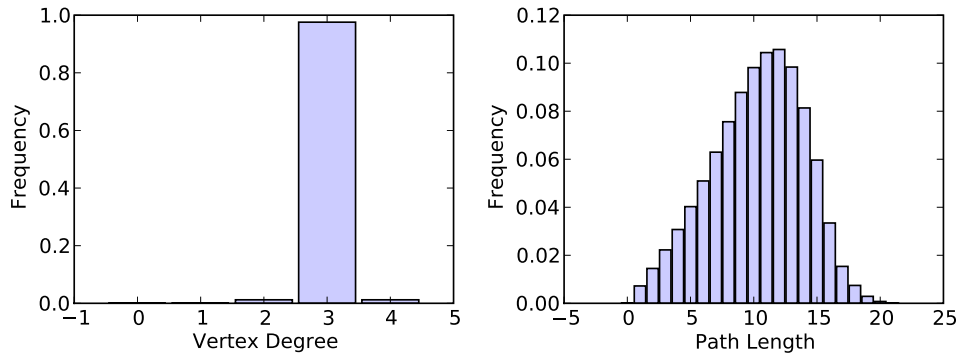
Property	hk3.00	hk3.01	hk3.02	hk3.04	hk3.08	hk3.16	hk3.32	hk3.64	hk3all
p	0.00	0.01	0.02	0.04	0.08	0.16	0.32	0.64	1.00
Components	1.000	1.000	1.000	1.000	1.000	1.000	1.200	1.800	3.100
Diameter	21.000	20.500	18.100	16.350	14.500	13.150	12.700	12.150	12.900
Girth	6.000	5.700	5.050	4.350	4.050	3.300	3.250	3.050	3.050
L	11.335	10.283	9.382	8.603	7.661	6.940	6.400	6.102	5.942
C	0.000	0.000	0.000	0.001	0.001	0.003	0.003	0.004	0.005
E_{glob}	0.196	0.209	0.220	0.234	0.254	0.274	0.291	0.301	0.307
E_{loc}	0.000	0.000	0.000	0.000	0.001	0.003	0.004	0.005	0.004

Table C.21: Properties and statistics for rewired L.hk3 lattice instances.

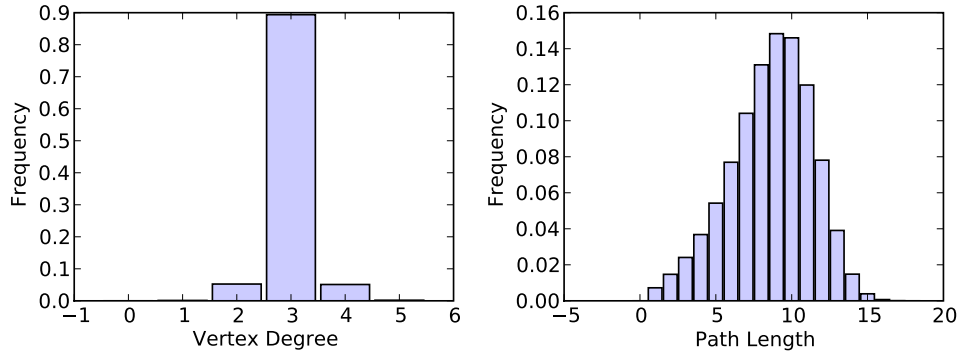


(a) (b)

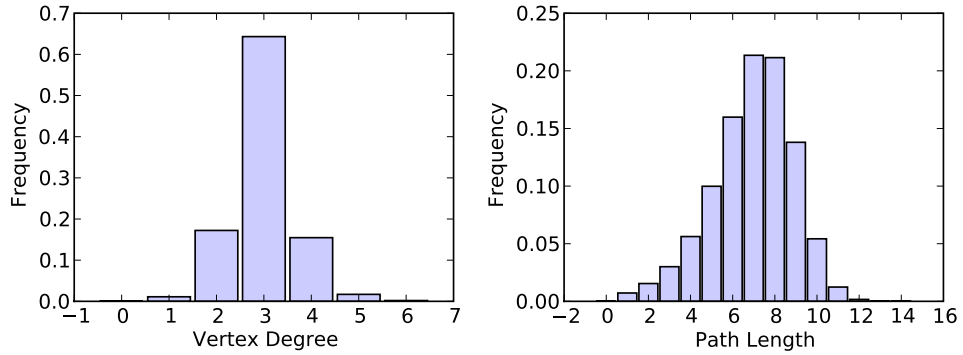
Figure C.69: Influence of p on (a) L/L_0 and C, and on (b) E_{glob} and E_{loc}.



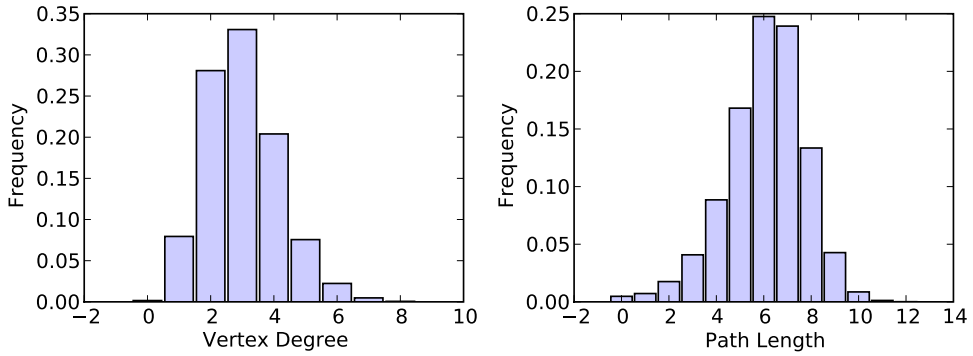
(a) $p = 0.01$



(b) $p = 0.04$



(c) $p = 0.16$



(d) $p = 0.64$

Figure C.70: Vertex degree and path length histograms for rewired L.hk3 lattices.

C.3.22 Regular Lattice Summary

Property	Regular Lattice Type							
	L.k4	L.k6	L.k8	L.k12	L.k4.b	L.k6.b	L.k8.b	L.k12.b
$n = 100$								
m	200	300	400	600	180	261	342	502
Diameter	10	7	5	5	18	14	9	9
Girth	4	3	3	3	4	3	3	3
Density	0.040	0.061	0.081	0.121	0.036	0.053	0.069	0.101
$\langle k \rangle$	4.000	6.000	8.000	12.000	3.600	5.220	6.840	10.040
L	5.051	3.990	3.384	2.778	6.667	5.391	4.680	3.586
C	0.000	0.400	0.429	0.455	0.000	0.416	0.458	0.487
E_{glob}	0.384	0.451	0.503	0.579	0.329	0.383	0.423	0.505
E_{loc}	0.000	0.767	0.798	0.813	0.000	0.788	0.828	0.832
$n = 400$								
m	800	1200	1600	2400	760	1121	1482	2202
Diameter	20	15	10	10	38	29	19	19
Girth	4	3	3	3	4	3	3	3
Density	0.010	0.015	0.020	0.030	0.010	0.014	0.019	0.028
$\langle k \rangle$	4.000	6.000	8.000	12.000	3.800	5.605	7.410	11.010
L	10.025	7.932	6.692	5.263	13.333	10.758	9.340	6.917
C	0.000	0.400	0.429	0.455	0.000	0.407	0.442	0.469
E_{glob}	0.222	0.266	0.301	0.368	0.189	0.224	0.251	0.316
E_{loc}	0.000	0.767	0.798	0.813	0.000	0.776	0.812	0.822
$n = 900$								
m	1800	2700	3600	5400	1740	2581	3422	5102
Diameter	30	22	15	15	58	44	29	29
Girth	4	3	3	3	4	3	3	3
Density	0.004	0.007	0.009	0.013	0.004	0.006	0.008	0.013
$\langle k \rangle$	4.000	6.000	8.000	12.000	3.867	5.736	7.604	11.338
L	15.017	11.885	10.017	7.759	20.000	16.130	14.004	10.250
C	0.000	0.400	0.429	0.455	0.000	0.405	0.437	0.464
E_{glob}	0.157	0.189	0.216	0.271	0.133	0.159	0.179	0.231
E_{loc}	0.000	0.767	0.798	0.813	0.000	0.773	0.807	0.819

Table C.22: Comparison of regular lattice details.

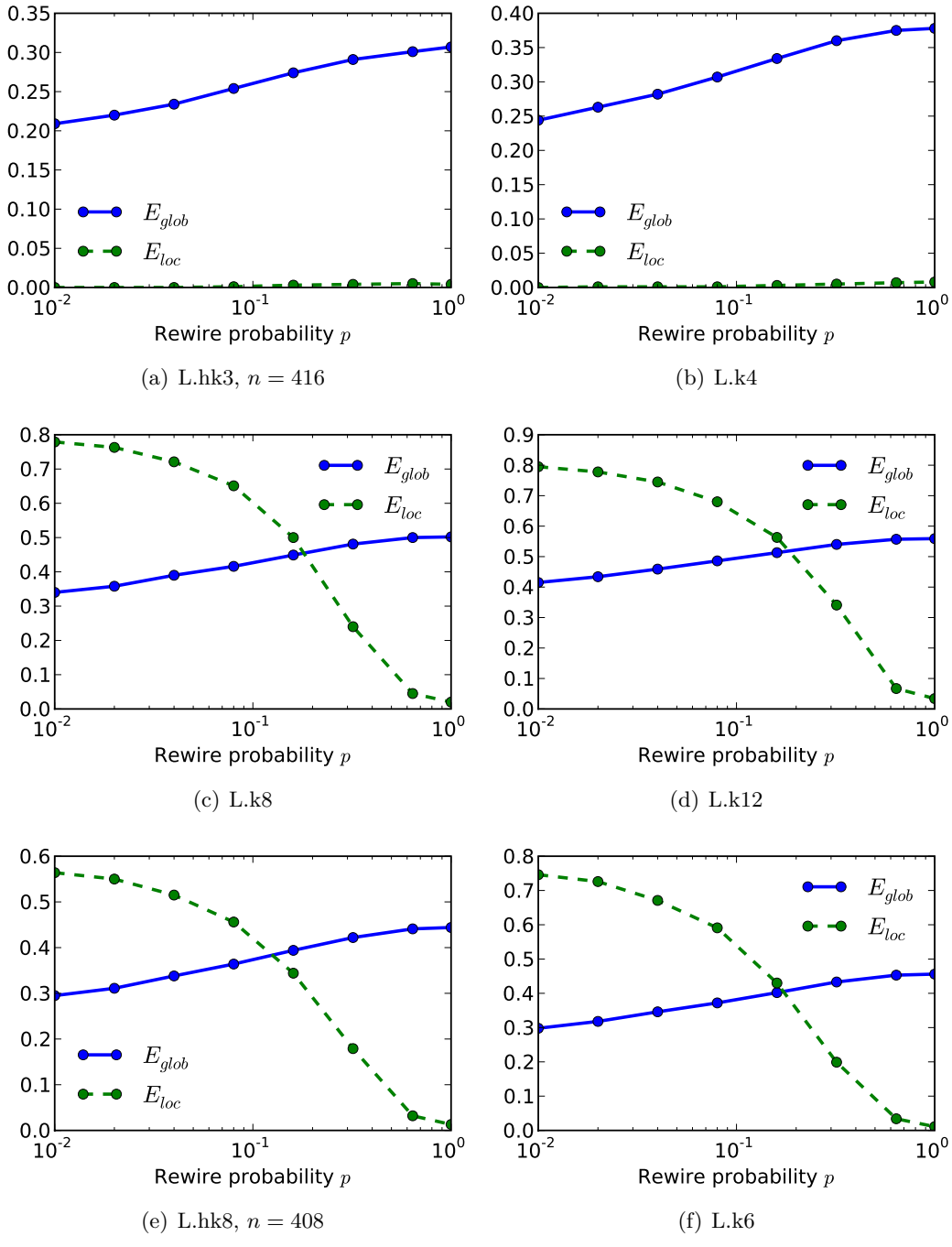
C.3.23 Hollow Lattice Summary

Property	Hollow Lattice Type					
	L.hk4	L.hk3	L.hk8	L.hk4.b	L.hk3.b	L.hk8.b
$n = 100$						
n	96	96	96	96	96	96
m	142	144	284	120	130	220
Diameter	10	10	6	20	18	11
Girth	4	6	3	8	6	3
Density	0.031	0.032	0.062	0.026	0.029	0.048
$\langle k \rangle$	2.958	3.000	5.917	2.500	2.708	4.583
L	5.561	5.474	3.724	7.596	7.440	5.314
C	0.000	0.000	0.383	0.000	0.000	0.346
E_{glob}	0.354	0.356	0.467	0.295	0.300	0.383
E_{loc}	0.000	0.000	0.627	0.000	0.000	0.556
$n = 400$						
n	408	416	408	408	416	408
m	574	624	1148	528	595	1012
Diameter	22	21	13	44	40	23
Girth	4	6	3	8	6	3
Density	0.007	0.007	0.014	0.006	0.007	0.012
$\langle k \rangle$	2.814	3.000	5.627	2.588	2.861	4.961
L	11.534	11.335	7.713	15.578	15.440	10.901
C	0.000	0.000	0.362	0.000	0.000	0.339
E_{glob}	0.196	0.196	0.268	0.164	0.165	0.220
E_{loc}	0.000	0.000	0.581	0.000	0.000	0.542
$n = 900$						
n	901	912	901	901	912	901
m	1224	1368	2448	1172	1325	2294
Diameter	34	31	18	67	60	34
Girth	4	6	3	8	6	3
Density	0.003	0.003	0.006	0.003	0.003	0.006
$\langle k \rangle$	2.717	3.000	5.434	2.602	2.906	5.092
L	17.281	16.773	11.533	23.135	22.875	16.194
C	0.000	0.000	0.344	0.000	0.000	0.338
E_{glob}	0.138	0.139	0.191	0.117	0.117	0.158
E_{loc}	0.000	0.000	0.550	0.000	0.000	0.547

Table C.23: Comparison of hollow lattice details.

C.3.24 Rewired Lattice Summary

Initial lattice density and local clustering configuration are the main influences of the variations observed when comparing the properties of rewired lattices.



(a) L.hk3, $n = 416$

(b) L.k4

(c) L.k8

(d) L.k12

(e) L.hk8, $n = 408$

(f) L.k6

Figure C.71: Comparing the influence of rewiring on E_{glob} and E_{loc}.

If density is initially low, as rewiring is applied it is possible that a graph can become disconnected (the number of components becomes > 1), forming isolated subgraphs. Whether the formation of subgraph is good or bad is application specific. For evolutionary computation, isolation and be an opportunity for specialisation, or a crippling loss of diversity.

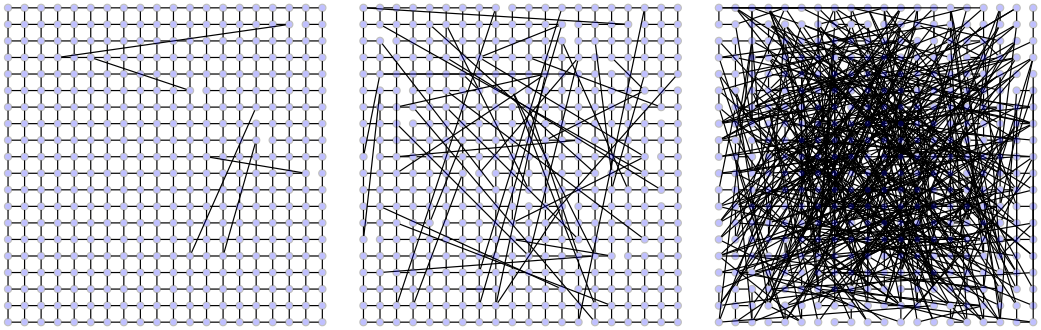
If a lattice initially contains no triangles, the process of rewiring may introduce them, and reduce the girth of a lattice to the minimal possible value of 3.

Table C.24 presents a comparison of details of interest for each of the rewired lattice types considered. As most graphs remain a single large component, unless values are shown the number of components is always 1. Similarly, lattices that originate with triangles will remain at the minimal girth value of 3 throughout the influence of rewiring. If not shown, the girth size is 3.

The main properties of interest that change in response to increased levels of rewiring p are a reduction in mean path length L (and graph diameter), and a decrease in local clustering C. Two similar measure of presenting this change are the global E_{glob} and local E_{loc} efficiency values, which by definition are normalised and unaffected by disconnected graphs. Figure C.71 shows lattice efficiency changes in response to rewiring for each of the lattice types surveyed. For additional details see each respective topology survey.

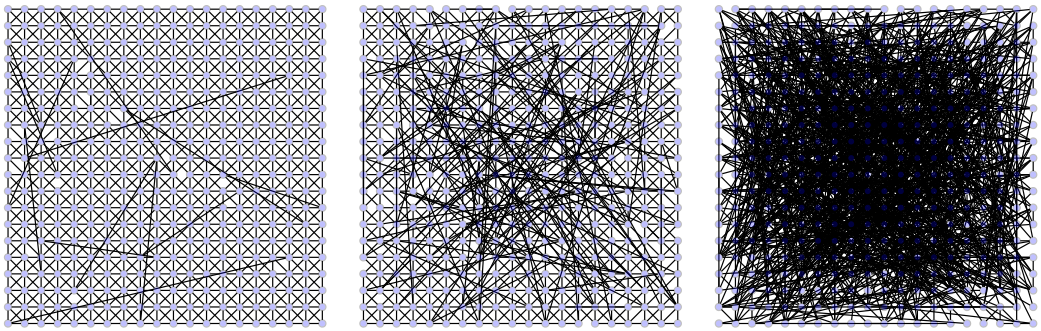
Property	Rewired Lattices								
p	0.00	0.01	0.02	0.04	0.08	0.16	0.32	0.64	1.00
L.hk3, $n = 416$, $m = 624$, density $= 0.007$, $\langle k \rangle = 3$									
Components	1.000	1.000	1.000	1.000	1.000	1.000	1.200	1.800	3.100
Diameter	21.000	20.500	18.100	16.350	14.500	13.150	12.700	12.150	12.900
Girth	6.000	5.700	5.050	4.350	4.050	3.300	3.250	3.050	3.050
L	11.335	10.283	9.382	8.603	7.661	6.940	6.400	6.102	5.942
C	0.000	0.000	0.000	0.001	0.001	0.003	0.003	0.004	0.005
E_{glob}	0.196	0.209	0.220	0.234	0.254	0.274	0.291	0.301	0.307
E_{loc}	0.000	0.000	0.000	0.000	0.001	0.003	0.004	0.005	0.004
L.k4, $n = 400$, $m = 800$, density $= 0.010$, $\langle k \rangle = 4$									
Components	1.000	1.000	1.000	1.000	1.000	1.000	1.050	1.050	1.250
Diameter	20.000	17.400	14.950	12.850	11.150	9.700	8.900	8.250	8.550
Girth	4.000	3.950	3.750	3.600	3.400	3.100	3.000	3.000	3.000
L	10.025	8.577	7.655	6.887	6.071	5.401	4.885	4.630	4.572
C	0.000	0.000	0.001	0.001	0.002	0.003	0.005	0.007	0.007
E_{glob}	0.222	0.244	0.263	0.282	0.307	0.334	0.360	0.375	0.378
E_{loc}	0.000	0.000	0.001	0.001	0.001	0.003	0.005	0.007	0.008
L.hk8, $n = 408$, $m = 1148$, density $= 0.014$, $\langle k \rangle = 5.627$, Girth $= 3$									
Diameter	13.000	12.100	11.250	9.950	8.800	7.450	6.950	6.350	6.600
L	7.713	6.732	6.171	5.441	4.882	4.364	3.970	3.731	3.696
C	0.362	0.351	0.341	0.317	0.280	0.215	0.120	0.027	0.012
E_{glob}	0.268	0.295	0.311	0.338	0.364	0.394	0.422	0.441	0.444
E_{loc}	0.581	0.564	0.550	0.515	0.456	0.344	0.179	0.032	0.013
L.k6, $n = 400$, $m = 1200$, density $= 0.015$, $\langle k \rangle = 6$, Girth $= 3$									
Components	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.100	1.000
Diameter	15.000	12.750	11.250	9.850	8.450	7.150	6.250	6.100	6.050
L	7.932	6.612	6.016	5.357	4.737	4.234	3.833	3.596	3.566
C	0.400	0.388	0.377	0.353	0.309	0.236	0.128	0.030	0.012
E_{glob}	0.266	0.298	0.318	0.346	0.372	0.402	0.433	0.453	0.456
E_{loc}	0.767	0.746	0.726	0.671	0.591	0.430	0.199	0.034	0.011
L.k8, $n = 400$, $m = 1600$, density $= 0.020$, $\langle k \rangle = 8$, Girth $= 3$									
Diameter	10.000	10.000	9.300	8.000	7.050	6.000	5.200	5.000	5.050
L	6.692	5.570	5.060	4.540	4.086	3.657	3.337	3.149	3.127
C	0.429	0.415	0.403	0.378	0.335	0.254	0.140	0.035	0.018
E_{glob}	0.301	0.340	0.358	0.390	0.416	0.449	0.481	0.500	0.502
E_{loc}	0.798	0.779	0.763	0.721	0.651	0.500	0.240	0.045	0.020
L.k12, $n = 400$, $m = 2400$, density $= 0.030$, $\langle k \rangle = 12$, Girth $= 3$									
Diameter	10.000	7.900	7.000	6.200	5.450	5.000	4.250	4.050	4.050
L	5.263	4.265	3.951	3.638	3.322	3.065	2.830	2.699	2.686
C	0.455	0.441	0.428	0.404	0.355	0.276	0.157	0.046	0.028
E_{glob}	0.368	0.415	0.434	0.459	0.486	0.513	0.540	0.557	0.559
E_{loc}	0.813	0.795	0.778	0.745	0.680	0.563	0.341	0.067	0.034

Table C.24: Comparison of rewired lattice details.



