

OAT: The Optimization Algorithm Toolkit

JASON BROWNLEE

Technical Report 20071220A

Complex Intelligent Systems Laboratory, Centre for Information Technology Research,
Faculty of Information and Communication Technologies, Swinburne University of Technology
Melbourne, Australia

jbrownlee@ict.swin.edu.au

or

jbrownlee AT users.sourceforge.net

I. INTRODUCTION

The *Optimization Algorithm Toolkit* (OAT) is an open source software project written in Java that provides a suite of Computational Intelligence optimization problem domains with problem instances, classical and state-of-the-art algorithms, visualisation, graphs, and much more. This work introduces the OAT including an overview of the software (Section II.), an overview of the vision and philosophy for the project (Section III.), and some suggestions for future work for the project (Section IV.).

II. THE SOFTWARE

This section considers the OAT from both a historic perspective and with regard to the three current modalities for its application: exploration, experimentation, and a platform. This work considers OAT 1.4 as of December 2007 that may be accessed via the OAT Software webpage <http://optalgtoolkit.sourceforge.net> and the OAT Project webpage <http://sourceforge.net/projects/optalgtoolkit>.

A. Problems and Algorithms

The OAT was formed in November 2006 from the integration of two software projects: *Function Optimization* and *Combinatorial Optimization* created by Jason Brownlee with contributions from Daniel Angus. Both of these previous software projects were publically released in March 2006¹. The Function Optimization software project provided a series of evolutionary algorithms and benchmark continuous function optimisation problem instances and was later adapted for participation in the Huygens Probe Optimization challenge at CEC2006² (also see [2]). The Combinatorial Optimisation software project was an adaptation of some of the framework from the Function Optimization project and predominantly provided a series of Ant Colony Optimization (ACO) algorithms (taken from [9]) applied to a small set of Travelling Salesman Problem (TSP) instances from TSPLIB³ [4]. Both software projects provided a rudimentary API for exploiting the algorithms and problem instances as well as a basic graphical interface that loosely resembled the current *OAT Explorer* interface. The principles of the parallel software development projects were to provide a framework for the easy and rapid implementation and verification of computational intelligence optimisation algorithms for specific problem domains such that the products of interest and labour (principally the algorithms) could be shared and applied with less concern with their technical implementation. Toward this end, algorithms were designed to be self-contained such that their implementation could be adapted and extended for specific application needs. The software development philosophy of code *robustness* and *understandability* was promoted over raw implementation efficiently to provide both a learning resource and promote

¹ Both projects were formally hosted on Jason Brownlee's personal website located at the URL <http://www.ict.swin.edu.au/personal/jbrownlee>

² Online: <http://ai.csse.uwa.edu.au/huygens/>

³ Online <http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/>

extensibility. These principles were carried over and elaborated in the creation and ongoing development of OAT.

B. OAT Explorer

The *OAT Explorer* interface provides an entry point for the interested novice and the research scientists alike in which algorithms and problem instances can be selected, configured, and executed. The focus of this entry point is on *informal exploratory experimentation* via run visualisation and information collection, and was inspired by the explorer interface in the WEKA machine learning workbench [5]⁴.

For the majority of the subfields of computer science, theory is preferred over experimentation given the fields strong basis in logic and mathematics [12]. In artificial intelligence, and computational intelligence in particular, experimentation is preferred over theoretical methods as the systems (strategies for adaptation and learning) are typically too complex for conventional analysis techniques. The experimental approach to computational intelligence results in many problems related to trust, reproducibility, and ultimately methodology (for example see Brownlee [6] and in particular the references in that work). Computational Intelligence algorithms are typically learning algorithms which means that even if they are implemented incorrectly or miss-configured they are still able to make progress in a given problem domain. This results in the problem of verifying whether an algorithm is implemented correctly. The focus on visualisation and information gathering regarding algorithm-problem runs in the explorer is intended to address two problems: (1) the experimental verification of algorithm implementation by assessing actual against expected behaviour and or performance, and (2) facilitating informal exploratory research via observation into new and or unfamiliar approaches. The so-called ‘playing’ with adaptive learning systems in the explorer interface is intended to promote creative thinking with regard to the design of formal experimentation and formalisation of system behaviour.

C. OAT Experimenter

The *OAT Experimenter* interface provides an entry point for interested novices and research scientists alike for the design, execution, and analysis of experimentation with computational intelligence optimisation algorithms. The focus of this entry point is the *formal exploratory experimentation* via the promotion of best practice experimental methodology, inherent repeatability, and externalisation of experiment design and execution results. Given that experimentation is a large part of the field of computational intelligence, experimental methodology is important to provide a rigours framework (see Cohen [11] for an excellent treatment of empirical methods in AI). There is a large body of literature criticising the experimental methodology in the related fields of applied optimisation, heuristics, metaheuristics, and computational intelligence (again, the reader is referred to the bibliography in Brownlee [6]). The focus of the OAT Experimenter interface is to promote best practice in experimental design, execution, and analysis. With regard to *experimental design*, a simple problem-algorithm matrix methodology is provided which is externalised to file to promote reproducibility and sharing. With regard to *experimental execution* a batch execution facility is provided. Finally, with regard to *experimental analysis* a suite of statistical hypothesis testing tools are provided to promote the rigorous reporting and interpretation of results.

