Dynamic Problems and Nature Inspired Meta-heuristics

Tim Hendtlass and Irene Moser
Centre for Information Technology Research
School of Information and Communication Technologies
Swinburne University, VIC 3001
{thendtlass, imoser}@ict.swin.edu.au

Marcus Randall
School of Information Technology
Bond University, QLD 4229
Australia
E-mail: mrandall@bond.edu.au

Abstract: Biological systems are, by their very nature, adaptive. However, the meta-heuristic search algorithms inspired by them have mainly been applied to static problems (i.e., problems that do not change while they are being solved). Recently, a greater body of work has been completed on the newer meta-heuristics, particularly ant colony optimisation, particle swarm optimisation and extremal optimisation. This survey paper examines representative works and methodologies of these techniques on this class of problems. Beyond this we outline the limitations of these methods.

Keywords: evolutionary and adaptive dynamics, ant colony optimisation, particle swarm optimisation, extremal optimisation.

1 Introduction

Many industrial optimisation problems are solved in environments that undergo continual change. These problems are referred to as *dynamic optimisation problems* and are characterised by an initial problem definition and a series of "events" that change it over time. An event defines some change either to the data of the problem or its structural definition *while the problem is being solved*. In comparison to static optimisation problems, dynamic optimisation problems often lack well defined objective functions, test data sets, criteria for comparing solutions and standard formulations [4, 15].

Evolutionary algorithms are those based on natural and biological systems. A very common example of these are genetic algorithms (GAs). For this class of algorithms, extensive modifications to accommodate dynamic optimisation problems have been made. A survey of these approaches is given by Yaochu and Branke [35]. However, for another group of evolutionary algorithms, *Ant Colony Optimisation* (ACO) [14], *Particle Swarm Optimisation* (PSO) [16] and *Extremal Optimisation* (EO) [8], suitable modifications and applications to these difficult problems are only starting to appear.

This paper presents a survey of representative ACO, PSO and EO works and methodologies for dynamic problems. Sections 2, 3, 4 describe how each of the identified meta-heuristics has been used to solve dynamic optimisation problems. Section 5 contains the comment on the limitations of the techniques.

2 Ant Colony Optimisation

ACO is a set of constructive population techniques. An overview of ACO can be found in Dorigo and Di Caro [14]. As a maturing meta-heuristic, applications to dynamic problems have started to emerge. Apart from the benchmark problem set (including the travelling salesman problem (TSP), quadratic assignment problem (QAP) and knapsack problem), industrial oriented research has been mainly in telecommunications [14] and manufacturing. We also discuss more generic frameworks for ACO and dynamic problems.

2.1 Benchmark Problems

Benchmark combinatorial optimisation problems, such as TSP, QAP and the knapsack problem, are usually presented as static problems. However, as there has been recent interest in adapting ACO to solve dynamic problems, some common problems have been changed so they have a dynamic (temporal) component. In particular, this has been done for the TSP. There are two broad ways in which a standard TSP can be transformed into a dynamic problem. These are dynamically varying the distances between cities and adding/dropping cities from the problem while it is being solved between separate runs of the ant colony solver.

Eyckelhof and Snoek [17] solve problems in which the distances between cities vary dynamically. They adapt the basic Ant System algorithm to avoid the twin problems

Proceedings of the Second IEEE International
Conference on e-Science and Grid Computing (e-Science'06)
0-7695-2734-5/06 $20.00 © 2006 IEEE

of a) certain edges becoming relatively unused because of low pheromone values and b) some edges having too much pheromone and hence dominating the decision process. This is achieved by effectively limiting the lower and upper bounds on pheromone values, implicitly implementing a form of $\mathcal{MAX} - \mathcal{MIN}$ Ant System. Using two small test data sets of 25 and 100 cities, they found that it is able to quickly adapt to the changing environment and is an effective solution method for these problems.