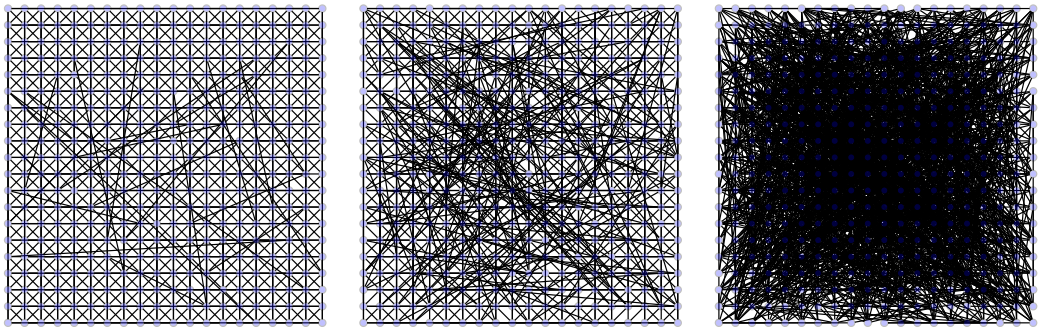
(a) $p = 0.01$ (b) $p = 0.08$ (c) $p = 0.64$

Figure C.72: Sample layout instances of rewired L.k4 lattices.



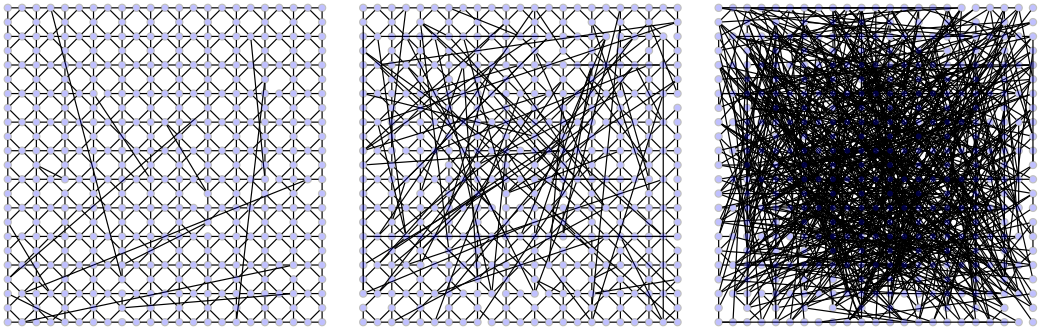
(a) $p = 0.01$ (b) $p = 0.08$ (c) $p = 0.64$

Figure C.73: Sample layout instances of rewired L.k8 lattices.



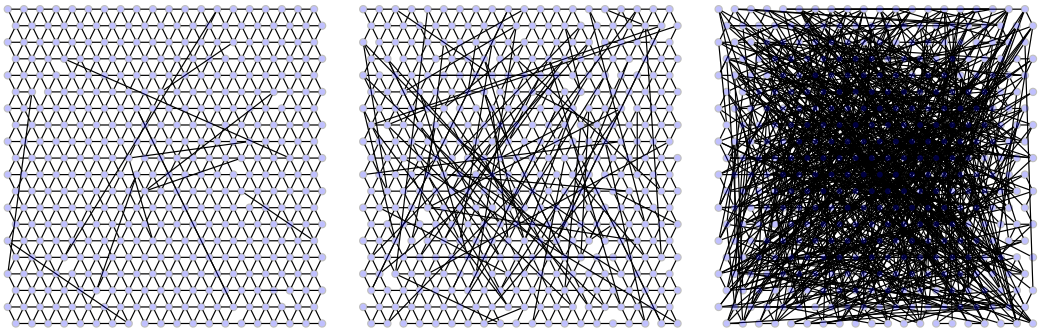
(a) $p = 0.01$ (b) $p = 0.08$ (c) $p = 0.64$

Figure C.74: Sample layout instances of rewired L.k12 lattices.



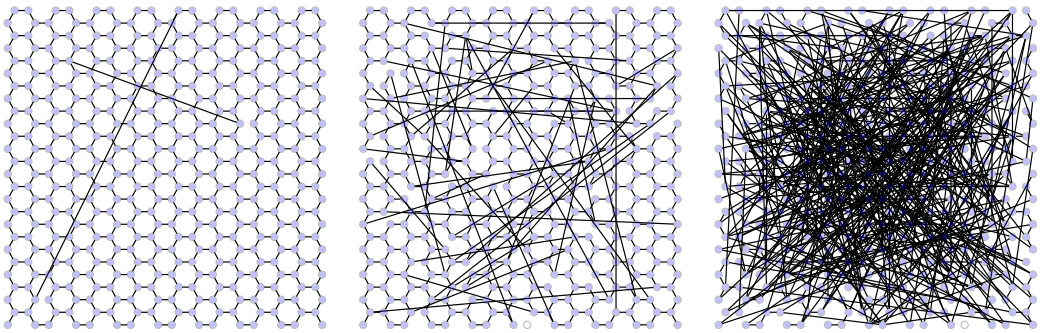
(a) $p = 0.01$ (b) $p = 0.08$ (c) $p = 0.64$

Figure C.75: Sample layout instances of rewired L.hk8 lattices.



(a) $p = 0.01$ (b) $p = 0.08$ (c) $p = 0.64$

Figure C.76: Sample layout instances of rewired L.k6 lattices.



(a) $p = 0.01$ (b) $p = 0.08$ (c) $p = 0.64$

Figure C.77: Sample layout instances of rewired L.hk3 lattices.

C.4 Star

Synopsis: A simple graph topology of a single central root vertex with all other vertices connected as leaves.

Star topology creates a single component graph of low cost with a highly critical central hub vertex. This results in a low mean path length (for which $\lim_{n\to\infty} L = 2$), but also results in no cycles (Girth is zero) or clustering ($C = E_{loc} = 0$).

Only the histograms for the $n = 100$ case are shown because of the simple analytically expected result. For any value of n the majority of $n - 1$ leaf vertices each have a single $k = 1$ connection to the centre hub vertex, which has a degree of $k = n - 1$.

Property	n100	n400	n900
n	100	400	900
m	99	399	899
Diameter	2	2	2
Density	0.020	0.005	0.002
$\langle k \rangle$	1.980	1.995	1.998
L	1.980	1.995	1.998
E_{glob}	0.673	0.668	0.667

Table C.25: Properties and statistics for star graph instances.

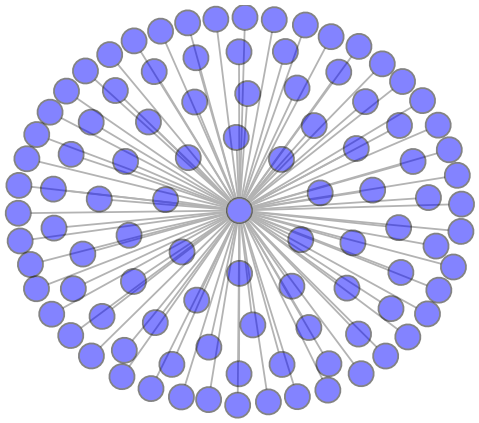


Figure C.78: Force-based layout for star graph of $n = 100$ size.

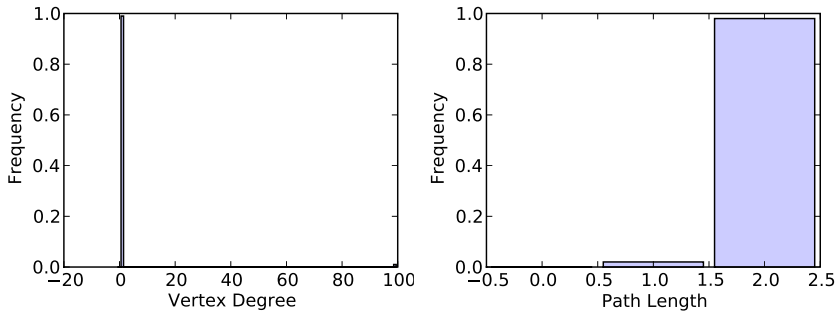


Figure C.79: Vertex degree and path length histograms for a star, $n = 100$.

C.5 Tree (T)

C.5.1 T.c2

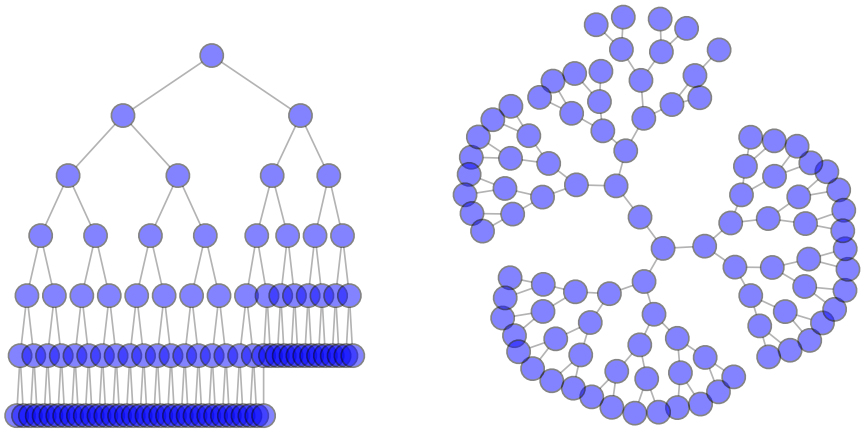
Synopsis: A rooted tree growth model that adds $c = 2$ new children to leaf vertices each growth step until the specified number of vertices n is obtained.

If an appropriate number of vertices n is selected for the value of c then the tree will be balanced. There will always be $n - 1$ edges.

The vertex degree histogram shows the two dominant species; parent vertices of $k = c + 1$ edges, and leaf vertices with only one edge to a parent. The single root vertex has a degree of exactly c. If the tree is not balanced then other vertex degree types will exist. Path length histogram distribution is exponential for a balanced tree, and is a close approximation for unbalanced trees.

Property	n100	n400	n900
n	100	400	900
m	99	399	899
Components	1	1	1
Diameter	12	16	18
Girth	0	0	0
Density	0.020	0.005	0.002
$\langle k \rangle$	1.980	1.995	1.998
L	7.731	11.427	13.726
C	0.000	0.000	0.000
E_{glob}	0.269	0.181	0.149
E_{loc}	0.000	0.000	0.000

Table C.26: Properties and statistics for T.c2 graph instances.



(a) (b)

Figure C.80: (a) Tree and (b) force-based layout for T.c2 graph.

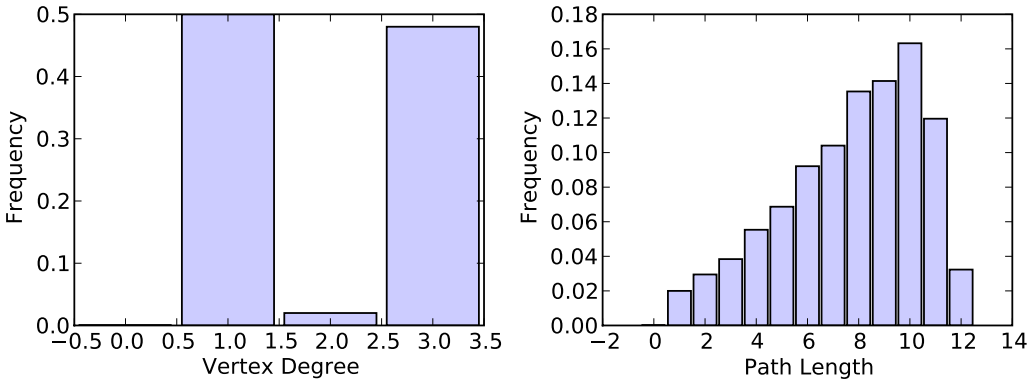


Figure C.81: Vertex degree and path length histograms for T.c2, $n = 100$.

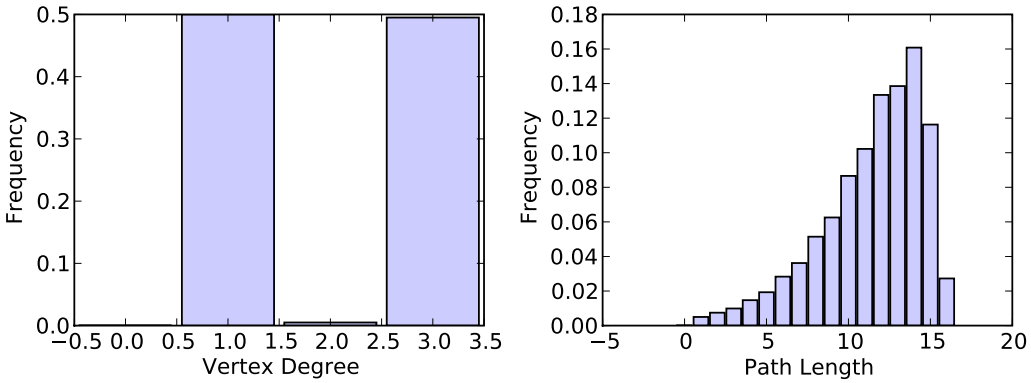


Figure C.82: Vertex degree and path length histograms for T.c2, $n = 400$.

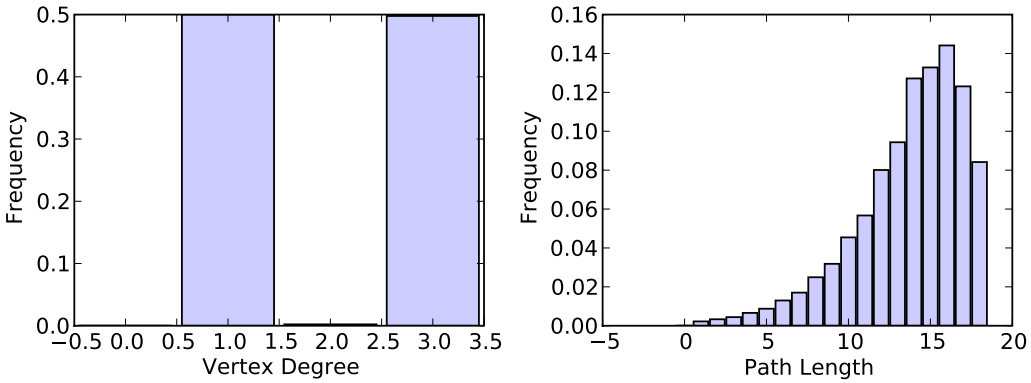


Figure C.83: Vertex degree and path length histograms for T.c2, $n = 900$.

C.5.2 T.c3

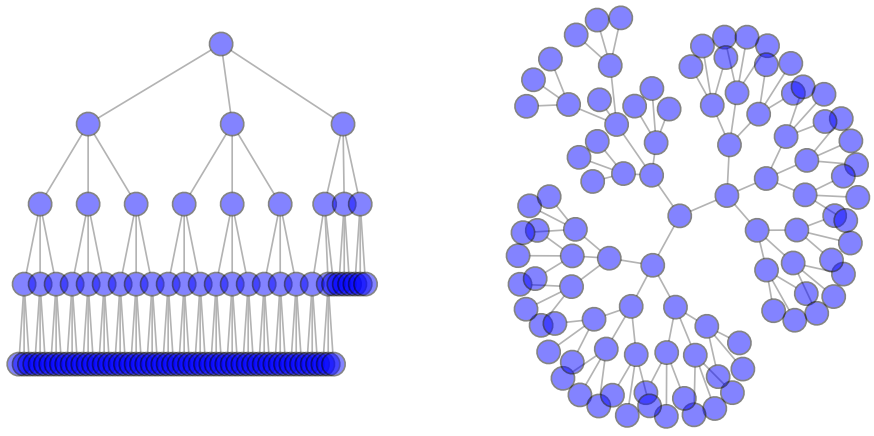
Synopsis: A rooted tree growth model with $c = 3$ children added each growth step until the required number of n vertices is reached.

Interestingly, the mean degree $\langle k \rangle$ remains relatively low despite the increased number of children added to parent vertices. Insight into this can be observed in the degree distribution; the increase in vertex degree of inner parent nodes ($k = 3$) is offset by the increased proportion of leaf vertices ($k = 1$). As a result, density also remains the same with respect to graph size n.

The change in mean path length L is a significant characteristic of Trees with larger values for C. With fewer edges between the root and leaf vertices, all vertices are closer to each other.

Property	n100	n400	n900
n	100	400	900
m	99	399	899
Components	1	1	1
Diameter	8	11	12
Girth	0	0	0
Density	0.020	0.005	0.002
$\langle k \rangle$	1.980	1.995	1.998
L	5.880	8.291	9.717
C	0.000	0.000	0.000
E_{glob}	0.324	0.233	0.198
E_{loc}	0.000	0.000	0.000

Table C.27: Properties and statistics for T.c3 graph instances.



(a) (b)

Figure C.84: (a) Tree and (b) force-based layout for T.c3 graph.

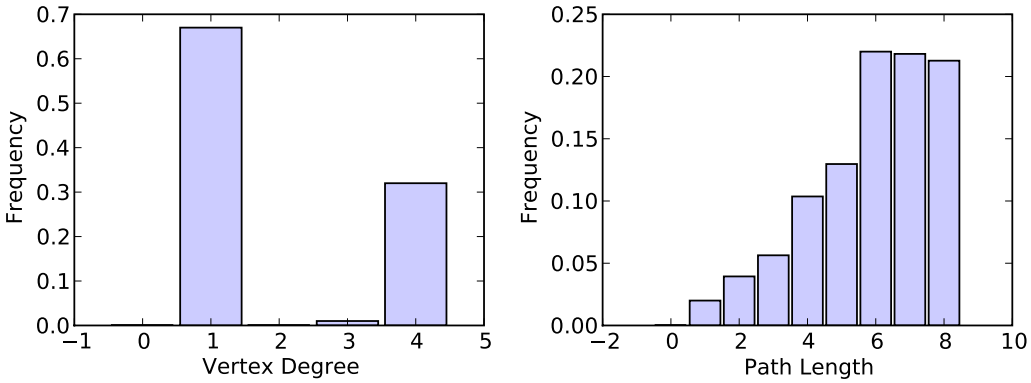


Figure C.85: Vertex degree and path length histograms for T.c3, $n = 100$.