D. OAT Platform

The graphical user interfaces in the OAT are intended to provide the average usage case for exploration and experimentation. In addition to user interfaces, OAT provides an extensible framework for the development and implementation of new problem domains, problem instances, and algorithms that provide solution-generating strategies to such problems (the so-

⁴ Software Online: <http://www.cs.waikato.ac.nz/ml/weka/>

called *Domain-Problem-Algorithm* pattern of organisation [7]). The focus of the *OAT Platform* is reusable solution generating strategies. The remainder of the framework exists to support the strategies, such as information gathering in the case of run probes and standards for assessment and comparison in the case of benchmark problem instances. Computational Intelligence (also called soft computing and messy artificial intelligence) optimisation strategies are the type of strategies that dominant the platform, although classical approaches are facilitated. Finally, optimization is the dominant type of problem addressed in the platform because of its ubiquity in the field and because many other problems can be represented as optimisation problems, such as regression, search, adaptation, classification, and function approximation.

III. THE VISION

Personal experience by the author has shown that there is much research in the field of Computational Intelligence that is hidden in obscurity and or obfuscation. In aggregation, this results in an unclear communication by the field on what is good and worth using (as well as the converse), and what the open and interesting problems are. A published work of an approach or problem without source code or a reproducible description has a contribution to the field of practically nothing. Those approaches that can be independently implemented are typically extremely brittle, requiring significant fixing to make them robust enough to be applicable to a broader set of problem instances. These experiences occurred during the research and implementation of the meagre set of optimization algorithms in the present version⁵. As a researcher in the field of Computational Intelligence the motivation should be to make contributions that can be understood and extended. Specifically, with regard to optimization algorithms researches must want their algorithms understood and used by colleagues and sufficient interest that their results are challenged and undependably verified.

The ultimate vision for the Optimization Algorithm Toolkit is to *promote progress in the field of Optimization with Computational Intelligence*. The OAT vision of progress is promoted through the construction and maintenance of an accessible and robust knowledgebase of computational intelligence optimization strategies, codified in the OAT. The following lists some ideals that are promoted in OAT and are expected to result in progress in the field of both Applied Optimization and Computational Intelligence:

- **Standardisation:** The best practice in computational intelligence refined through aggregate contribution from the field and codified firstly in a software platform, but ultimately in an online knowledge repository. Specifically the capture of best practice in computational intelligence optimisation algorithm implementation, experimentation, and application, and the moving away from a diverse ecosystem of typically poorly executed and communicated research.
- **Openness:** The complete availability and accessibility of information through open source implementation and online distribution. Specifically this refers to the openness of computational intelligence algorithm implementations and related problem definitions and configuration specifications, and the moving away of poorly described and obfuscated contributions to the field.
- **Repeatability:** The replication of experimentation, observation, and behaviour facilitated through openness. Specifically this refers to the ability to repeat historic, seminal, and newly published experimentation results promoting verification and extension, and moving away from hidden, lost, and obscured contributions to the field.
- **Promotion:** The visibility and acceptance of the strengths and limitations of strategies. Specifically the promotion of those thrusts of research that lead to promising results, and the demonstration of those thrusts that do not, moving away from the ambiguity fostered through lack of openness and repeatability.

⁵ As of writing: OAT 1.4 and preceding versions.

- **Participation:** The involvement of a larger and more diverse pool of contributors provided through variety entry points of varying expertise and complexity. Specifically, the participation and contribution by new and seasoned research scientists from Computational Intelligence and related fields, as well as interested novices.

The following scenarios provide grounding for the OAT vision:

Consider being able to make significant contributions to computational intelligence via exploratory and experimental research without writing a line of code (for example citizen science). Consider realising a new strategy as an algorithm implementation in a framework with pre-implemented suites of benchmark problem instances and pre-calculated comparative results on standard indicators. Consider formulating a new problem instance and having a suite of classical and state-of-the-art approaches pre-implemented. Consider a computational intelligence software project used, maintained, and refined by thousands of people including both the leaders in the field and novices alike.