In contrast, Angus and Hendtlass [1] solve a problem in which cities are added or dropped rather than dynamically changing distances. The addition and removal of cities occurs in separate runs of the solver. This work showed that the ant algorithm could adapt quickly to the new problem scenario. Upon the removal or addition of a city, the pheromone levels on the problem edges were normalised.

2.2 Telecommunication Problems

Ant based algorithms have been used in the routing of telephone calls as well as to regulate network packet flow.

Schoonderwoerd, Holland, Bruten and Rothcrantz [31] use a highly abstracted ant based system for solving problems involving the routing of telephone calls between origin and destination nodes using a switched network. The networks they study are not capable of connecting all calls at a given time so the objective is to minimise the number of lost calls. In their ant based solution, ants operate independently of the calls. Each ant travels between an origin and a random destination node based on a function of pheromone, distance and node congestion. Calls are routed according the pheromone level on each neighbouring node. The link with the maximum pheromone is always chosen. This approach compares favourably with existing mobile agent methods of British Telecom [31].

Di Caro and Dorigo [13] have designed an ant colony system (ACS) to build forwarding tables for packet routing problems (such as those that occur on the Internet). The authors have shown that their system (called AntNet) compares favourably with existing routing algorithms. Zhang, Khun and Fromherz [36] have made substantial adaptations to the system to accommodate ad-hoc wireless sensor networks.

Similar work to AntNet has also been performed by White, Pagurek and Oppacher [34] except that separate ant colonies are used to determine the routes, allocate traffic to the routes and deallocate routes. Preliminary work has also been carried out on routing with fibre optic cables [33].

2.3 Industrial Manufacturing

Some of the most common problems in industrial manufacturing are job shop scheduling and machine sequencing.

Cicirello and Smith [12] solve a dynamic problem of routing jobs on a shop floor using an algorithm based on ACO principles. In their approach, each job is assigned to an ant that makes the routing decisions through the various machines on the shop floor. Pheromone is deposited on the route that each ant/job takes. This was shown to effectively balance the workload of the factory machines.

Aydin and Öztemel [2] describe a system for solving dynamic job shop sequencing problems using intelligent agents. In this paper, an agent does not solve a complete problem, but simply reacts to the requests from a simulated environment and develops an appropriate job priority list. There are two stages; a learning stage and a production stage. In the learning stage, the simulated environment gives the agent feedback about the performance which it then uses as a part of a reinforcment learning program. Arrival times and processing durations on particular machines for particular jobs are given by an exponential distribution. However, machine breakdowns and other events (such as rush orders) are not dealt with by this approach.

2.4 General Approaches

There have only been limited attempts to modify standard ACO algorithms to process dynamic problems more seemlessly.

Population ACO (P-ACO) [19, 20] is an ACO strategy that is capable of processing dynamic optimisation problems. It achieves this by using a different pheromone updating strategy. Only a set of elite solutions are used as part of the pheromone updating rules. At each iteration, one solution leaves the population and a new one (from the current iteration) enters. The candidate solution to be removed can be selected on age, quality, a probability function or a combination of these factors. The authors argue that this arrangement lends itself to dynamic optimisation, as extensive adjustments, due to the problem change, need not be made to the pheromone matrix. Instead, a solution modified to suit the new problem is used to compute the new pheromone information. This modification process works for full solutions only and is tailored for particular problems. Experimental work on the dynamic TSP and QAP showed that P-ACO could adapt to small changes in the problem better than restarting the entire algorithm.

A set of generic modifications have been proposed to allow ACO to solve dynamic optimisation problems [30]. This framework defines categories of events that change the problem definition (i.e., data and/or structure) while ACO solves it. Rather than discarding solutions and restarting the construction process, if an event occurs, the process of deconstruction begins. Deconstruction removes the components from a solution that make it infeasible to the changed problem. Results for a dynamic version of the multidimen-

Proceedings of the Second IEEE International
Conference on e-Science and Grid Computing (e-Science'06)
0-7695-2734-5/06 $20.00 © 2006 IEEE

sional knapsack problem showed that the modified ACO could quickly adapt to the problem changes. Further work on the dynamic aircraft landing problem [18] (using a real-time simulator) indicates that the approach is capable of producing good schedules in a timely fashion.