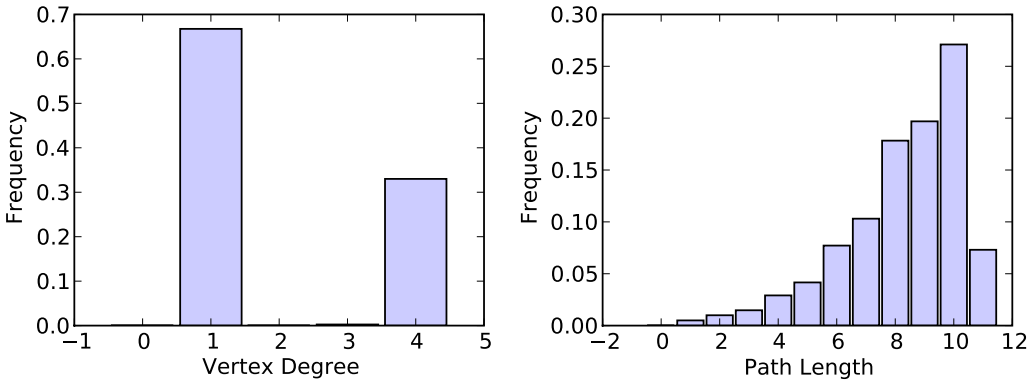


Figure C.86: Vertex degree and path length histograms for T.c3, $n = 400$.

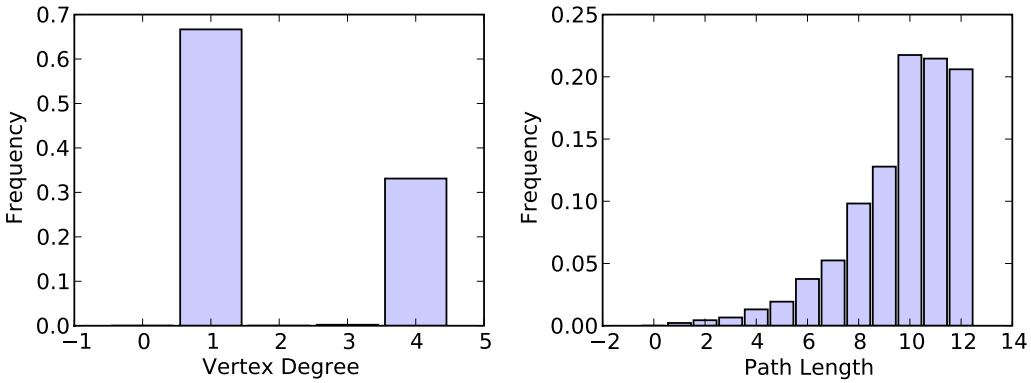


Figure C.87: Vertex degree and path length histograms for T.c3, $n = 900$.

C.5.3 T.c4

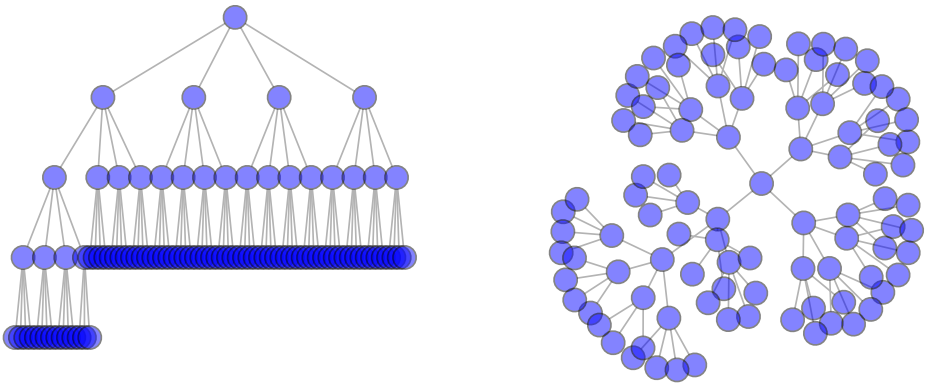
Synopsis: A rooted tree growth model with $c = 4$ children added each growth step until the required number of n vertices is reached.

As in the comparison for T.c2 to T.c3, we see very little change in the mean degree or the density. There are more leaf vertices of degree $k = 1$ to approximately balance the increased mean degree of inner parent vertices. The mean path length is again reduced.

It influence of the unbalanced number of nodes for a complete exponential tree is also more noticeable in the degree distribution, where the number of intermediate vertex degree species is greater.

Property	n100	n400	n900
n	100	400	900
m	99	399	899
Components	1	1	1
Diameter	7	9	10
Girth	0	0	0
Density	0.020	0.005	0.002
$\langle k \rangle$	1.980	1.995	1.998
L	5.099	7.013	8.155
C	0.000	0.000	0.000
E_{glob}	0.358	0.266	0.230
E_{loc}	0.000	0.000	0.000

Table C.28: Properties and statistics for T.c4 graph instances.



(a) (b)

Figure C.88: (a) Tree and (b) force-based layout for T.c4 graph.

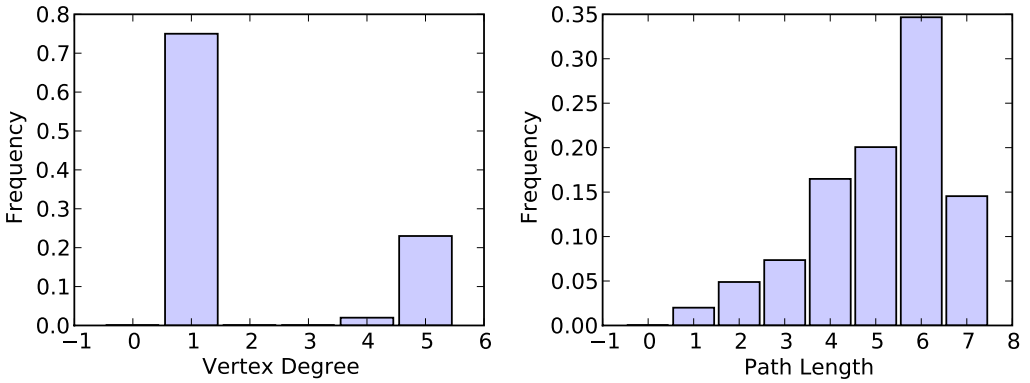


Figure C.89: Vertex degree and path length histograms for T.c4, $n = 100$.

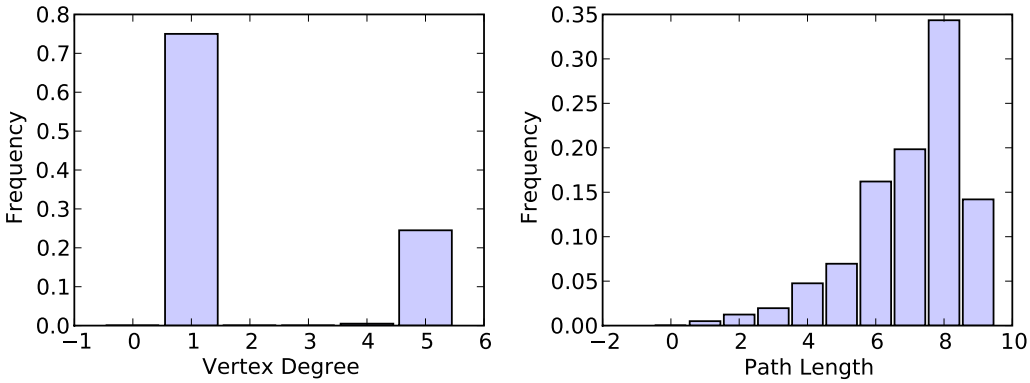


Figure C.90: Vertex degree and path length histograms for T.c4, $n = 400$.

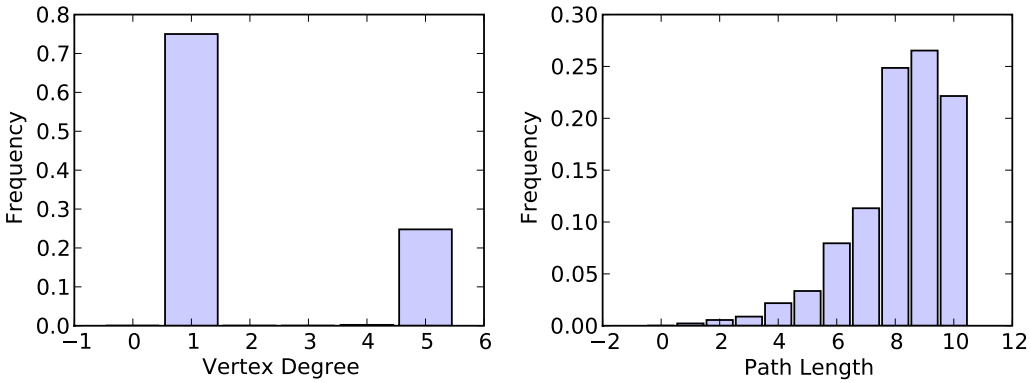


Figure C.91: Vertex degree and path length histograms for T.c4, $n = 900$.

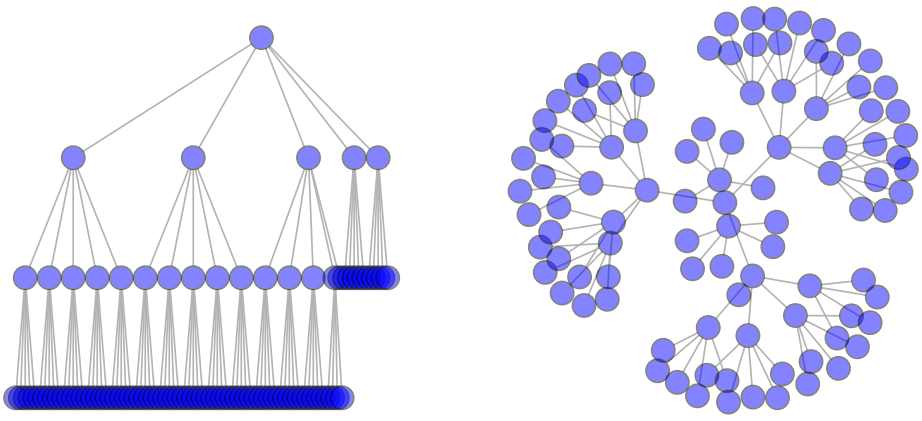
C.5.4 T.c5

Synopsis: A rooted tree growth model with $c = 5$ children added each growth step until the required number of n vertices is reached.

The density and mean degree remain stable with respect to the Tree instances already presented. The influence of an incomplete (unbalanced) tree is again noticeable in the degree distributions. Mean path length again decreases and the number of path length species is few (as seen in the path length distribution histograms).

Property	n100	n400	n900
n	100	400	900
m	99	399	899
Components	1	1	1
Diameter	6	8	9
Girth	0	0	0
Density	0.020	0.005	0.002
$\langle k \rangle$	1.980	1.995	1.998
L	4.648	6.276	7.276
C	0.000	0.000	0.000
E_{glob}	0.382	0.290	0.253
E_{loc}	0.000	0.000	0.000

Table C.29: Properties and statistics for T.c5 graph instances.



(a) (b)

Figure C.92: (a) Tree and (b) force-based layout for T.c5 graph.

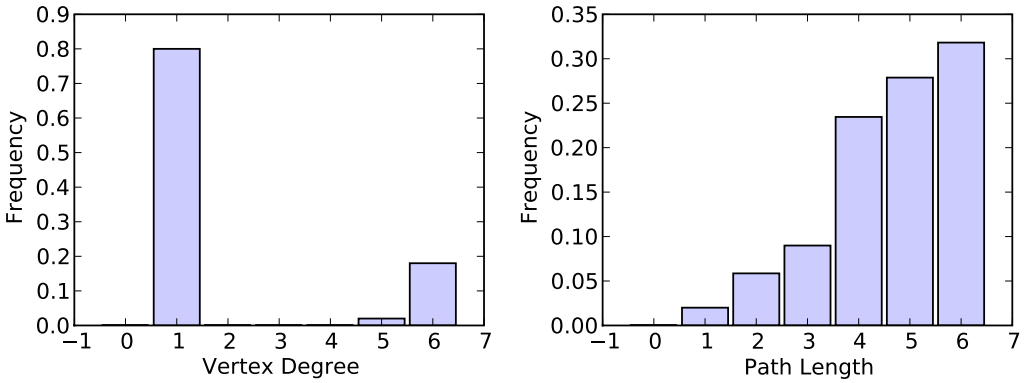


Figure C.93: Vertex degree and path length histograms for T.c5, $n = 100$.

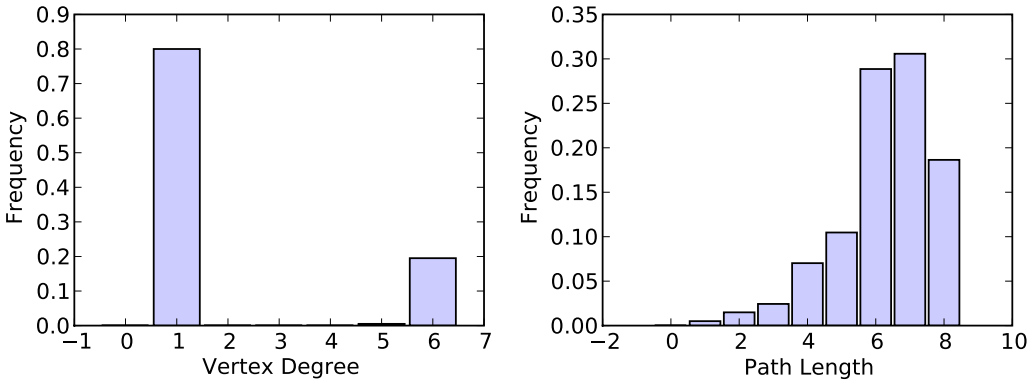


Figure C.94: Vertex degree and path length histograms for T.c5, $n = 400$.

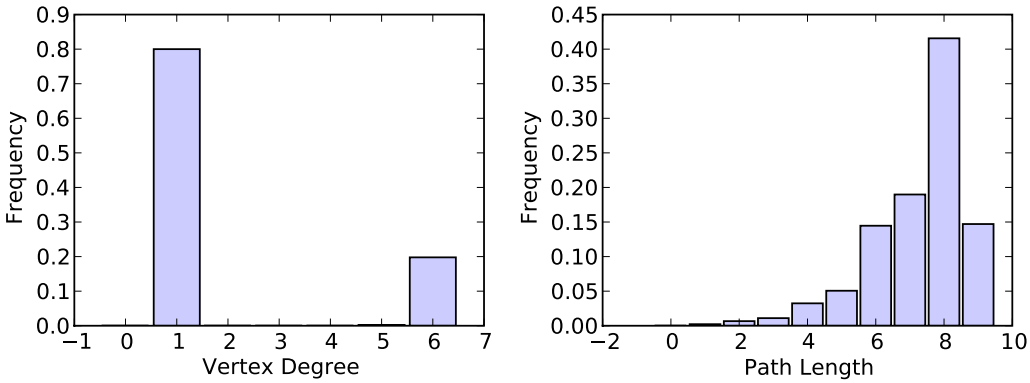


Figure C.95: Vertex degree and path length histograms for T.c5, $n = 900$.

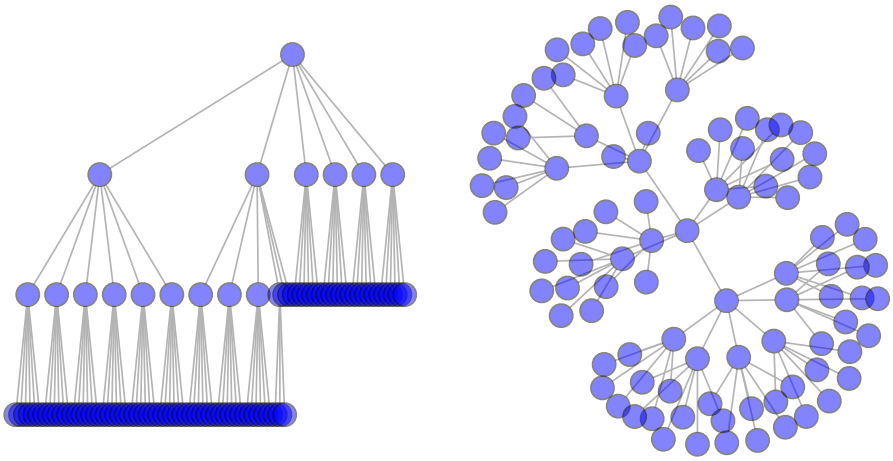
C.5.5 T.c6

Synopsis: A rooted tree growth model with $c = 6$ children added each growth step until the required number of n vertices is reached.

The continuing trends of reduced mean path length and stable density are present. With each parent having so many children, the majority of graph vertices are clearly leaves ($\approx 85\%$) with only a minority of parents ($\approx 15\%$).

Property	n100	n400	n900
n	100	400	900
m	99	399	899
Components	1	1	1
Diameter	6	7	8
Girth	0	0	0
Density	0.020	0.005	0.002
$\langle k \rangle$	1.980	1.995	1.998
L	4.350	5.798	6.719
C	0.000	0.000	0.000
E_{glob}	0.400	0.309	0.270
E_{loc}	0.000	0.000	0.000

Table C.30: Properties and statistics for T.c6 graph instances.



(a) (b)

Figure C.96: (a) Tree and (b) force-based layout for T.c6 graph.

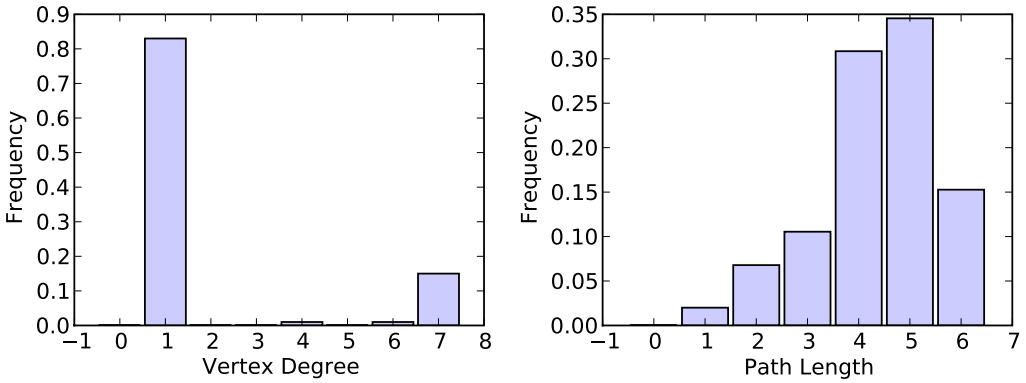


Figure C.97: Vertex degree and path length histograms for T.c6, $n = 100$.

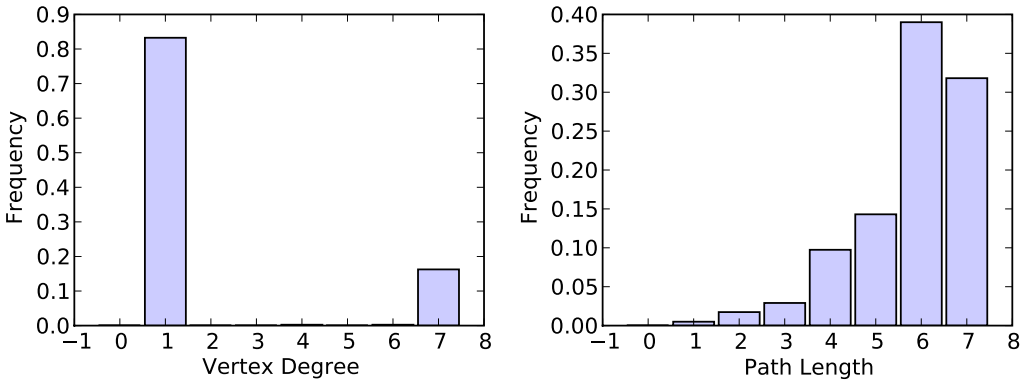


Figure C.98: Vertex degree and path length histograms for T.c6, $n = 400$.