Repeatability promotes the independent implementation of approaches and problems and the replication of experiments, although it is important to highlight that with effective openness (structured accessibility for example) the source code and results are already available. Thus, one may independently replicate if one desires, or may directly exploit the quantitative product of an experimentation reducing the duplication of effort. This highlights an important part of the OAT vision, inspired by Goldberg is his methodology for assessing genetic algorithms [3]. Specifically that the implementation and quantitative experimentation of a technique on a limited set of problem cases is in fact low-cost qualitative research. One may consider such research zeroth level and of limited contribution. Once implementation, assessment, and comparison are standardised and systematised (and perhaps largely automated), one may focus energies on the patchwork integration of finding towards coherent theories of adaptation, optimization, and or learning. *This important point highlights that the vision of progress in OAT shifts the focus of the field from implementation and small-scale experimentation toward broader, open, and more interesting problems.*

IV. THE FUTURE

The vision for OAT is enthusiastically grand, although there is much tangible work that may provide measurable progress. The following provides an outline of some sub-projects for important future work on the OAT:

- **Implement Everything:** There are *a lot* of published optimization algorithms, problem domains, problem instances, standard measures, and benchmark suites in the fields of computational intelligence, heuristics, operations research, mathematics, physics, and other fields. A concerted effort from a few core developers, and a broader team of implementation and test developers could see the implementation of a vast amount of optimization related research in a very short amount of time. The WEKA machine learning platform [5] and the R project⁶ are cited as machine learning and statistical methods respectively as examples of projects where this has occurred.
- **Online Repository:** An online repository for artefacts generated by the software project can be developed beyond the current *OAT Software* and *OAT Project* webpage's. This repository would allow the submission of extension, patches, and plug-ins, such that it fostered project extensibility and community. This repository would also involve a backend database for storing experimental statistics of standard approaches on benchmark problem instances with web-based reporting⁷.
- **Automated Discovery:** A large and robust suite of approaches and problem instances may be subjected to automated methods of discovery. This may include but is not limited to statistical racing of techniques (for example see [1,8,10]), statistical sensitivity analysis of

⁶ Online: <http://www.r-project.org/>

⁷ For example see Duch with artificial neural networks: <http://www.is.umk.pl/projects/datasets.html>

parameters, and the data mining of indicators from large numbers of experimental observations (many techniques on many problem instances in a database).

It is believed that the extensive implementation of domains, problems, and algorithms will promote the formation of an online repository, which will in turn promote the mining of knowledge captured in such a repository. It is expected that the culmination will be an enterprise in the field of computational intelligence based optimization that will contribute strongly suggestions at the general applicability and constraints of existing techniques (clear general contributions of the field) and at the important and open problems to be pursued in the field study.

Acknowledgements

Thankyou to Tim Hendtlass for his support. Thankyou to Daniel Angus for his ongoing feedback in the discussion on what OAT and computational intelligence is and could be, and his contributions in the formative period for the software project.

Bibliography

- [1] Bo Yuan and Marcus Gallagher, "Statistical racing techniques for improved empirical evaluation of evolutionary algorithms," *Parallel Problem Solving from Nature - PPSN VIII, 8th International Conference*, Birmingham, UK, pp. 161-171, 2004.
- [2] C. MacNish, "Benchmarking Evolutionary Algorithms: The Huygens Suite," *Late breaking paper at Genetic and Evolutionary Computation Conference (GECCO-2005)*, Washington DC, USA, 2005.
- [3] David Edward Goldberg. *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*, USA: Kluwer Academic Publishers, 2002.
- [4] G. Reinelt, TSPLIB - A Traveling Salesman Problem Library *ORSA Journal on Computing*, vol. 3, pp. 376--384, 1991.
- [5] Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools with Java implementations*, San Francisco: Morgan Kaufmann, 2000.
- [6] Jason Brownlee, "A Note on Research Methodology and Benchmarking Optimization Algorithms," Complex Intelligent Systems Laboratory (CIS), Centre for Information Technology Research (CITR), Faculty of Information and Communication Technologies (ICT), Swinburne University of Technology, Victoria, Australia, Technical Report ID: 070125, Jan 2007.
- [7] Jason Brownlee, "OAT HowTo: High-level Domain, Problem, and Algorithm Implementation," Complex Intelligent Systems Laboratory (CIS), Centre for Information Technology Research (CITR), Faculty of Information and Communication Technologies (ICT), Swinburne University of Technology, Victoria, Australia, Technical Report 20071218A, Dec 2007.
- [8] M. Birattari, P. Balaprakash, and M. Dorigo, "ACO/F-Race: Ant colony optimization and racing techniques for combinatorial optimization under uncertainty," *MIC 2005: The Sixth Metaheuristics International Conference*, Department of Business Administration, University of Vienna, Vienna, Austria, pp. 107-112, 2005.
- [9] Marco Dorigo and Thomas Stützle. *Ant Colony Optimization*, USA: The MIT Press,

2004.

- [10] Mauro Birattari, Thomas Stützle, Luis Paquete, and Klaus Varrentrapp, "A Racing Algorithm for Configuring Metaheuristics," *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 11-18, 2002.
- [11] Paul R. Cohen. *Empirical Methods for Artificial Intelligence*, Cambridge, Massachusetts, USA; London, England: The MIT Press, 1995.
- [12] W. F. Tichy , Should computer scientists experiment more? *Computer*, vol. 31, no. 5, pp. 32-40, May, 1998. IEEE Press.