3 Particle Swarm Optimisation

The classical particle swarm optimisation (PSO) algorithm [25] was inspired by the swarming behaviour of birds and fish. While the metaphor is a dynamic one, there have been few applications to these problems. PSO tries to mimic this behaviour and apply it to a number of particles moving through problem space. This movement occurs under the influence of a number of factors, some of which can be considered personal in that no other particle is involved while others are social and involve more than one particle.

When the velocity of each particle is updated, the particle keeps a part of its velocity at the last iteration. The fraction that is kept, referred to as the momentum of the particle, prevents any drastic velocity changes and may permit the particle to pass through a local optimum. This is clearly a personal factor. A further personal influence is a tendency to return to the best position yet found by this particle ($pbest$).

Apart from the personal there are also social influences. The particle is attracted towards the best position currently experienced by other particles that form its local neighbourhood ($lbest$) and/or towards the best position found by any particle in the swarm so far ($gbest$).

Each particle updates its velocity simultaneously using some combination of personal and social influences. Momentum is always used and generally two others, at least one of which must be social. Using $pbest$ and $lbest$ encourages the parallel exploration of multiple local optima while using $gbest$ and $lbest$ encourages the whole swarm to converge on the best optimum encountered. Using $pbest$ and $gbest$ is also a viable option as experience shows that the number of particles that constitute a local neighbourhood is not critical. The interplay of the chosen influences produces an efficient search mechanism.

3.1 Adapting PSO for Dynamic Problems

There are a number of non-biological adaptations that need to be made to the classical swarm algorithm so that it suits dynamic problems. These can be summarised as: preventing the complete convergence of the swarm, keeping personal and social reference points up to date and maintaining or generating explorer particles far from any current point of convergence. Approaches that achieve at least one of these aims will be considered.

Preventing Total Convergence. Social influences between particles, attractions to $gbest$ and $lbest$, will tend to result in total convergence. To change this it is necessary to introduce some counter influence. One method [6] is to give at least some particles a charge so that, by analogy with electrostatics, two particles experience a repulsive force as they approach and the swarm would then not be able to fully converge. The particles would in time reach some (possibly dynamic) equilibrium between the convergence and divergence effects, but this does not mean that they are actively exploring. A second method [9] is to divide the swarm into sub-swarms so that not all particles converge on the same point. A particle and its closest neighbours may form a sub-swarm if the variance in the fitness of the particles is less than some threshold. Any particle that is not a member of a sub-swarm belongs to the main swarm. These sub-swarms may merge, acquire extra particles from the main swarm or collapse back into the main swarm. While developed for multi-modal functions this niching behaviour could also be used, in principle, to limit total swarm convergence. However the algorithm depends on a uniform distribution of particles in the search space, a condition that may be able to be met after initialisation but which is not met after convergence into the sub-swarms has taken place. An alternative approach to niching is described in Bird and Li [5].

Refreshing the Best Positions. If an attraction to $pbest$ is being used these best positions may be updated by allowing particles to replace their previous best position with the current position periodically [10]. Choosing a suitable period without knowledge of problem being optimised can be problematic. If an attraction to $gbest$ is being used then the fitness at this position may be periodically re-evaluated [22]. As the fitness at that point deteriorates, the probability that it will be replaced by another position as a result of the current fitness at that position increases. Again a suitable re-calculation frequency has to be chosen.