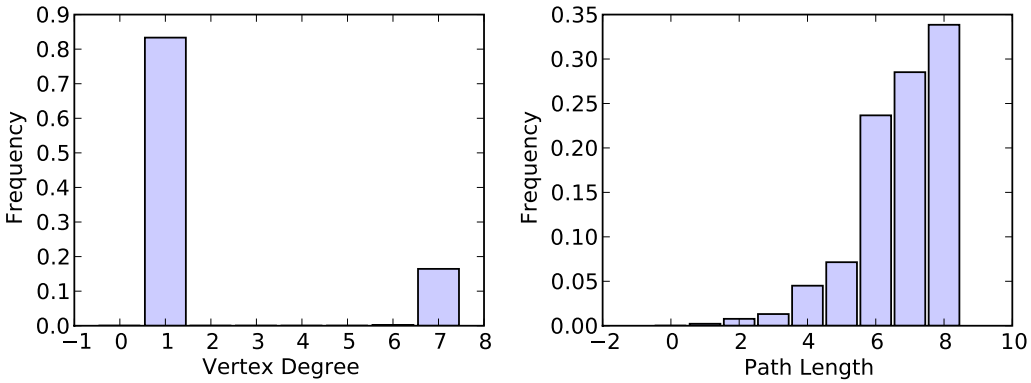


Figure C.99: Vertex degree and path length histograms for T.c6, $n = 900$.

C.5.6 Tree Summary

In all Tree graph cases the vertex number n and edge number m were fixed. Because of the lack of cycles, Girth, C and E_{loc} are all zero. Diameter is essentially two times the number of growth steps required for a balance tree to exceed the required size n. It is clear that the mean path length and diameter decrease the larger the number of children added each growth step, and so the global efficiency increases.

Property	Children c				
	2	3	4	5	6
$n = 100$, $m = 99$, density= 0.020					
Diameter	12	8	7	6	6
L	7.731	5.880	5.099	4.648	4.350
E_{glob}	0.269	0.324	0.358	0.382	0.400
$n = 400$, $m = 399$, density= 0.005					
Diameter	16	11	9	8	7
L	11.427	8.291	7.013	6.276	5.798
E_{glob}	0.181	0.233	0.266	0.290	0.309
$n = 900$, $m = 899$, density= 0.002					
Diameter	18	12	10	9	8
L	13.726	9.717	8.155	7.276	6.719
E_{glob}	0.149	0.198	0.230	0.253	0.270

Table C.31: Comparison of tree models for different children number.

The tree sizes selected for profiles were based simply on the vertex count requirement of prior lattice models. As a result all of the tree instances created are unbalanced, and this can be important with respect to a hierarchical process.

Table C.32 lists the size growth for a complete (balanced) trees. The exponential and hierarchical growth nature becomes clear.

Step	Children c								
	1	2	3	4	5	6	7	8	9
0	1	1	1	1	1	1	1	1	1
1	2	3	4	5	6	7	8	9	10
2	3	7	13	21	31	43	57	73	91
3	4	15	40	85	156	259	400	585	820
4	5	31	121	341	781	1555	2801	4681	7381
5	6	63	364	1365	3906	9331	19608	37449	66430
6	7	127	1093	5461	19531	55987	137257	299593	597871
7	8	255	3280	21845	97656	335923	960800	2396745	5380840
8	9	511	9841	87381	488281	2015539	6725601	19173961	48427561
9	10	1023	29524	349525	2441406	12093235	47079208	153391689	435848050

Table C.32: Balanced tree growth for c children from 1 to 9 for 9 growth steps.

The Star model is equivalent to a single growth step tree graph, where the number of children added is simply $n - 1$.

C.6 Erdös-Rényi (ER)

C.6.1 ER.01

Synopsis: A simple Erdös-Rényi (ER) random graph model $G(n, p)$ for n vertices, with a probability of $p = 0.01$ that an edge will exists between each unique pair of vertices.

On average each graph will have $\langle m \rangle = pn(n-1)/2$ edges. So, for $n = [100, 400, 900]$, the number edges should average to $\langle m \rangle = [49.5, 798, 4045.5]$ respectively.

Since this is a stochastic graph model, statistics for each model are sampled for 30 instances for each scale size n. Note the large number of components for the smaller scale graphs, however the $n = 900$ is usually a single connected component.

It can be seen that the density of the graph is equal to the value of p used, and this is also approximately equal to the clustering coefficient C. The lack of dense or regular local structure results in an overall low local efficiency at all scales, however the global efficiency does improve as the scale increases and the number of components is typically 1.

It is expected that the degree distribution will be Poisson with an average degree $\langle k \rangle = p(n-1) \approx pn$, and this can be confirmed in the vertex degree histograms (when the number of components approaches one) and against the values in Table C.33. The expected values for $\langle k \rangle$ for the scales of $n = [100, 400, 900]$ are $[0.99, 3.99, 8.99]$ respectively which matches reasonably well.

Property	n100	n400	n900
n	100	400	900
m	50.700	791.467	4047.000
Components	50.067	8.700	1.067
Diameter	10.000	9.533	5.867
Girth	1.767	3.000	3.000
Density	0.010	0.010	0.010
$\langle k \rangle$	1.014	3.957	8.993
L	3.819	4.469	3.344
C	0.010	0.010	0.010
E_{glob}	0.034	0.372	0.475
E_{loc}	0.003	0.009	0.011

Table C.33: Properties and statistics for ER.01 graph instances.

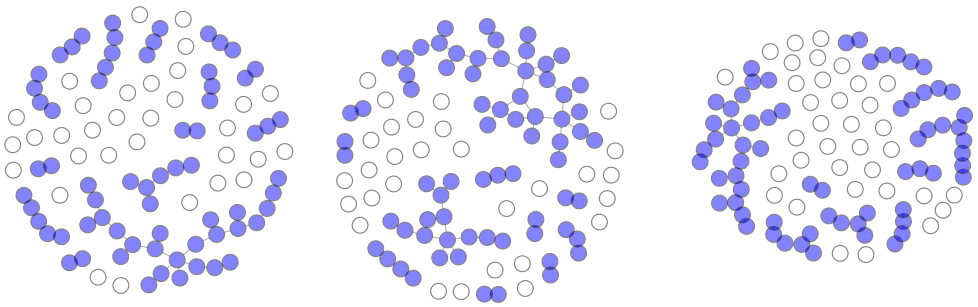


Figure C.100: Force-based layout for three different ER.01 instances.

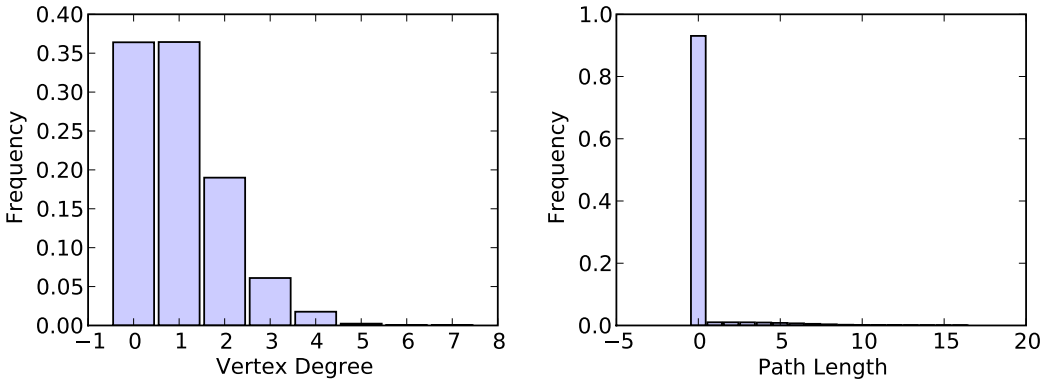


Figure C.101: Vertex degree and path length histograms for ER.01, $n = 100$.

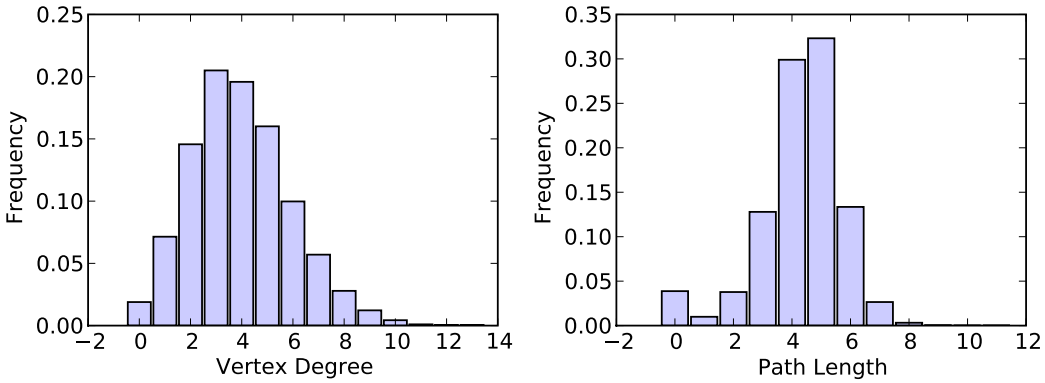


Figure C.102: Vertex degree and path length histograms for ER.01, $n = 400$.

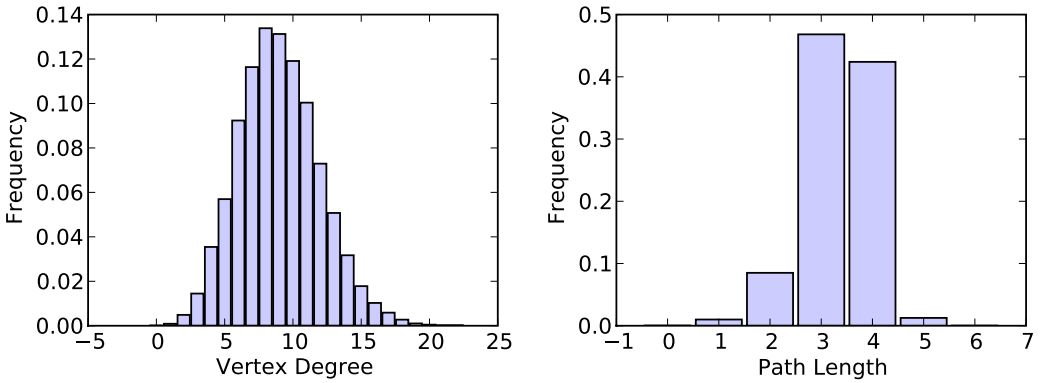


Figure C.103: Vertex degree and path length histograms for ER.01, $n = 900$.

C.6.2 ER.02

Synopsis: An ER $G(n, p)$ random graph model for $p = 0.2$.

For instances of this model and graph sizes of $n = [100, 400, 900]$, it is expected that the number of edges should average to $\langle m \rangle = [99, 1596, 8091]$ respectively, and that the average degree $\langle k \rangle$ will be $[1.98, 7.98, 17.98]$ respectively.

Note that the number of components for the $n = 100$ scale is quite large, however for the larger graph scales, with $n(n-1)/2$ edges, there is also greater number of opportunities for edges and this increases the probability of a single component forming.

Property	n100	n400	n900
n	100	400	900
m	98.600	1603.567	8103.000
Components	17.600	1.033	1.000
Diameter	13.700	5.400	4.000
Girth	3.133	3.000	3.000
Density	0.020	0.020	0.020
$\langle k \rangle$	1.972	8.018	18.007
L	5.626	3.104	2.669
C	0.023	0.020	0.020
E_{glob}	0.219	0.505	0.559
E_{loc}	0.014	0.021	0.025

Table C.34: Properties and statistics for ER.02 graph instances.

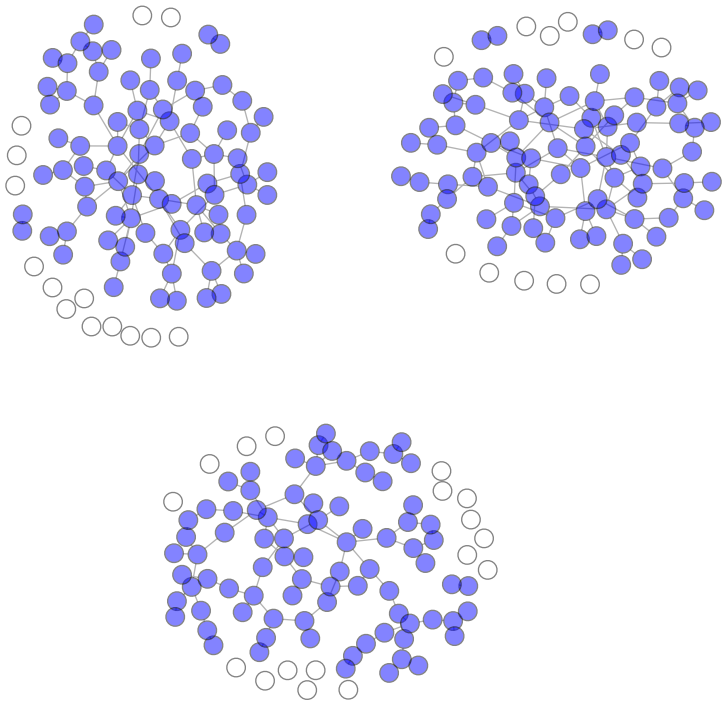


Figure C.104: Force-based layout for three different ER.02 instances.

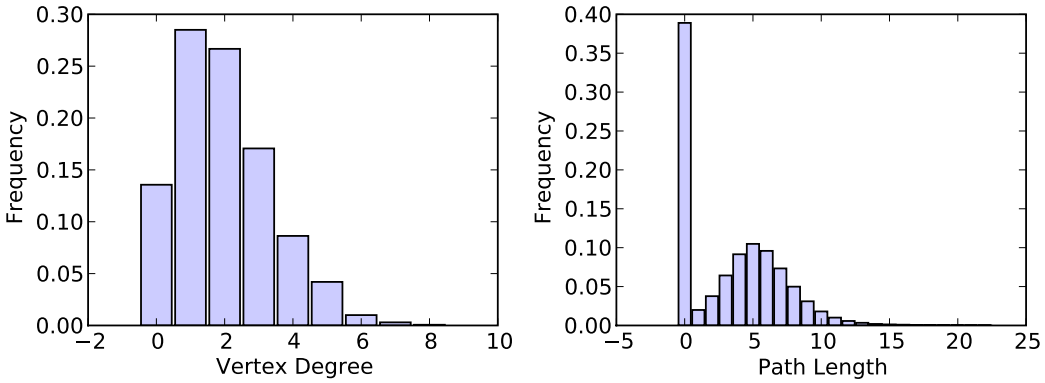


Figure C.105: Vertex degree and path length histograms for ER.02, $n = 100$.

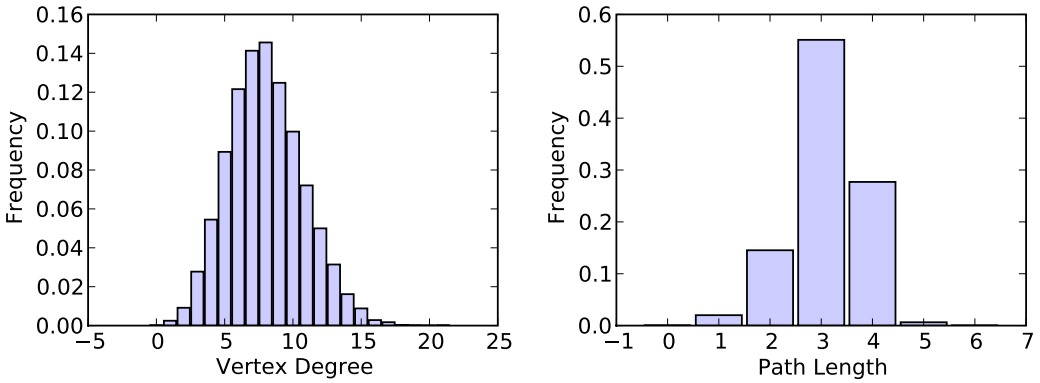


Figure C.106: Vertex degree and path length histograms for ER.02, $n = 400$.

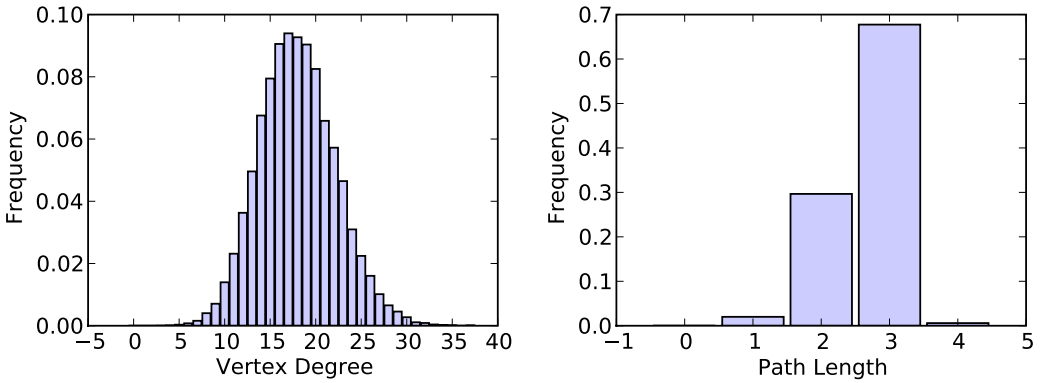


Figure C.107: Vertex degree and path length histograms for ER.02, $n = 900$.

C.6.3 ER.03

Synopsis: The ER $G(n, p)$ random graph model for $p = 0.3$.

It is expected that for graph sizes of $n = [100, 400, 900]$, the number edges should average to $\langle m \rangle = [148.5, 2394, 12136.5]$ respectively, and that the average degree $\langle k \rangle$ will be $[2.97, 11.97, 26.97]$ respectively. A sample of 30 instances for each graph size is used for profile details.

Property	n100	n400	n900
n	100	400	900
m	148.733	2388.867	12178.033
Components	6.267	1.000	1.000
Diameter	9.500	4.300	3.400
Girth	3.033	3.000	3.000
Density	0.030	0.030	0.030
$\langle k \rangle$	2.975	11.944	27.062
L	4.164	2.680	2.400
C	0.029	0.030	0.030
E_{glob}	0.381	0.560	0.604
E_{loc}	0.024	0.037	0.058

Table C.35: Properties and statistics for ER.03 graph instances.

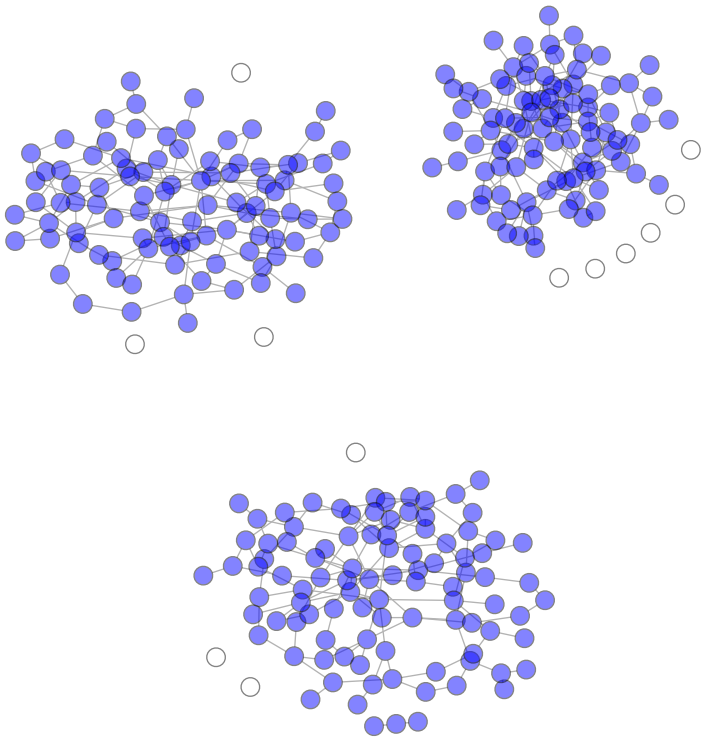


Figure C.108: Force-based layout for three different ER.03 instances.