Forcing Explorer Particles. The simplest approach just requires that a number of particles be periodically moved to randomly chosen points and have their fitness re-evaluated [11]. Another approach organises particles in a tree with each particle being influenced by the particle above it (social) and itself (best position and momentum). A particle swaps with the one above it if it out performs it. This gives a dynamic neighbourhood that does require extensive calculation. This has been adapted to dynamic problems by Janson and Middendorf [23, 24]. After the value of the best-known position ($gbest$) changes (it is re-evaluated every cycle) a few sub-swarms are reinitialised while the rest are reset (have their old personal best information erased and replaced with the current position). The sub-swarms then search for the new optimum. Blackwell and Branke [7] introduce a more elaborate approach using quantum particles. Using an analogy to quantum mechanics, a particle on measurement is placed randomly within a given radius of its net current point of attraction. A uniform

Proceedings of the Second IEEE International
Conference on e-Science and Grid Computing (e-Science'06)
0-7695-2734-5/06 $20.00 © 2006 IEEE

distribution is used and a function chosen so that a finite probability exists of a movement to a distance far from the point of attraction.

Meeting all Three Requirements. The approaches described above could, if used in combination, be used to track dynamic problems. However, one further approach (WoSP) [21] has the ability to meet all three requirements simultaneously by altering the normal PSO requirement to inhibit total convergence to one that reinforces the tendency to totally converge.

The approach was originally developed to sequentially explore an arbitrarily large number of optima. An extra short-range force of attraction was added to the basic swarm equation. As a result of the discrete way in which fitness evaluations and updates to the velocity of the particles is done, an aliasing effect causes pairs of particles to approach, pass each other and then continue at very high velocities. The probability that this will happen increases the closer particle approach each other, a condition that is most likely to be met when the swarm converges. There is no longer a need to stop the particles fully converging. As the velocity with which the particles leave the swarm is variable exploration can continue both locally and at a distance. The total swarm is automatically broken into a number of sub-swarms called waves, each with its own $gbest$ value. Particles that leave a converging swarm as a result of this aliasing effect leave the wave they were in and join the highest numbered wave (creating a new one if no higher numbered wave exists). A form of evolution takes place with lower performing waves being compulsorily recruited into better performing higher numbered waves. Waves that run out of particles (owing to promotion or recruitment) die out. In this way there is a continual automatic updating of best position information available to the successive waves.

The main difference between the algorithm for static and dynamic problems is that in the former each particle keeps a tabu list of places that it has already explored and was repelled from any place on its tabu list. In this way re-exploration of any point in problem space is largely (but not totally) eliminated. For dynamic problems this tabu list can be completely removed on the grounds that any particular point in problem space may be a good optimum at several disjoined times. Alternatively extending the idea from Janson and Middendorf [24], each of these previously explored optima could be periodically re-examined and only those points whose fitness had significantly changed are removed from the tabu lists. It is not clear at this stage how the evolutionary pressure that is an important part of WoSP would respond to a dynamic problem.

4 Extremal Optimisation

The paradigm of self-organising criticality (SOC) [3] explains a wide range of natural dynamical systems. EO [8] uses elements of SOC [3] by replacing the worst individual (in this case a solution component) in a population by a random one. Over a number of iterations it is expected that the overall solution quality will increase. The original version mutated the worst component only. The absence of noise made the solver very deterministic and vulnerable to local minima. To counteract this, τ-EO was introduced. It assigns each solution component a rank k based on its current fitness within the solution and mutates it according to a probability distribution of $k^{-\tau}$.

Only a few attempts to apply EO to dynamic problems have been made. These use EO's flexible solving mechanism to adapt to underlying fluctuations automatically. EO is guided only by the component fitnesses, which are associated with the objective function. Typically EO algorithms will incorporate subtle changes automatically, as long as they are reflected in the amended objective function.

4.1 Benchmark Problems

Menai [27] investigates the use of EO on Incremental Satisfiability (ISAT) problems. Two different types of dynamics are added to static satisfiability problems from the SATLIB library: ISAT1 is obtained by starting from one clause and adding the others one by one, ISAT2 divides the same standard problems into two equal parts and solves the first half before adding the second half of the clauses.