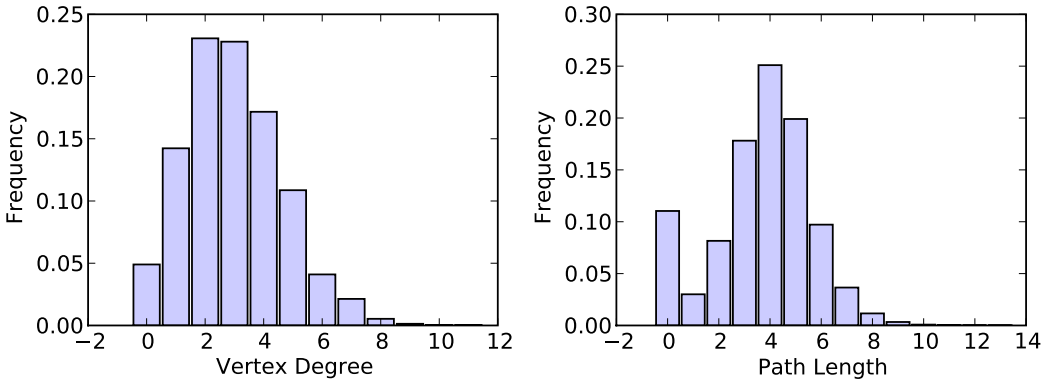


Figure C.109: Vertex degree and path length histograms for ER.03, $n = 100$.

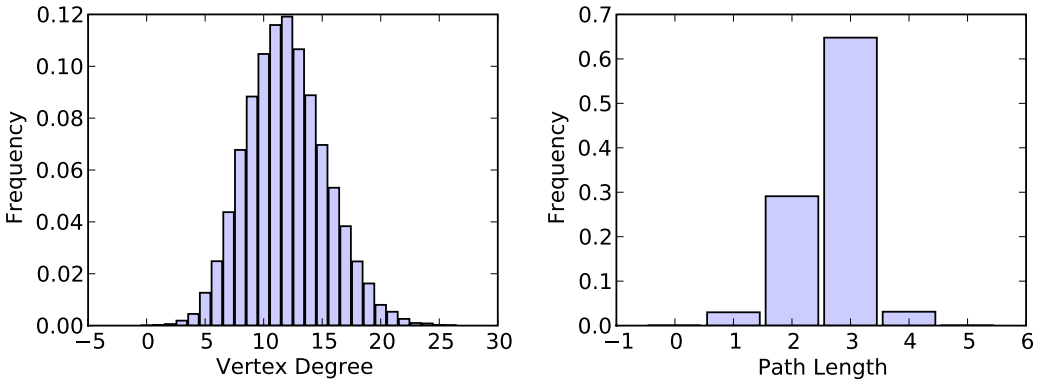


Figure C.110: Vertex degree and path length histograms for ER.03, $n = 400$.

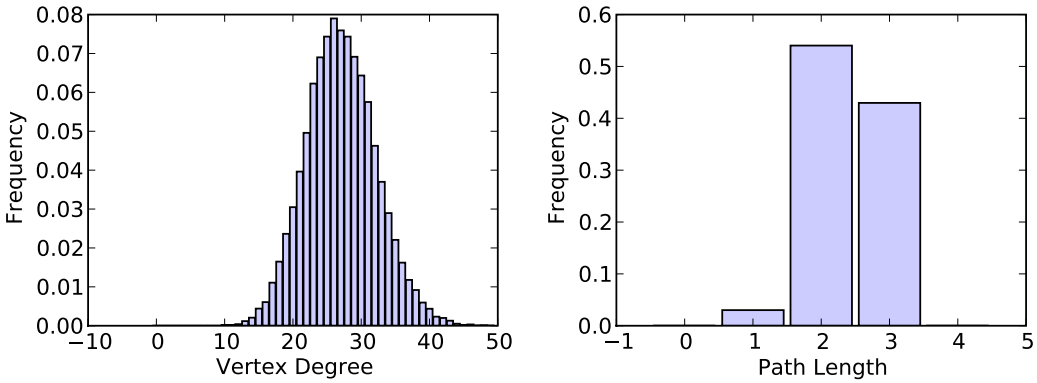


Figure C.111: Vertex degree and path length histograms for ER.03, $n = 900$.

C.6.4 ER.04

Synopsis: The ER $G(n, p)$ random graph model for $p = 0.4$.

For the graph sizes of $n = [100, 400, 900]$, the expected number edges should average to $\langle m \rangle = [198, 3192, 16182]$ respectively, and the average degree $\langle k \rangle$ to be $[3.96, 15.96, 35.96]$ respectively. A sample of 30 instances for each graph size is used for profile details.

Property	n100	n400	n900
n	100	400	900
m	201.467	3183.467	16188.667
Components	2.500	1.000	1.000
Diameter	7.167	4.000	3.000
Girth	3.000	3.000	3.000
Density	0.041	0.040	0.040
$\langle k \rangle$	4.029	15.917	35.975
L	3.395	2.471	2.188
C	0.039	0.040	0.040
E_{glob}	0.474	0.595	0.642
E_{loc}	0.037	0.063	0.159

Table C.36: Properties and statistics for ER.04 graph instances.

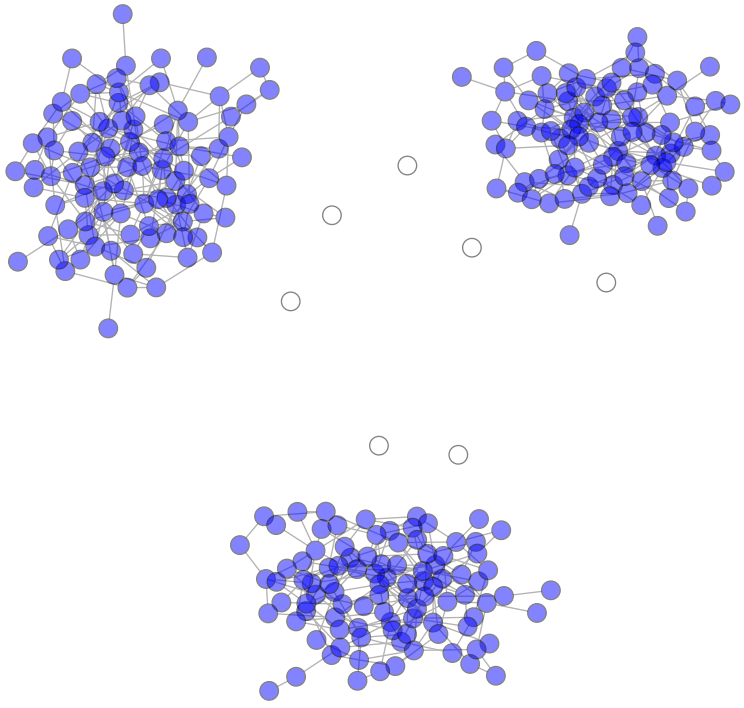


Figure C.112: Force-based layout for three different ER.04 instances.

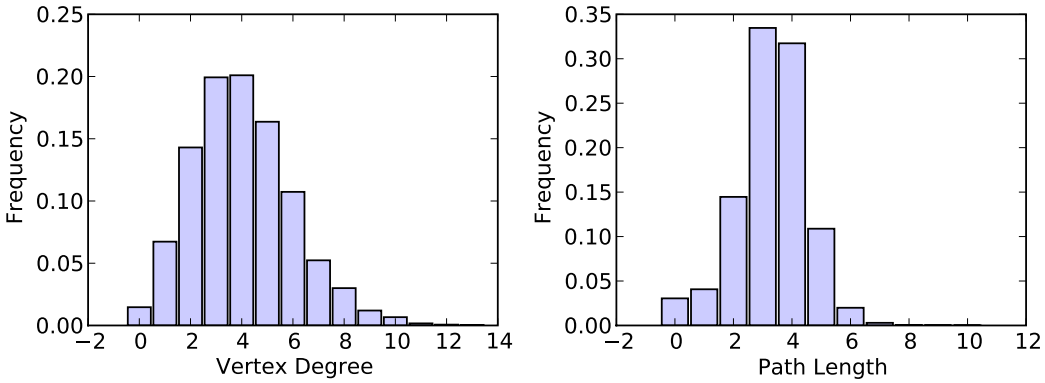


Figure C.113: Vertex degree and path length histograms for ER.04, $n = 100$.

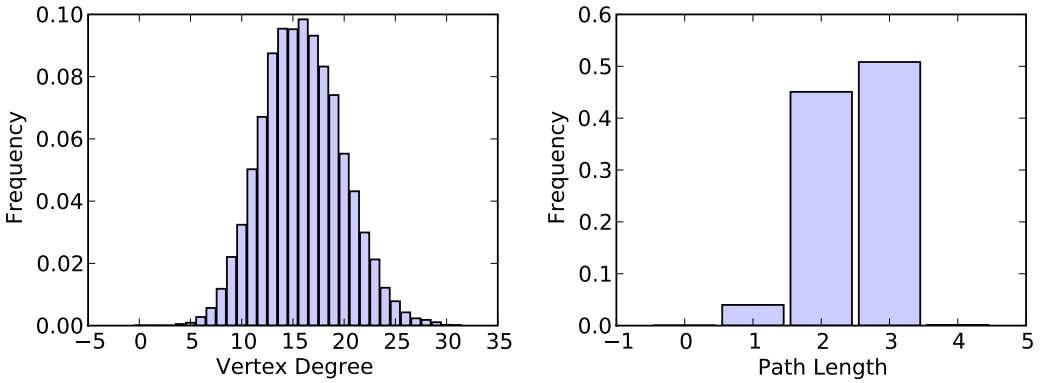


Figure C.114: Vertex degree and path length histograms for ER.04, $n = 400$.

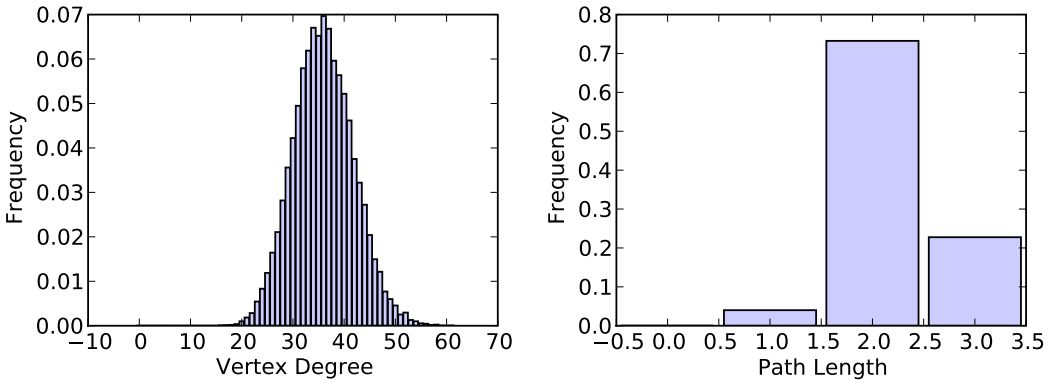


Figure C.115: Vertex degree and path length histograms for ER.04, $n = 900$.

C.6.5 ER.05

Synopsis: The ER $G(n, p)$ random graph model for $p = 0.5$.

For the graph sizes of $n = [100, 400, 900]$, the expected number edges should average to $\langle m \rangle = [247.5, 3990, 20227.5]$ respectively, and the average degree $\langle k \rangle$ to be $[4.95, 19.95, 44.95]$ respectively. A sample of 30 instances for each graph size is used for profile details.

Property	n100	n400	n900
n	100	400	900
m	246.067	3983.067	20236.067
Components	1.933	1.000	1.000
Diameter	6.033	3.533	3.000
Girth	3.000	3.000	3.000
Density	0.050	0.050	0.050
$\langle k \rangle$	4.921	19.915	44.969
L	3.029	2.302	2.050
C	0.056	0.050	0.050
E_{glob}	0.517	0.625	0.667
E_{loc}	0.060	0.109	0.322

Table C.37: Properties and statistics for ER.05 graph instances.

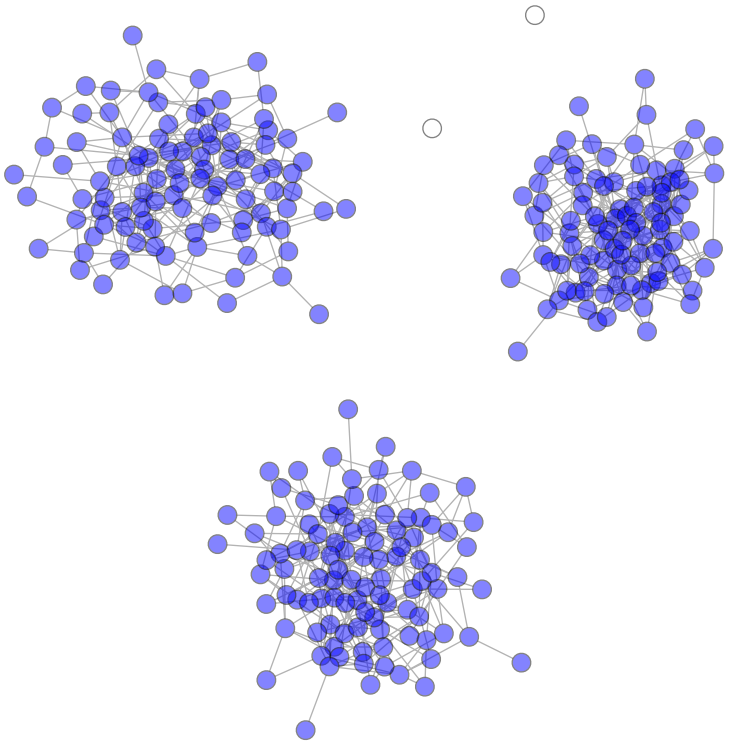


Figure C.116: Force-based layout for three different ER.05 instances.

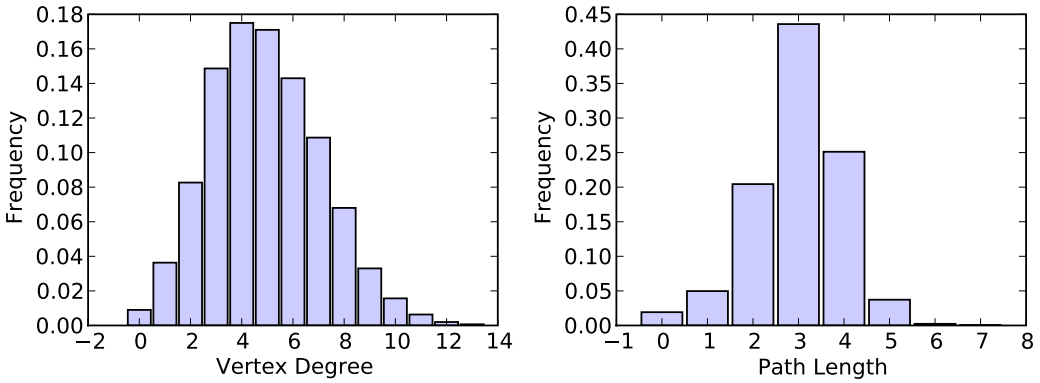


Figure C.117: Vertex degree and path length histograms for ER.05, $n = 100$.

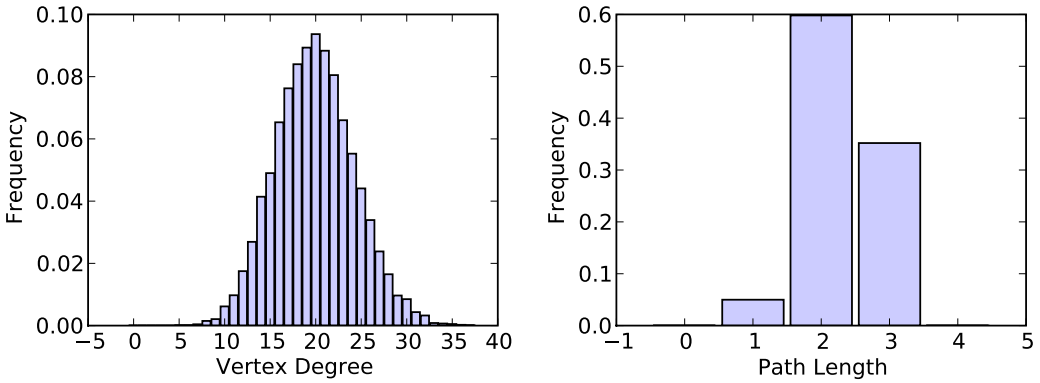


Figure C.118: Vertex degree and path length histograms for ER.05, $n = 400$.

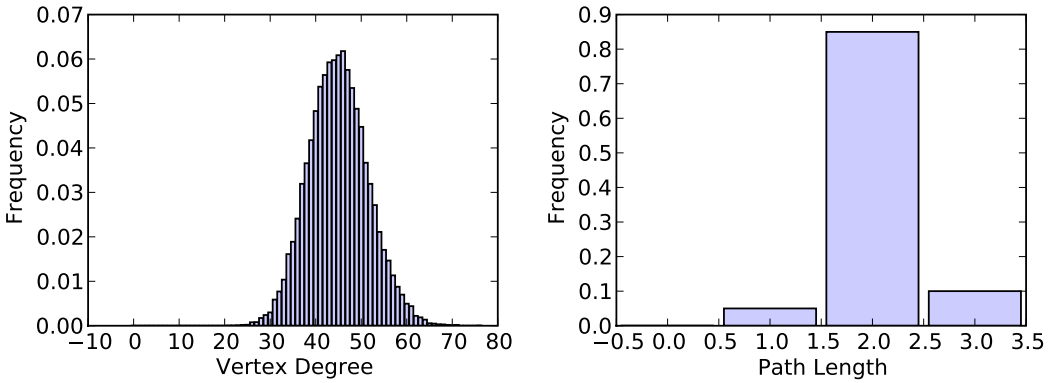


Figure C.119: Vertex degree and path length histograms for ER.05, $n = 900$.

C.6.6 ER Summary

In an ER graph instance of $G(n, p)$, the average Density and clustering coefficient C values are known to be essentially equal to the specified edge probability p.

An interesting feature of the ER $G(n, p)$ model is that all properties, including the direct Density $\approx C \approx p$ relationship, have definite analytical relationships to the specified values of n and p. This model does not require an understanding of any other processes or features.

Property	Edge Probability p				
	ER.01	ER.02	ER.03	ER.04	ER.05
$n = 100$					
m	50.7	98.6	148.7	201.5	246.1
Components	50.067	17.600	6.267	2.500	1.933
Diameter	10.000	13.700	9.500	7.167	6.033
Girth	1.767	3.133	3.033	3.000	3.000
Density $\approx C \approx p$	0.010	0.020	0.030	0.040	0.050
$\langle k \rangle$	1.014	1.972	2.975	4.029	4.921
L	3.819	5.626	4.164	3.395	3.029
E_{glob}	0.034	0.219	0.381	0.474	0.517
E_{loc}	0.003	0.014	0.024	0.037	0.060
$n = 400$					
m	791.5	1603.6	2388.9	3183.5	3983.1
Components	8.700	1.033	1.000	1.000	1.000
Diameter	9.533	5.400	4.300	4.000	3.533
Girth	3.000	3.000	3.000	3.000	3.000
Density $\approx C \approx p$	0.010	0.020	0.030	0.040	0.050
$\langle k \rangle$	3.957	8.018	11.944	15.917	19.915
L	4.469	3.104	2.680	2.471	2.302
E_{glob}	0.372	0.505	0.560	0.595	0.625
E_{loc}	0.009	0.021	0.037	0.063	0.109
$n = 900$					
m	4047.0	8103.0	12178.0	16188.7	20236.1
Components	1.067	1.000	1.000	1.000	1.000
Diameter	5.867	4.000	3.400	3.000	3.000
Girth	3.000	3.000	3.000	3.000	3.000
Density $\approx C \approx p$	0.010	0.020	0.030	0.040	0.050
$\langle k \rangle$	8.993	18.007	27.062	35.975	44.969
L	3.344	2.669	2.400	2.188	2.050
E_{glob}	0.475	0.559	0.604	0.642	0.667
E_{loc}	0.011	0.025	0.058	0.159	0.322

Table C.38: Comparison of ER model properties for n and p values.

C.7 Watts-Strogatz (WS)

C.7.1 WS.001

Synopsis: A regular one dimensional (1D) circular lattice with a neighbourhood (nei) size of 3 and a rewiring probability of $p = 0.001$.

This topology is created using the Watts-Strogatz small-world model. It uses a regular lattice, typically a ring or toroid, and rewires exiting edges (or alternatively add new edges). As the amount of rewiring p is increased, the graphs mean path length reduces (on average) while the initial highly clustered locally efficiency lattice features remain. If rewiring is increased to $p = 1.0$ the graph reverts to a simple random topology.

Property	n100	n400	n900
n	100	400	900
m	300	1200	2700
Components	1.000	1.000	1.000
Diameter	17.000	56.967	97.967
Girth	3.000	3.000	3.000
Density	0.061	0.015	0.007
$\langle k \rangle$	6.000	6.000	6.000
L	8.191	24.094	40.616
C	0.596	0.596	0.597
E_{glob}	0.310	0.137	0.084
E_{loc}	0.852	0.852	0.852

Table C.39: Properties and statistics for WS.001 graph instances.

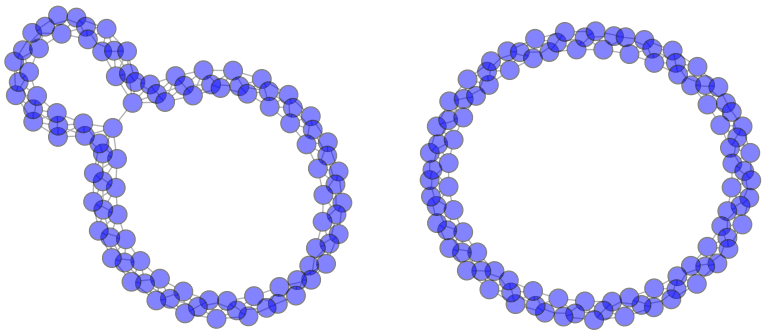


Figure C.120: Force-based layout for two different WS.001 instances.

In the instances used here, a ring lattice (single dimension and circular) with a neighbourhood size of $nei = 3$ is used and results in a mean degree of $k = 6$.

With a small rewiring probability of $p = 0.001$ the graph retains the base lattice characteristics (as can be see visually in Figure C.120 and in the vertex degree histograms). A sample of 30 instances for lattices sizes of n of 100, 400 and 900 is averaged and shown in Table C.39.

Note the large amount of clustering ($C \approx 0.6$) but also a large mean path length whose value is related to the neighbourhood size and the total number of nodes in the base lattice. The path length histograms are spread over a wide range of length values, due to the nature of the base lattice.

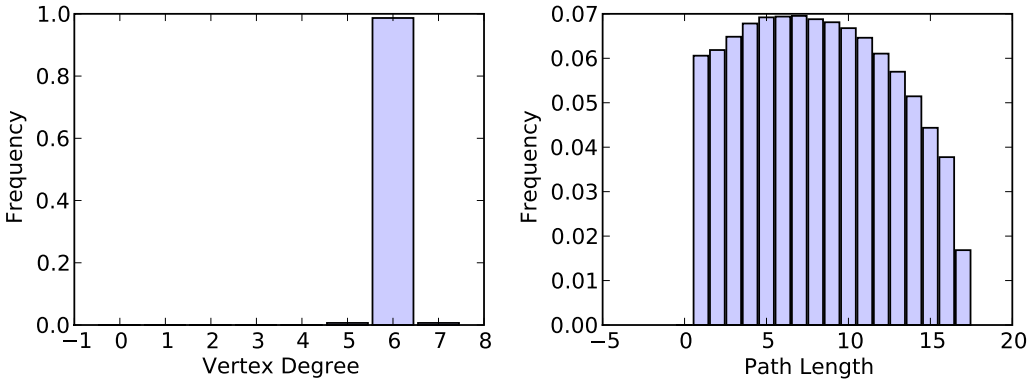


Figure C.121: Vertex degree and path length histograms for WS.001, $n = 100$.

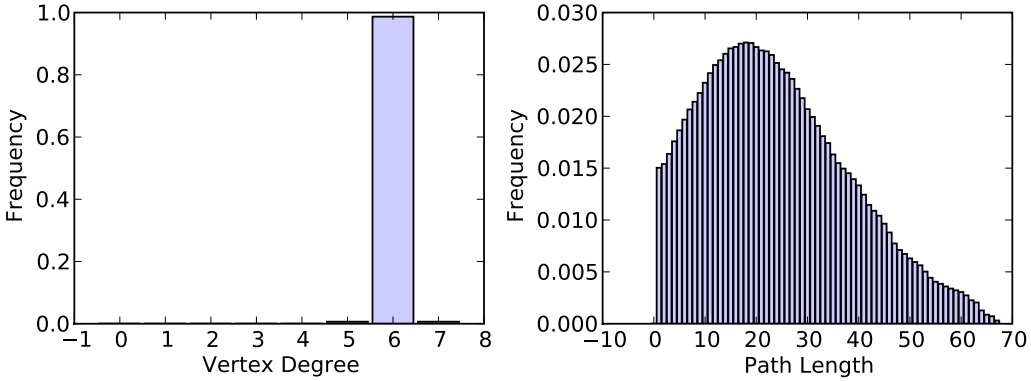


Figure C.122: Vertex degree and path length histograms for WS.001, $n = 400$.

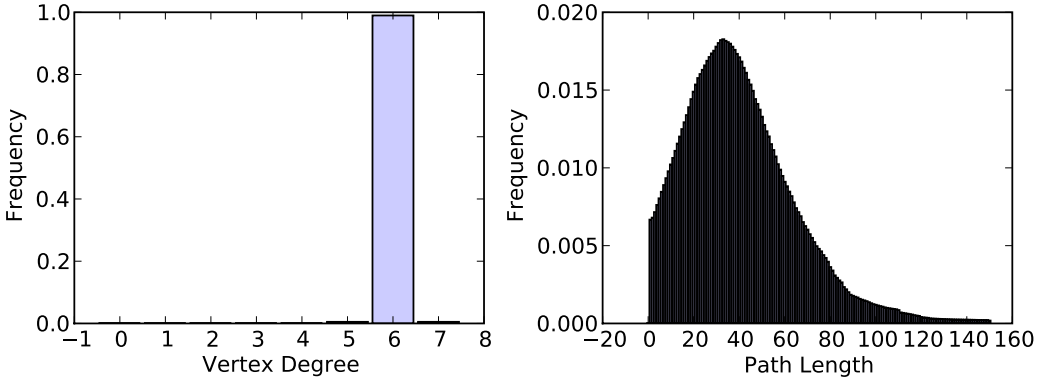


Figure C.123: Vertex degree and path length histograms for WS.001, $n = 900$.

C.7.2 WS.01

Synopsis: A regular one dimensional circular lattice with a neighbourhood size of $nei = 3$ and a rewiring probability of $p = 0.01$.

This level of lattice rewiring brings the initial lattice into the "small-world" region of characteristics, where a relatively high clustering value C is retained but also a much reduced mean path length L.

The graph characteristics are altered towards those of a random (ER) graph which can be seen in the layout examples, statistics and histograms.

Property	n100	n400	n900
n	100	400	900
m	300	1200	2700
Components	1.000	1.000	1.000
Diameter	12.267	20.067	24.633
Girth	3.000	3.000	3.000
Density	0.061	0.015	0.007
$\langle k \rangle$	6.000	6.000	6.000
L	5.655	9.072	11.165
C	0.568	0.563	0.562
E_{glob}	0.373	0.238	0.191
E_{loc}	0.825	0.818	0.818

Table C.40: Properties and statistics for WS.01 graph instances.

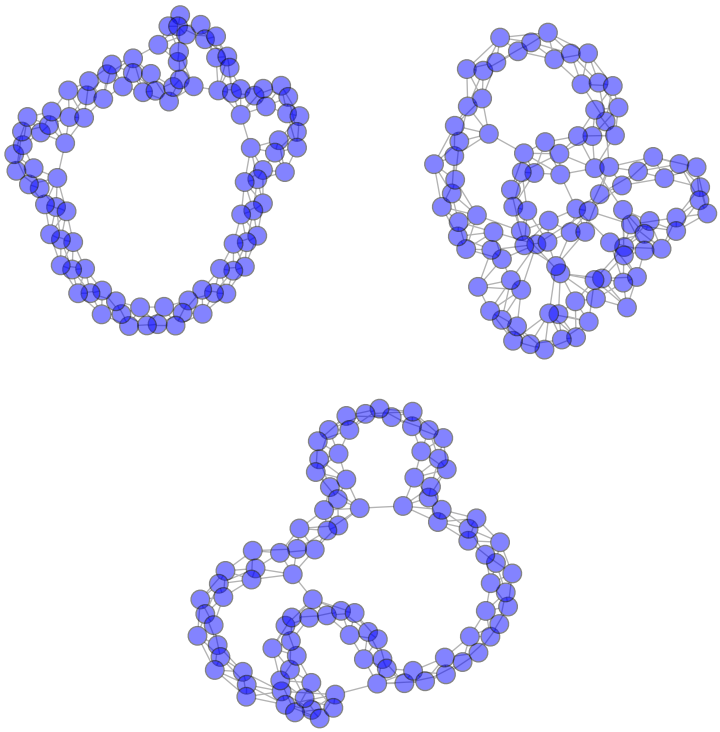


Figure C.124: Force-based layout for three different WS.01 instances.

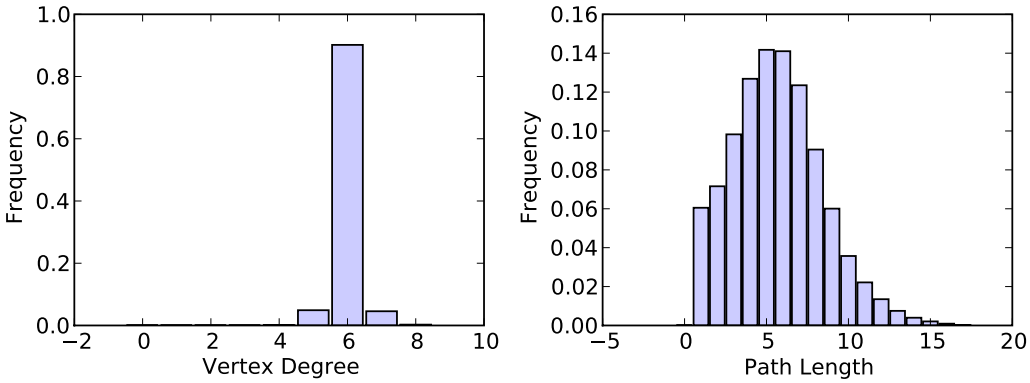


Figure C.125: Vertex degree and path length histograms for WS.01, $n = 100$.

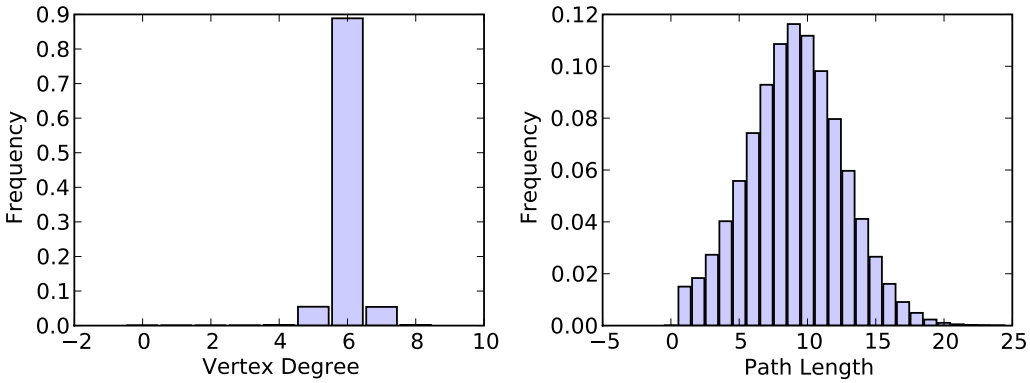


Figure C.126: Vertex degree and path length histograms for WS.01, $n = 400$.

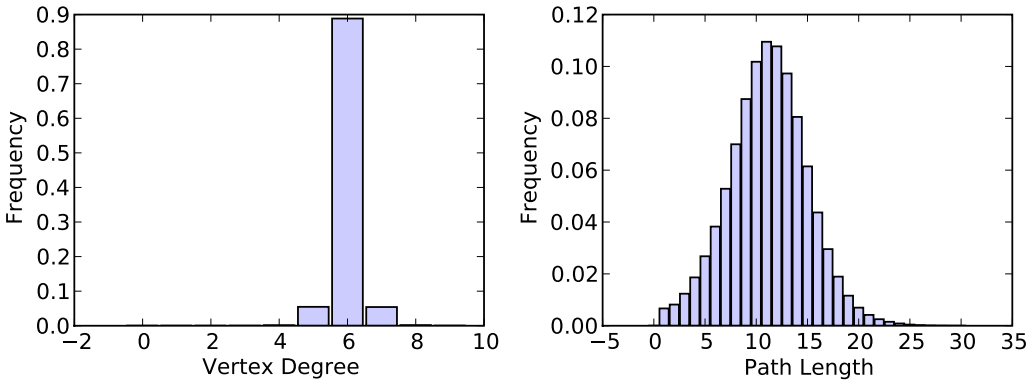


Figure C.127: Vertex degree and path length histograms for WS.01, $n = 900$.

C.7.3 WS.1

Synopsis: A regular one dimensional circular lattice with a neighbourhood size of $nei = 3$ and a rewiring probability of $p = 0.1$.

In this case, the level of lattice rewiring brings the graph towards the simpler characteristics of a random (ER) graph. This can be seen in the layout examples, and the statistics and histograms for the average of 30 instances at each size n.

Property	n100	n400	n900
n	100	400	900
m	300	1200	2700
Components	1.000	1.000	1.000
Diameter	6.133	7.867	8.967
Girth	3.000	3.000	3.000
Density	0.061	0.015	0.007
$\langle k \rangle$	6.000	6.000	6.000
L	3.272	4.391	5.082
C	0.328	0.313	0.312
E_{glob}	0.505	0.393	0.345
E_{loc}	0.518	0.496	0.492

Table C.41: Properties and statistics for WS.1 graph instances.

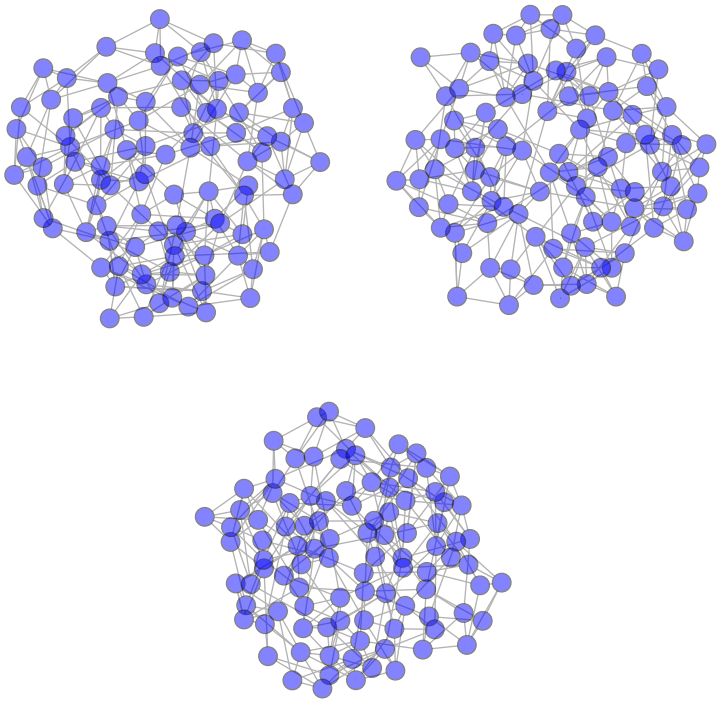


Figure C.128: Force-based layout for three different WS.1 instances.

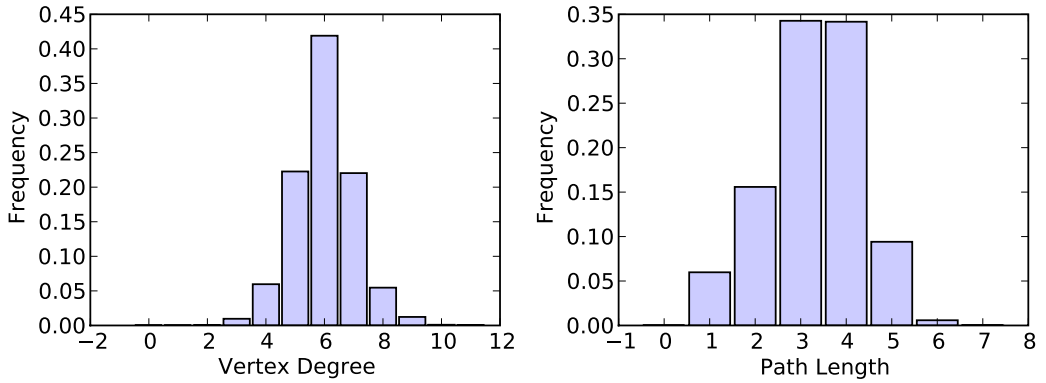


Figure C.129: Vertex degree and path length histograms for WS.1, $n = 100$.

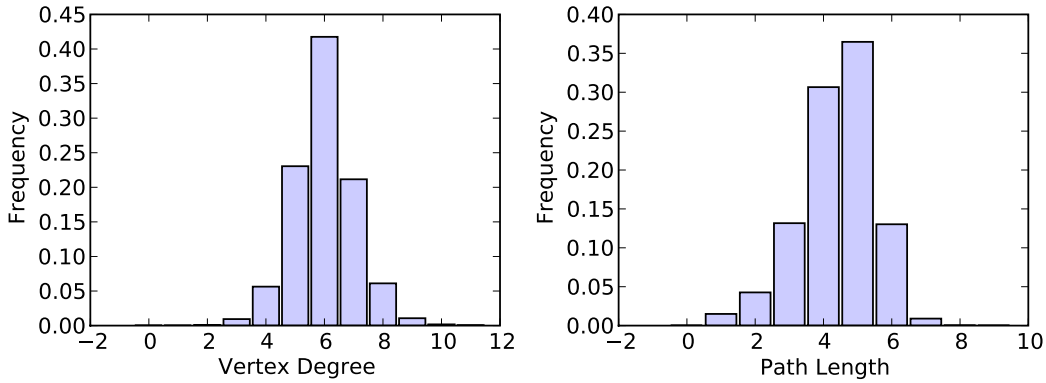


Figure C.130: Vertex degree and path length histograms for WS.1, $n = 400$.

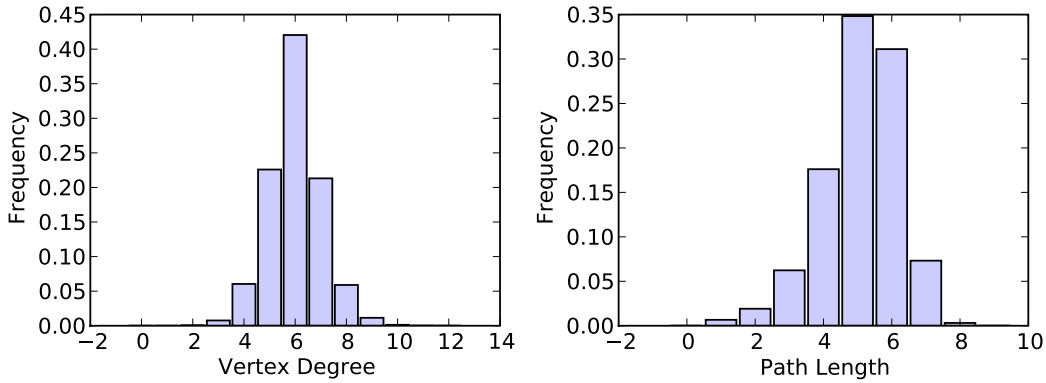


Figure C.131: Vertex degree and path length histograms for WS.1, $n = 900$.