The performance of EO is compared with a variation of the WalkSAT algorithm described in detail in McAllister, Sellman and Krautz [26]. Unfortunately, the benchmark algorithm (called R-Novelty) solves only case. This is somewhat surprising as the algorithm is very similar to EO and it is equally easy to adapt to dynamics. It is likely that the dynamics added to the problems for EO contribute to the challenge rather than making the task easier. This is corroborated by the performance of EO on a more dynamic problem (ISAT1) compared to the problem in which only one change occurs (ISAT2). EO has a better average success rate when solving ISAT2 and needs a similar number of iterations to find the solutions to ISAT2 as R-Novelty takes to solve the static problem.

An EO has been used to solve a version of the knapsack problem in which there was ten different types of changes [28, 29]. The results showed that EO will adapt quickly to changes, provided that they are reflected in the fitness function. Whenever the change incurs the inclusion of additional components, the quality of the solution will rise slowly. If, however, the change leads to a reduced capacity, very good solutions are found immediately. The

Proceedings of the Second IEEE International
Conference on e-Science and Grid Computing (e-Science'06)
0-7695-2734-5/06 $20.00 © 2006 IEEE

comparison with an ACS solver showed that EO cannot compete with its precision on smaller problem sizes, especially in a static environment. As soon as the problem size grows beyond a choice of 400 components for the next move, EO performs better. The simplicity of EO allows it to scale well. The computational effort shows only slightly more than polynomial growth when the cost limit is augmented.

4.2 A Defense Application

The Swedish Defense Research Agency investigates target recognition in which increasing numbers of moving sensors produce reports that may or may not contain information on the targets. Fast optimisation techniques are necessary to cluster the incoming reports according to the objects, before the relevant information on target objects can be processed by a tracker module. Svenson [32] compares EO and τ-EO on this task, which requires the ability to include bursts of incoming reports whenever they arrive.

The experiment therein stops short of an experiment with dynamically added reports. It only observes that EO performs better with $\tau = 1.5$ than when setting $\tau = \infty$, and that the pairwise comparison of a smaller subset of records to establish the current fitness of the report within a cluster leads to a better result than the evaluation of a larger subset of pairs of reports.

5 Limitations

The previous sections have demonstrated that all three algorithms have the capacity to solve dynamic optimisation problems. This last part of this paper considers what determines the maximum rate at which changes can occur before the algorithm performance essentially is reduced to that of random search.

For population-based algorithms that use a history of previous performance to guide future search, such as ACO and PSO, the obvious limitation comes from how fast they can learn to disregard the now irrelevant part of their history. For population based ACO this either implies a short history list or a method of detecting and removing all historic paths that are no longer valid. For non-population based ACO this will require careful handling of the evaporation to deposition ratios in the time immediately after a change has been detected. For PSO either the $gbest$ value needs to be periodically re-evaluated and a reset of it and all $lbest$ positions made as soon as $gbest$ alters or, alternatively a sequence of $gbest$ and $lbest$ (or $pbest$) values have to be used (as in WoSP) so that changes can be responded to without having to be explicitly detected.

For the single individual algorithm EO there is no explicit historical information used to guide future exploration[1]. Instead the algorithm will detect a change when the current solution suddenly becomes invalid. The current stored best solution then has to be cancelled and a special repair procedure will then have to be done on the current individual. This will take a number of changes and after this enough time must be allowed so that a good solution can be built up.

While it is possible to describe the factors that will determine the time it will take any of these algorithms to respond to a change and again have a good valid solution, the stochastic nature of the algorithms (amongst other factors) makes it impossible to quantify what this delay will be thus allowing the maximum rate of problem change to be specified. This limiting change rate is difficult to quantify. However, some limiting rate must exist, albeit it problem and algorithm dependent. Should any of these algorithms be used on a problem changing faster that the relevant limiting rate, the effect will be, that the algorithm makes too infrequent a sampling of problem space. By analogy to the aliasing effect seen when a digital data stream is sampled below the Nyquest frequency when high frequencies appear as lower frequencies, it can be predicted that the algorithms may well present solutions to problems that, in fact, have never been posed. There may be no clear indication that this has most undesirable effect has occurred.