C.7.4 WS Summary

A summary of the comparative statistics of the Watts-Strogatz (WS) topology models is presented in Table C.42. For each graph size n selected, three rewiring probabilities p were used to broadly cover the range of the model from a regular lattice ($p = 0.001$) towards a disrupted ($p = 0.1$) random model.

The number of edges is set at $m = 3n$ in each case, and the density decreases as scale n increases. Diameter in each scale is large, as expected for a regular lattice model, and drops to levels familiar to random graph instances.

It is expected that for instances with low p the mean path length L will be high, and should drop as a greater degree of rewiring is applied. This is clearly evident in each sample of instances, and is supported by the increased global efficiency as L decreases.

Similarly, clustering C is high in the initial near-regular lattice instance, and stays relatively high despite some rewiring ($p = 0.01$) unlike L which dropped at this level of p. Eventually C begins to decrease later when the amount of rewiring becomes significant ($p = 0.1$). This is also reflected in the local efficiency values which are initially high and fall significantly only when the amount of rewiring is large.

The ability of a graph to have both low mean path length and relatively high levels of clustering are the defining characteristics of small-world networks.

p	Diameter	L	C	E_{glob}	E_{loc}
$n = 100$, $m = 300$, density= 0.061					
0.001	17.000	8.191	0.596	0.310	0.852
0.01	12.267	5.655	0.568	0.373	0.825
0.1	6.133	3.272	0.328	0.505	0.518
$n = 400$, $m = 1200$, Density= 0.015					
0.001	56.967	24.094	0.596	0.137	0.852
0.01	20.067	9.072	0.563	0.238	0.818
0.1	7.867	4.391	0.313	0.393	0.496
$n = 900$, $m = 2700$, Density= 0.007					
0.001	97.967	40.616	0.597	0.084	0.852
0.01	24.633	11.165	0.562	0.191	0.818
0.1	8.967	5.082	0.312	0.345	0.492

Table C.42: WS model comparison for each size and rewiring probability.

C.8 Barabási-Albert (BA)

Synopsis: The Barabási-Albert (BA) preferential attachment growth model, using a simple power relationship of $p = 1$.

As discussed in the thesis, of particular note for graphs of this model is the linear slope of a vertex degree plot (or histogram) using a log-log scale (not shown). This linear scale-free behaviour is indicative of preferential attachment and growth models. In this basic form cycles are not formed, and so the girth and clustering values are zero. Overall cost is also very low, with overall robustness and several critical hub vertices.

Property	n100	n400	n900
n	100	400	900
m	99.000	399.000	899.000
Components	1.000	1.000	1.000
Diameter	10.800	14.433	16.867
Girth	0.000	0.000	0.000
Density	0.020	0.005	0.002
$\langle k \rangle$	1.980	1.995	1.998
L	4.692	6.010	6.749
C	0.000	0.000	0.000
E_{glob}	0.394	0.316	0.285
E_{loc}	0.000	0.000	0.000

Table C.43: Properties and statistics for BA.p1 graph instances.

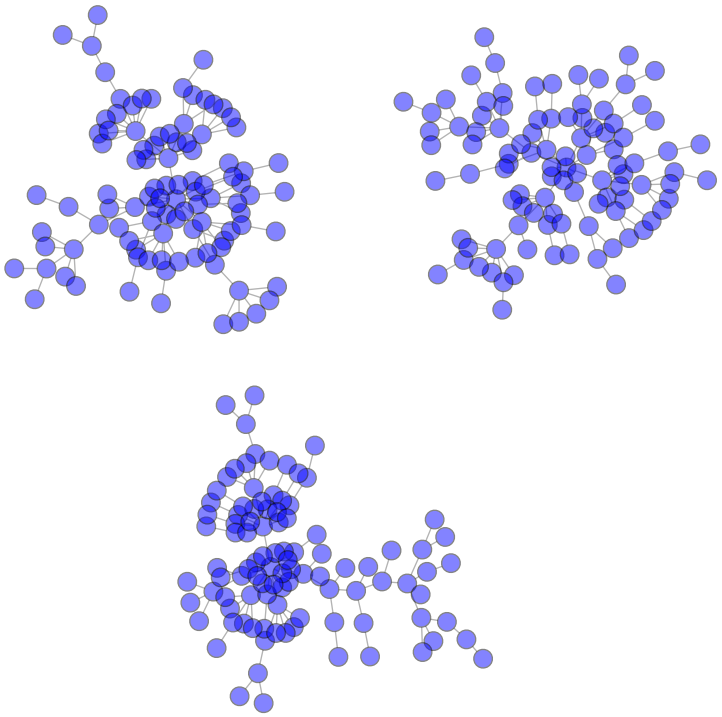


Figure C.132: Force-based layout for three different BA.p1 instances.

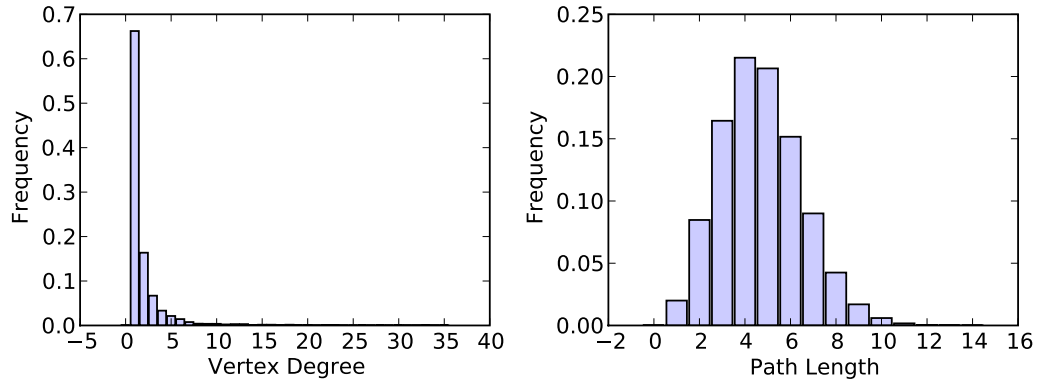


Figure C.133: Vertex degree and path length histograms for BA.p1, $n = 100$.

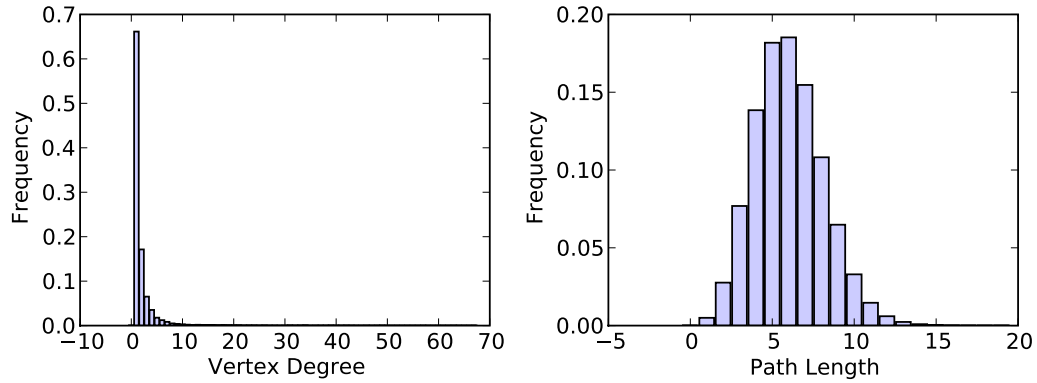


Figure C.134: Vertex degree and path length histograms for BA.p1, $n = 400$.

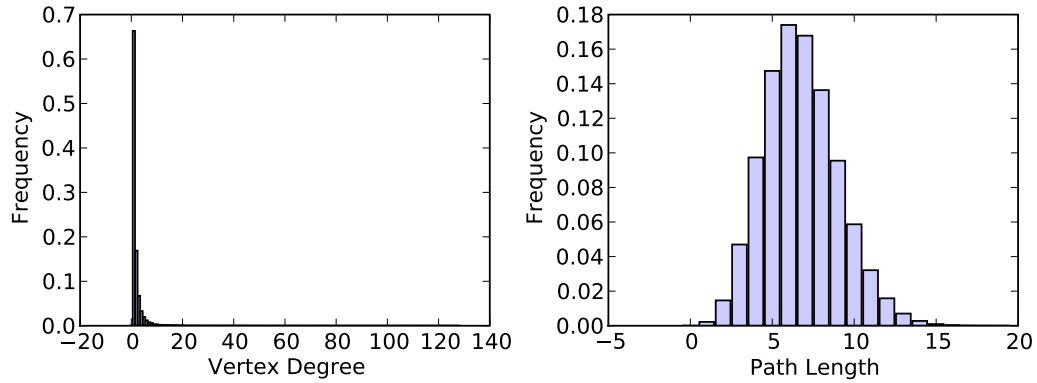


Figure C.135: Vertex degree and path length histograms for BA.p1, $n = 900$.

C.9 Merge-Regenerate (MR)

Synopsis: The Merge-Regenerate graph generation method begins initially with an Erdös-Rényi (ER) model which is then modified with a "merge-regenerate" (MR) process that merges connected vertices and adds replacement vertices with a mean number of edges ($k = 5$ in this case). The MR process is applied a number of times (50 for the $n = 100$ case here).

Property	n100	n400	n900
n	100	400	900
m	228.533	749.833	1368.933
Components	5.200	27.333	95.233
Diameter	6.400	9.233	12.167
Girth	3.000	3.000	3.000
Density	0.046	0.009	0.003
$\langle k \rangle$	4.571	3.749	3.042
L	3.017	4.214	5.291
C	0.070	0.014	0.005
E_{glob}	0.486	0.352	0.265
E_{loc}	0.077	0.013	0.004

Table C.44: Properties and statistics for MR.5 graph instances.

With an underlying ER model, the initial features of an MR graph match an equivalent ER profile, and as the number of MR steps increases features can resemble parts of Tree and Star graphs. It is possible and likely that some vertices remain isolated. (See Figure C.136 where isolated vertices are indicated as unfilled circles.)

Cycles and triangles are created so a minimal girth of 3 is frequent. The overall cost of the graph is relatively low, proportional to the k selected and the number of times the MR process is applied. The mean path length L also remains very low due to the concentrating nature of merges.

Unfortunately the MR model introduces several parameters which are not explored in detail here. These include the initial probability of connections p or the number of connections m in the base ER graph, the mean degree k selected when adding new regenerated vertices (or the type of distribution used), and the number of steps – all influence the final graph properties.

The values selected here were arbitrarily selected to be suitable for interesting features; low mean path length similar to an ER or WS model, but with strong clustering (and its influence on mean path length distribution) more typical of growth such as the BA or hierarchical model.

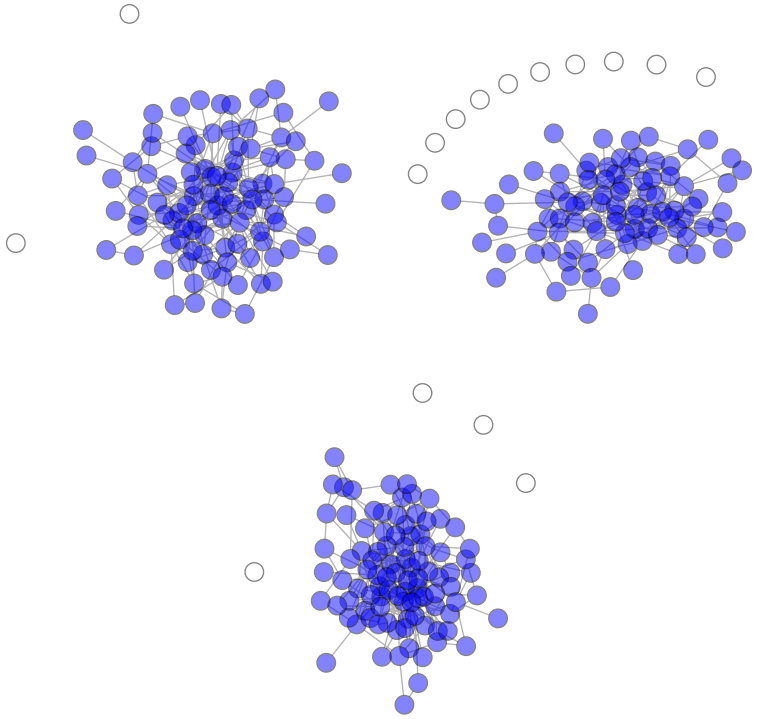


Figure C.136: Force-based layout for three different MR.5 instances.

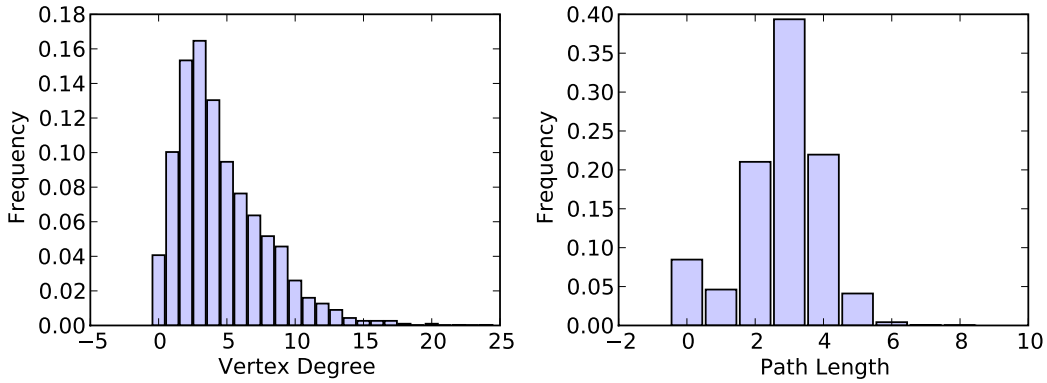


Figure C.137: Vertex degree and path length histograms for MR.5, $n = 100$.

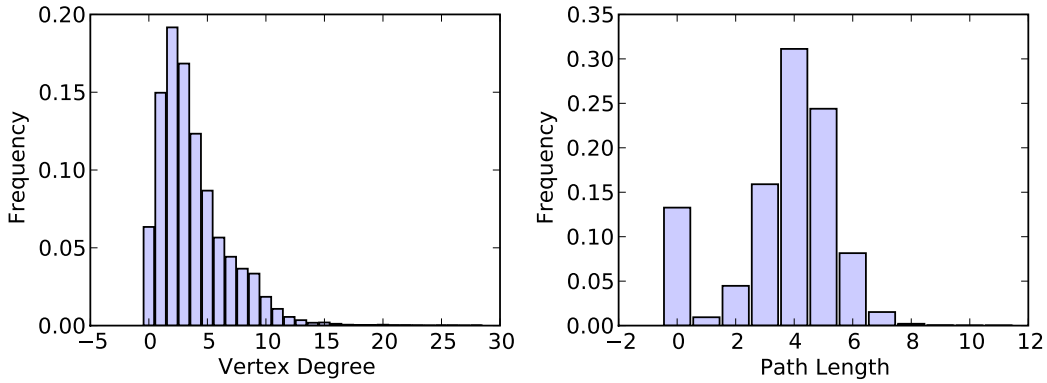


Figure C.138: Vertex degree and path length histograms for MR.5, $n = 400$.

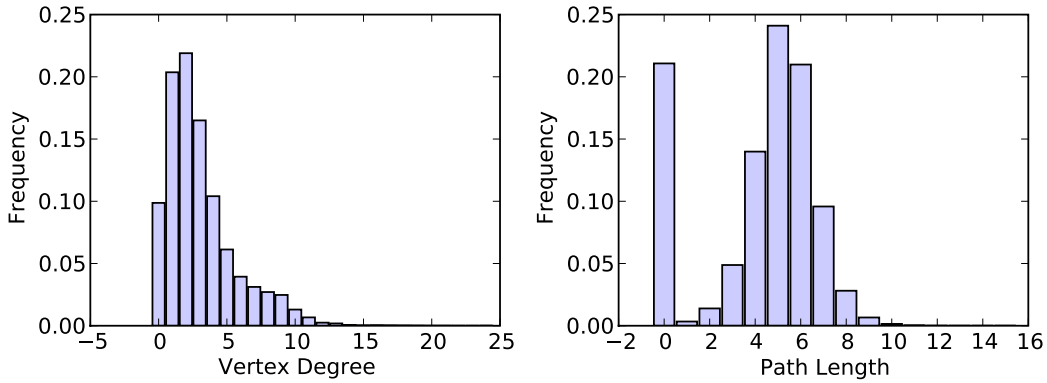


Figure C.139: Vertex degree and path length histograms for MR.5, $n = 900$.

Appendix D

CDROM Guide

As an overview, the CDROM contains the following files and folder structure. Where appropriate, folders contain an individual `readme.txt` file with additional details.

- `readme.txt` CDROM details, notes and instructions.
- `index.html` HTML file list and links to content.
- `links.html` HTML file with links to online resources.
- `thesis.pdf` Thesis document as a single PDF file.
- `esec/` The `esec` Python package and support files:
  - `esec/` Module source files.
  - `docs/` Module documentation generated from source files.
  - `test/` The `esec` module test files.
  - `testorama.bat` Windows Quick-Test Script.
  - `testorama.sh` Linux/OS X Quick-Test Script.
  - `run.py` Executes configuration and batch files.
  - `report.py` Generates batch reports using batch file details.
- `chap6/` Chapter 6 Population Investigation:
  - `cfgs/` Batch Configuration Files.
  - `reports/` Report Files (HTML and PDF).
- `chap7/` Chapter 7 Community and Ecosystem Examples:
  - `cooperate/` Cooperative Symbiosis.
  - `compete/` Competitive Predator-Prey.
  - `island/` Classic Island Model.
  - `convoluted/` Convoluted Ecosystem.

Appendix E

The `esec` Python Package

E.1 Introduction

E.1.1 Purpose

The `esec` Python package contains a number of "modules" designed to support research using an ecosystem model of evolutionary computation. `esec` has extensive support for different ecosystem models at community and population levels. It is also a robust and flexible platform for simpler traditional models of evolutionary computation.

The original scope of `esec` was to support the research objectives of this thesis, and while accomplishing this goal it became clear that other users could benefit. It is hoped that by providing this package under a liberal license others will not only find it useful, but also contribute to its ongoing development.

With this in mind, the wider group of intended users of `esec` now includes university students at both undergraduate and postgraduate levels, and other EC researchers interested in either the existing features or in using a Python based approach to algorithm development and research.

E.1.2 Features

The `esec` package was created to enable research of ecosystem models of evolutionary computation. It has the following features:

- Supports a wide range of classic EC models.
- Many standard benchmark problem, including classic real valued continuous function optimisation and binary problem landscapes.
- An abstracted and flexible model of evolution based on system levels. Evolution can be specified at different levels within the same framework, and different systems levels can be mixed and structured together.
- Evolution using complex topological structures; from localised population graphs, up to community and ecosystem structures.
- Flexible Python dictionary-based programmable configuration files. Settings are hierarchical and different configurations can be compounded (overlaid) for easy adaptation and extension of existing simulations.
- Automatic and integrated configuration syntax and value validation.
- Easily override any configuration setting from the command line.

- Flexible reporting and data logging features.
- Batch-level experiment configuration, with automatic report summaries and compression of result data files.
- Integrated module and method level documentation using Python "docstring" details.
- Operating system independent; wherever Python and the required Python packages are supported `esec` can be used.

The decision to use Python has been a positive and rewarding feature of the ESEC implementation. In contrast to building complex systems in other languages, the flexibility of Python has resulted in a clear and compact code base without large amounts of unnecessary "boiler-plate" code. Documenting code is rewarded by the integrated use of Python "docstring" features. Testing and profiling are also well supported in Python.

Python is platform independent and flexible high-level language with a clear syntax. It has excellent support for using other libraries and packages. In particular, this has been essential in enabling `esec` to take advantage of two very useful and mature packages; `numpy` and `igraph`. `numpy` provides powerful and efficient numerical calculation routines, and the `igraph` library supports nearly all of the desired topological features.

E.2 Architecture

The top level package is named `esec`, which contains submodules and an `Application` class which is the recommended way of using other package features.