Since the limiting change rate for the three algorithms discussed in this paper are almost certain not to be the same, the solution may be to run two or more of the algorithms in parallel, only accepting the solutions when they are at least similar.

References

[1] D. Angus and T. Hendtlass. *Ant Colony Optimisation Applied to a Dynamically Changing Problem*, volume 2358 of *Lecture Notes in Artificial Intelligence*, pages 618–627. Springer-Verlag, 2002.

[2] M. Aydin and E. Öztemel. Dynamic job shop scheduling using reinforcement learning agents. *Robotics and Autonomous Systems*, 33:169–178, 2000.

[3] P. Bak, C. Tang, and K. Wiesenfeld. Self-organized criticality: An explanation of 1/f noise. *Physical Review Letters*, 59:381–384, 1987.

[4] J. Beasley, M. Krishnamoorthy, Y. Sharaiha, and D. Abramson. The displacement problem and dynamically scheduling aircraft landings. *Journal of the Operational Research Society*, 55:54–64, 2004.

[5] S. Bird and X. Li. Adaptively choosing niching parameters in a PSO. In *Proceedings of the 8$^{\text{th}}$ Conference on Genetic and Evolutionary Computation*, pages 3–10, 2006.

[1] Because of the fitness build up and collapse cycle inherent in the behaviour of EO it may be that the currently best known solution was found some time ago it could thus be considered historic. However, since the development is unaware of this best value and always modifies the current individual there is no historic information used as on ACO and PSO.

Proceedings of the Second IEEE International
Conference on e-Science and Grid Computing (e-Science'06)
0-7695-2734-5/06 $20.00 © 2006 IEEE

[6] T. Blackwell and P. Bentley. Dynamic search with charged swarms. In *Proceedings of Genetic and Evolutionary Computation Conference*, pages 19–26, 2002.

[7] T. Blackwell and J. Branke. Multi-swarms, exclusion, and anti-convergence in dynamic environments. *IEEE Transactions on Evolutionary Computation*, 2006.

[8] S. Boettcher and A. Percus. Nature's way of optimizing. *Artificial Intelligence*, 119:275–286, 2000.

[9] R. Brits, A. Englebrecht, and F. van der Bergh. A niching particle swarm optimiser. In *Proceedings of the Asia Pacific Conference on Simulated Evolution and Learning*, pages 692–696, 2002.

[10] A. Carlisle and G. Dozire. Adapting particle swarm optimization to dynamic environments. In *Proceedings of the International Conference on Artificial Intelligence*, pages 429–434, 2000.

[11] A. Carlisle and G. Dozire. Tracking changing extrema with adaptive particle swarm optimizer. In *Proceedings of the World Automation Congress*, pages 265–270, 2002.

[12] V. Cicirello and S. Smith. Ant colony control for autonomous decentralized shop floor routing. In *Proceedings of the* 5^{th} *International Symposium on Autonomous Decentralized Systems*, pages 383–390. IEEE Computer Society Press, 2001.

[13] G. Di Caro and M. Dorigo. AntNet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research*, 9:317–365, 1998.

[14] M. Dorigo and G. Di Caro. The ant colony optimization meta-heuristic. In *New Ideas in Optimization*, pages 11–32. McGraw-Hill, London, 1999.

[15] M. Dror and W. Powell. Stochastic and dynamic models in transportation. *Operations Research*, 41:11–14, 1993.

[16] R. Eberhart and J. Kennedy. A new optimizer using particles swarm theory. In *Proceedings of the* 6^{th} *International Symposium on Micro Machine and Human Science*, pages 39–43, 1995.

[17] C. Eyckelhof and M. Snoek. *Ant Systems for a Dynamic TSP: Ants Caught in a Traffic Jam*, volume 2463 of *Lecture Notes in Computer Science*, pages 88–99. Springer-Verlag, 2002.