Outside of the top level module are two useful Python scripts: `run.py` and `report.py`. The `run.py` script is used to run individual simulation configurations or batch experiments, and supports a number of command line options. The `report.py` script can be used to automatically create HTML and PDF reports using batch data.

The `esec` package contains several major submodules:

- `esec.ea` Contains an abstract `EA` base class for all dialects to extend. The `esec.ea.dialect` module contains concrete `EA` instances.
- `esec.landscape` A collection of problem landscapes divided into modules based on value type. The `binary` and `real` modules currently contain the majority of problem landscapes, including all of the domains described in Appendix B.
- `esec.species` Supports all things related to the management and manipulation of species and their genomes, including the tracking of statistics. A separate submodule is created for each type of species and any custom genome specific operations.
- `esec.selection` Selection methods for use in various different ESEC contexts. Divided into submodules for `uniform`, `tournament`, `truncate`, `proportional` and `special` types of selection.
- `esec.system` The isolation and abstraction of the organisational systems and processes in the ESEC model: `population`, `community` and `ecosystem`. Each system contains an internal system specific `controller`. System controllers use a command queue model where breeding operations are stored and each called in turn. System and process complexities are determined at system initialisation, so that the operation of the system is then trivial. This module also contains classes for individuals, groups of individuals and other system support.

- `esec.utils` A collection of supportive classes and functions. In particular configuration dictionary handling.
- `esec.monitor` Classes used by an `Application` instance to monitor and report on the progress of an `EA` instance.

The `Application` class is able to process a configuration dictionary, create appropriate EA object, problem landscape and monitor instances. The `Application.run()` method is then called to perform the EA operations as configured, with the progress and the result present by the monitor instance.

E.3 Dependencies

Experiments and results presented in this thesis were performed using Python version 2.5.2 on both Windows XP and Mac OS X 10.5 (Leopard) operating systems. The package has also been successfully tested with Python 2.6. Although Python 3 was available, it breaks backward compatibility with version 2 and not all dependencies were available, and so was not considered for use.

The `esec` package makes use of a number of other packages, some of which are optional. All of these projects are open source and free, typically released under BSD or GNU GPL style licenses.

- `igraph` A well respected software library written in C for creating and analysing graphs. Use extensively by `esec`. See http://igraph.sourceforge.net. Version 0.5.1 was used in this work.
- `numpy` is well known as a "fundamental" package for scientific computing with python and supports powerful N-dimensional array objects, sophisticated "broadcasting" functions and numerous mathematical and statistical routines. Although the basic use of `esec` does not require `numpy`, some landscape and the report generation features do, and so this package is considered a required dependency. See http://numpy.scipy.org/. Version 1.2.1 was used in this work.
- `psyco` (optional). A specialising compiler that, as an extension module, can greatly speed up the execution of Python code at the expense of memory usage. There are limitations: generates only 32-bit x86 code and is limited to Python 2.5 and 2.6 versions. See http://psyco.sourceforge.net/. Version 1.6 was used in this work.
- `matplotlib` (optional). A plotting library, mainly 2D, used to produce figures for batch result reports. See http://matplotlib.sourceforge.net/. Version 0.98.5.2 was used in this work.
- `nose` (optional). To quote the project website `nose` is a "unittest-based testing framework for Python that makes writing and running tests easier". Once installed `nose` provides a "nosetests" script that can discover and run tests for a project. See http://code.google.com/p/python-nose/. Version 0.10.4 was used in this work.

Most of these packages can be installed from a command line prompt using the Python "easy_install" script that is part of the `setuptools` package and included with most recent Python distributions.

E.4 Installation and Testing

E.4.1 Installation and Setup

The typical installation and usage of the current `esec` package is to simply copy the `esec` folder from the CDROM to a work location, and work within it. New configuration files should be placed within the `esec/cfgs` folder and executed from a command prompt using the `esec/run.py` script with appropriate parameters.

Although the `esec` package can be installed as part of the global site-packages available to Python, the process is not described in detail here.

Future important development work for `esec` will provide a more complete range of installation options using standard Python approaches including the creation of "egg" archive files, platform specific installers and making the package available via standard online package repositories.

E.4.2 Running Self-Tests

There are two quick and easy "high level" ways to verify a working environment for `esec` and its dependences; using a simple script to run a small number of quick example simulations, or running the suite of package tests.

A simple windows "bat" file named `testorama.bat`, and an equivalent shell script named `testorama.sh`, are located in the base `esec` directory. By executing the platform appropriate "testorama" file from the command line (or possibly "double-click" on the file using a GUI and the right OS environment settings) should launch a series of small and sample simulations, each using the `esec/run.py` script.

The second and more detailed self-test is to run the `nosetests` command in the `esec/tests` directory and observe the results. It is possible that some tests will fail as some are stochastic operations are tested using sampling techniques and these can occasionally fail. If there are errors, run tests again to assess if errors are consistent or sampling artefacts.

E.5 Basic Usage

Once installed, use the `esec/run.py` script to run a simulation. Figure E.1 shows a sample execution of the script with no parameters (default settings). The result is a simple genetic algorithm (GA) tested on a real-value problem (RVP) landscape named "Sphere", with two parameters (n2) and executing for 10 generations. We can see that the small population of 10 individuals makes some improvement during the simulation.

The `run.py` script supports a number of command line options. These include:

- `-v` VERBOSE level. 0 is lowest, 5 is highest. Particularly useful when investigating complicated configurations. The default behaviour is the refer to the configuration file which defaults to level 1.
- `-o` OPTIMISE. Use the `psyco` module (if available) to optimise execution. Default is False. Highly recommended if available.
- `-p` PROFILE. Uses the Python `cProfile` module to analyse the execution time of the script and prints a call report.
- `-c` CONFIG. A set of configuration names, as a single string joined by "+" characters. If we create a "test1.py" configuration file, and save it to the `esec/cfgs` directory, we can then use `python run.py -c test1` (no ".py" needed) to use the

```
C:\projects\python\esec>python run.py
------------------------------------------------------------
ESEC: EcoSystem Evolutionary Computation
Copyright (c) Clinton Woodward 2007-2009
------------------------------------------------------------
  ** Psyco Optimisation! **
  ** Configuration names:  RVP+Sphere+n2+GA
------------------------------------------------------------
  Simulation limits (run_stop):gen
   * Generation limit: 10
  Random number seed: 12345
  Run Count: 1
------------------------------------------------------------
  Using landscape Sphere defined on 2 parameter(s)
------------------------------------------------------------
>> EA name:GA - Genetic Algorithm
>> Type:Generational
------------------------------------------------------------
>> Population info:
 * parents: 10 and offspring: 10
------------------------------------------------------------
>> Report configuration:brief,status+best
------------------------------------------------------------
>> New Seed: 12345 + 0 (offset)

   #gen.  |  births  |  evals.  | best-fit. ###
       1         10         10   4.234470e+001
       2         20         20   4.234470e+001
       3         30         30   4.234470e+001
       4         40         40   4.234470e+001
       5         50         50   4.454935e+001
       6         60         60   4.454935e+001
       7         70         70   4.454935e+001
       8         80         80   4.481228e+001
       9         90         90   4.484959e+001
      10        100        100   5.083816e+001
>>GEN_LIMIT
------------------------------------------------------------
  stop why?  | b.date  |  evals.  |  fitness  ###
   GEN_LIMIT      100        100   5.083816e+001
------------------------------------------------------------

-->> DONE <<-- in  0.0276920043539
```

Figure E.1: Sample execution of the run.py script using the esec package.

configuration. As another example, the equivalent default sample can be specified using `-c "RVP+Sphere+n2+GA"`. Multiple configurations are loaded and overlaid on each other in the order specified.

- `-b` BATCH. The name of a single batch file.
- `-s` SETTINGS. Override any configuration settings using a quoted and semi-colon separated string. Each setting value is evaluated to support correct types. For example, the default generation limit of the default sample could be extended to 20 generations using `-s "application.gen_limit=20"`.

The typical usage of the `esec` package is to write a configuration file, save it to the `esec/cfgs` directory, and to load and execute the configuration using the `esec/run.py` script. Configuration dictionaries have been described in some detail in the thesis body, and are also detailed in the package documentation and code comments.

In particular, see the series of batch experiment files (located in the `chap6/cfg` directory) as examples of configuration and report creation details. A full presentation and discussion of the capabilities of batch and report features is beyond the scope of this document.

E.6 Documentation

The source code has been extensively commented for a wide range of audience knowledge levels. API documentation is generated directly from the Python source code using a modified version of the `epydoc` software. The documentation is available in HTML format located in the `/docs/api/` directory. Documentation includes module and class hierarchy details, and a detailed "identifier index" which supports navigation.

E.7 Design and Implementation Notes

E.7.1 Language Selection: Why Python?

The use of the Python programming language for EC research has not been widely adopted. It was selected for implementation of the ESEC model for a number of reasons, in particular because it is a nimble and expressive language, and it is good at "gluing" other useful and established software components, such as the igraph library.

For the reader not familiar with Python or its features, the following is a list of positive general features based loosely on the points of Travis Oliphant in [266].

- Clean and clear language syntax. Blocks are defined by whitespace indentation. A human reader should have the same idea of what code means as the computer, and without spurious character taking up precious screen space.
- Code in procedural or object-oriented style as needed. An OO design or language is not a *panacea* to good software design, or correct and verifiable operation.
- A liberal Open Source Initiative (OSI) approved license, supporting liberal commercial usage and no GPL-like "copyleft" restrictions.
- An interpreter that is supported on many multiple platforms. There is also a Java based interpreter Jython[1], so anywhere Java is supported, Python can be used. Similary, IronPython[2] is a .NET implementation of Python that is closely integrated with the .NET Framework but with some compatibility issues with standard CPython.

[1] See http://www.jython.org.
[2] See http://ironpython.net.

- Interactive interpreter supporting live experimentation which eliminates the compile step from the traditional write-compile-test routine of development. The IPython project[3] shell enhances this even further.
- Many ways to extend Python and let it do the important things as fast as your hardware will allow. The argument of slow Python speed does not need to apply; it's simply a matter of spending time optimising when needed.
- Easily interact with other existing pieces of software through either standard libraries or the ability to bind to many other libraries and protocols.
- The "batteries included" mentality of Python development has result in many useful built-in standard libraries, and many other freely available and easily installed libraries.
- Bindings to all standard GUI toolkits including TK, QT and wx.
- Package based distribution and installation of python modules, as well as a growing online repository that vastly simplifies package management.

Perhaps the main criticism against using Python for this type of application, evolutionary computation, is the lack of machine execution speed. Python code, being a flexible interpreted language, can run much slower than a complied language. However, it is relatively easy to identify performance issues (using built-in profiling tools) and to selectively spend effort to improve performance in the areas identified.

When the need for machine speed becomes the priority, python provides many ways to ease the transition from Python code to, for example, optimised C code extension modules. As with the optimisation of any software or process, prematurely optimising code can result in wasted effort, complex and perhaps brittle code that resists change and make it inflexible to new research requirements.

E.7.2 Software Quality

A significant amount of time, effort and thought has been applied to the testing and verification of the `esec` package. This is based on the principle that software quality is a designed objective, and that it needs to be monitored and maintained throughout the entire software lifecycle. Although tests do not prove that software is correct, without a methodology of testing, software quality is very difficult to maintain.

As already described, `esec` has used the `nose` package for testing during development. The `esec/tests` directory contains an extensive number of unit tests.

One unfortunate result of a strong test-driven development mentality is that more code is written for testing then for the actual software features. This is a situation worth avoiding if the coverage and quality of testing can be maintained. A common approach to writing unit tests is the use a unit testing framework (such as JUnit for Java or NUnit for .NET code) and to write unit tests that inherit to specify testing details. This often results in a lot of so-called "boiler-plate" code, as well as mock objects and other dependencies.

The "nose" package uses an alternative approach to only writing unit tests, and which automatically collects tests as long as they are written using some simple guidelines. Using `nose` writing and running tests is simpler and easier and ideally this enables researchers to continue with good testing practices while still adding all the new features they want to experiment and investigate with.

[3] See http://ipython.scipy.org.

Appendix F

Population Topology Experiments

The following population experiment reports are available as an electronic appendices in both PDF and HTML formats on the accompanying CDROM. A description of the report content presentation was included in Chapter 5 as part of the discussion on performance comparison.

- Topology Influence
  - Base Results: `batch01a_base`
  - Additional Problem Results: `batch01b_extended`
  - Real Genome Results: `batch00_real_species`
- Topology Scale
  - Scale-up Results: `batch02a_n400` and `batch02b_n900`
  - Scale Comparison Results: `batch02c_compare`
  - Full Graph Comparison Results: `batch00_full`
- Circular and Bound Lattices:
  - `batch03_bound`
- Influence of Order and Mate Selection
  - Lattices and Line Sequence:
    `batch04a_FLS`, `batch04b_FLSR` and `batch04c_ZigZag`
  - Lattices and FLS with Competition:
    `batch05a_FLS_c`, `batch05b_FLSR_c` and `batch05c_compare`
  - Lattices and Spiral:
    `batch06a_SpiralIn`, `batch06b_SpiralOut` and `batch06c_Compare`
  - Fitness sequence:
    `batch08a_FIT`, `batch08b_FITR` and `batch08c_compare`
- Juveniles with Delayed Competition:
  - `batch09_delayed`
- Rewired Lattices:
  - `batch10_rewire`

Appendix G

Classic Small-World Simulation

G.1 The Small-World Model

A small-world simulation was created and performed by Watts and Strogatz for their classic and often cited Nature paper in 1998 [360].

However there seems to be few recreations of the experiment available, perhaps because it is relatively simple. As a useful reference the recreation of the Watts and Strogatz small-world model, using open source software, is included here in the hope that it may assist others.

G.2 Required Software

Two specific pieces of software are required. The first is "R"[1] which is a "free software environment for statistical computing and graphics" and a very appropriate tool for replicating this particular experiment. The second piece of software is the igraph library[2], "a free software package for creating and manipulating undirected and directed graphs". igraph is a C library that also has an R package and a Python extension (used extensively for this thesis).

G.3 The Code

A large proportion of this code is simply related to presentation (plot appearance), and practically it is one of the most useful aspects of this code. (Getting a plot to look right can take some effort with any tool, and R is no exception.)

Listing G.1: Recreation of the Watts-Strogatz classic small-world ring lattice simulation using R and igraph

```
# Need the igraph library
library(igraph)

# Set up required parameters and create Watts/Strogatz lattice model
# 1998 model is n=1000, k=10 (average), normalised L(0) and C(0)
# average of 20 runs

n <- 1000
k <- 5 # == avg degree 10
reps <- 20
px = 0.52^(seq(14, 0, -1)) # from ~0.0002 to 1.0
```

[1] http://www.r-project.org
[2] http://cneurocvs.rmki.kfki.hu/igraph/

```
c_avg <- vector(mode='numeric')
l_avg <- vector(mode='numeric')

# Using the model
for (i in 1:length(px)) {
  # prep somewhere to store the results
  l_set <- vector(mode='numeric')
  c_set <- vector(mode='numeric')
  # do repeat model samples of random graph instances
  cat('p=',px[i])
  for (j in 1:reps) {
    g <- watts.strogatz.game(dim=1, size=n, nei=k, p=px[i])
    l_set <- c(l_set, average.path.length(g))
    c_set <- c(c_set, transitivity(g, type="average"))
    cat('.') # show progress
    flush.console()
  }
  # save the average of the runs
  l_avg <- c(l_avg,mean(l_set))
  c_avg <- c(c_avg,mean(c_set))

  # user feedback
  cat('\n')
  flush.console()
}
cat('\n')

# create a normalisation values and normalise
g <- watts.strogatz.game(dim=1, size=n, nei=k, p=0)
L0 <- average.path.length(g)
C0 <- transitivity(g, type="average")
l_norm <- l_avg / L0 # y1
c_norm <- c_avg / C0 # y2

options(scipen=5) # force avoid scientific notation

# Plot the normalised l (mean path length) values
plot(x=px, y=l_norm, log="x", axes=F, frame=T,
     xlim=c(0.0001,1.0), ylab=NA, xlab='p', font.lab=3,
     type="b", pch=19, col='purple')

# Plot the normalised c (clustering) values
par(new=TRUE) # treat as if a "new" device (so don't clean)
plot(x=px, y=c_norm, log="x", axes=F, frame=T,
     xlim=c(0.0001,1.0), ylab=NA, xlab=NA,
     type="b", pch=21, col='blue')

# Create big and small ticks for a log graph
bigtcks = c(0.0001, 0.001, 0.01, 0.1, 1.0)
smltcks <- vector("numeric") # the in-between points
for (i in 1:(length(bigtcks)-1)) {
  smltcks <- c(smltcks, bigtcks[i]*(2:9))
}

# Left+Right side as per defaults
axis(2, tcl=0.5, las=1) # las=0 default, see par
axis(4, tcl=0.5, labels=F)

# Bottom with big/small ticks
axis(1, at=bigtcks, tcl=0.8, labels=expression(0.0001,0.001,0.01,0.1,1.0))
axis(1, at=smltcks, tcl=0.4, labels=F)
```

```
# Top, no labels
axis(3, at=bigtcks, tcl=0.8, labels=F)
axis(3, at=smltcks, tcl=0.4, labels=F)

# paste some text within the plot at the plot coords indicated
text(0.005, 0.8, "C(p)/C(0)", cex=1.5)
text(0.0005, 0.2, "L(p)/L(0)", cex=1.5)
```

G.4 The Result

The small-world model is discussed in Section 4.5.5, and the result of this code is presented in Figure 4.13. The plot (without a detailed caption) is presented in Figure G.1.

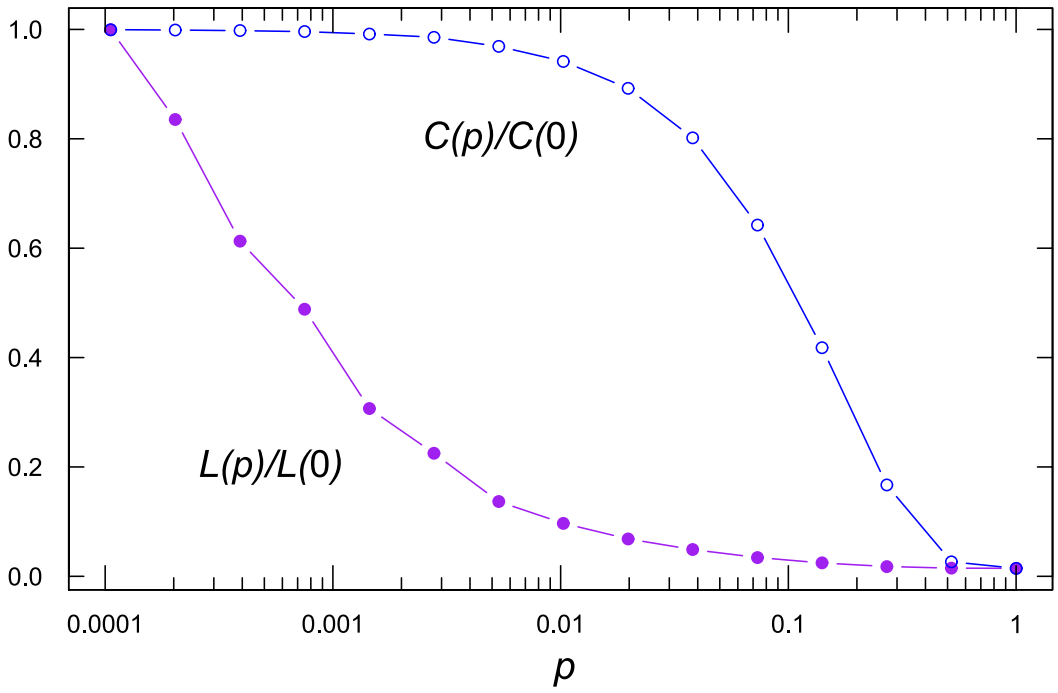


Figure G.1: The effect of p on L and C in the small-world (SW) Model. This figure is a recreation of the model and experiment data as described by Watts and Strogatz in Figure 2 of [360].