[18] S. Gauthier. Solving the dynamic aircraft landing problem using ant colony optimisation. Masters Thesis, School of Information Technology, Bond University, 2006.

[19] M. Guntsch and M. Middendorf. *Applying Population Based ACO to Dynamic Optimization Problems*, volume 2463 of *Lecture Notes in Computer Science*, pages 111–122. Springer-Verlag, 2002.

[20] M. Guntsch and M. Middendorf. *A Population Based Approach for ACO*, volume 2279 of *Lecture Notes in Computer Science*, pages 72–81. Springer-Verlag, 2002.

[21] T. Hendtlass. WoSP: A multi-optima particle swarm algorithm. In *Proceedings of the Congress of Evolutionary Computing*, pages 727–734. IEEE Press, 2005.

[22] X. Hu and R. Eberhart. Adaptive particle swarm optimisation: Detection and response to dynamic systems. In *Proceedings of the Congress on Evolutionary Computing*, pages 1666–1670. IEEE Press, 2002.

[23] S. Janson and M. Middendorf. A hierarchical particle swarm optimizer. In *Proceedings of the Congress on Evolutionary Computing*, pages 1666–1670. IEEE Press, 2003.

[24] S. Janson and M. Middendorf. *A Hierarchical Particle Swarm Optimizer for Dynamic Optimization Problems*, volume 3005 of *Lecture Notes in Computer Science*, pages 513–524. Springer, 2004.

[25] J. Kennedy and R. Eberhart. Particle swarm optimization. In *IEEE Conference on Neural Networks*, pages 1942–1947, 1995.

[26] D. McAllester, B. Selman, and H. Kautz. Evidence for invariants in local search. In *Proceedings of International Joint Conference on Artificial Intelligence*, 1997.

[27] M. Menai. *An Evolutionary Local Search Method for Incremental Satisfiability*, volume 3249 of *Lecture Notes in Computer Science*, pages 143–156. Springer, 2004.

[28] I. Moser and T. Hendtlass. *On the Behaviour of Extremal Optimisation when Solving Problems with Hidden Dynamics*, volume 4031 of *Lecture Notes in Artificial Intelligence*, pages 292–301. Springer, 2006.

[29] I. Moser and T. Hendtlass. Solving problems with hidden dynamics - comparison of extremal optimisation and ant colony. In *Proceedings of the IEEE World Congress on Computational Intelligence*, 2006.

[30] M. Randall. A dynamic optimisation approach for ant colony optimisation using the multidimensional knapsack problem. In *Recent Advances in Artificial Life*, volume 3 of *Advances in Natural Computation*, pages 215–226. World Scientific, New Jersey, 2005.

[31] R. Schoonderwoerd, O. Holland, J. Bruten, and L. Rothkrantz. Ant-based load balancing in telecommunications networks. *Adaptive Behavior*, 2:169–207, 1996.

[32] P. Svenson. Extremal optimization for sensor report preprocessing. In *Proceedings of Signal Processing, Sensor Fusion, and Target Recognition XIII*, pages 162–171, 2004.

[33] G. Varela and M. Sinclair. Ant colony optimisation for virtual-wavelength-path routing and wavelength allocation. In *Proceedings of the Congress on Evolutionary Computation*, 1999.

[34] T. White, B. Pagurek, and F. Oppacher. Connection management by ants: An application of mobile agents in network management. In *Proceedings of Combinatorial Optimization*, 1998.

[35] J. Yaochu and J. Branke. Evolutionary optimization in uncertain environments: A survey. *IEEE Transactions on Evolutionary Computation*, 9:303–317, 2005.

[36] Y. Zhang, L. Kuhn, and M. Fromherz. *Improvements on Ant Routing for Sensor Networks*, volume 3172 of *Lecture Notes in Computer Science*, pages 154–165. Springer, 2004.

Proceedings of the Second IEEE International
Conference on e-Science and Grid Computing (e-Science'06)
0-7695-2734-5/06 $20.00 © 2006 IEEE