Clever Algorithms: Benchmarking Algorithms*

Jason Brownlee
jasonb@CleverAlgorithms.com
The Clever Algorithms Project
http://www.CleverAlgorithms.com

December 06, 2010
Technical Report: CA-TR-20101206-1

Abstract

The Clever Algorithms project aims to describe a large number of Artificial Intelligence algorithms in a complete, consistent, and centralized manner, to improve their general accessibility. The project makes use of a standardized algorithm description template that uses well-defined topics that motivate the collection of specific and useful information about each algorithm described. This report provides an introduction to some of the problems related to benchmarking algorithms from the Clever Algorithms project.

Keywords: `Clever, Algorithms, Racing, Benchmarking`

1 Introduction

The Clever Algorithms project aims to describe a large number of algorithms from the fields of Computational Intelligence, Biologically Inspired Computation, and Metaheuristics in a complete, consistent and centralized manner [8]. The project requires all algorithms to be described using a standardized template that includes a fixed number of sections, each of which is motivated by the presentation of specific information about the technique [9].

When it comes to evaluating an optimization algorithm, every researcher has their own thoughts on the way it should be done. Unfortunately, many empirical evaluations of optimization algorithms are performed and reported without addressing basic experimental design considerations. This provides a summary of the literature on experimental design and empirical algorithm comparison methodology. This summary contains rules of thumb and the seeds of best practice when attempting to configure and compare optimization algorithms, specifically in the face of the no-free-lunch theorem.

2 Issues of Benchmarking Methodology

Empirically comparing the performance of algorithms on optimization problem instances is a staple for the fields of heuristics and biologically inspired computation, and the problems of effective comparison methodology have been discussed since the inception of these fields. Johnson is an excellent place to start suggesting that the coding of an algorithm is the easy part of the process, that the difficult work is getting meaningful and publishable results [27]. He goes on to

provide a very through list of questions to consider before racing algorithms, as well as what he describes as his 'pet peeves' within the field of empirical algorithm research.

Hooker [25] (among others) practically condemns what he refers to as competitive testing of heuristic algorithms, calling it '*fundamentally anti-intellectual*'. He goes on to strongly encouraging a rigorous methodology of what he refers to as scientific testing where the aim is to investigate algorithmic behaviors.

Barr, Golden, et al. [1] list a number of properties worthy of a heuristic method making a contribution, which can be paraphrased as; efficiency, efficacy, robustness, complexity, impact, generalizability and innovation. This is interesting given that many (perhaps a majority) of conference papers focus on solution quality alone (an aspect of efficacy). In their classical work on reporting empirical results of heuristics Barr, Golden et al. specify a loose experimental setup methodology with the following steps:

1. Define the goals of the experiment.
2. Select measure of performance and factors to explore.
3. Design and execute the experiment.
4. Analyze the data and draw conclusions.
5. Report the experimental results.

They then suggest eight guidelines for reporting results, in summary they are; reproducibility, specify all influential factors (code, computing environment, etc), be precise regarding measures, specify parameters, use statistical experimental design, compare with other methods, reduce variability of results, ensure results are comprehensive. They then go on to clarify these points with examples.

Peer, Engelbrecht et al. [35] summarize the problems of algorithm benchmarking (with a bias toward particle swarm optimization) to the following points; duplication of effort, insufficient testing, failure to test against state-of-the-art, poor choice of parameters, conflicting results, invalid statistical inference. Eiben and Jelasity [17] site four problems with the state of benchmarking evolutionary algorithms; 1) test instances are chosen ad hoc from the literature, 2) results are provided without regard to research objectives, 3) scope of generalized performance is generally too broad, 4) results are hard to reproduce. Gent and Walsh provide a summary of simple dos and don'ts for experimentally analyzing algorithms [23]. For an excellent introduction to empirical research and experimental design in artificial intelligence see Cohen [13].

The theme of the classical works on algorithm testing methodology is that there is a lack of rigor in the field. The following sections will discuss three main problem areas to consider before benchmarking, namely 1) treating algorithms as complex systems that need to be tuned before applied, 2) considerations when selecting problem instances for benchmarking, and 3) the selection of measures of performance and statistical procedures for testing experimental hypotheses. A final section 4) covers additional best practices to consider.

2.1 Selecting Algorithm Parameters

Optimization algorithms are parameterized, although in the majority of cases the effect of adjusting algorithm parameters is not fully understood. This is because unknown non-linear dependencies commonly exist between the variables resulting in the algorithm being consider a complex system. Further, one must be careful when generalizing the performance of parameters across problem instances, problem classes, and domains. Finally, given that algorithm parameters are typically a mixture of real and integer numbers exhaustively enumerating the parameter space of an algorithm is commonly intractable.

There are many solutions to this problem such as self-adaptive parameters, meta-algorithms (for searching for good parameters values), and methods of performing sensitivity analysis over parameter ranges. A good introduction to the parameterization of genetic algorithms is Lobo, Lima, et al. [30]. The best and self-evident place to start (although often ignored [17]) is to investigate the literature and see what parameters been used historically. Although not a robust solution, it may prove to be a useful starting point for further investigation. The traditional approach is to run an algorithm on a large number of test instances and generalize the results [40]. We haven't really come much further than this historical methodology other than perhaps the application of more and differing statistical methods to decrease effort and better support findings.

A promising area of study involves treating the algorithm as a complex systems, where problem instances may become yet another parameter of the model [39, 10]. From here, sensitivity analysis can be performed in conjunction with statistical methods to discover parameters that have the greatest effect [11] and perhaps generalize model behaviors.

Francois and Lavergne [21] mention the deficiencies of the traditional trial-and-error and experienced-practitioner approaches to parameter tuning, further suggesting that seeking general rules for parameterization will lead to optimization algorithms that offer neither convergent or efficient behaviors. They offer a statistical model for evolutionary algorithms that describes a functional relationship between algorithm parameters and performance. Nannen and Eiben [33, 32] propose a statistical approach called Calibration and Relevance Estimation (CRE) to estimating the relevance of parameters in a genetic algorithm. Coy, Golden, et al. [15] use a statistical steepest decent method procedure for locating good parameters for metaheuristics on many different combinatorial problem instances.

Bartz-Beielstein [4] use a statistical experimental design methodology to investigate the parameterization of the Evolutionary Strategy (ES) algorithm. A sequential statistical methodology is proposed by Bartz-Beielstein, Parsopoulos, et al. [3] for investigating the parameterization, and comparison between the Particle Swarm Optimization (PSO) algorithm, the Nelder-Mead Simplex Algorithm (direct search), and the Quasi-Newton algorithm (derivative-based). Finally, an approach that is popular within the metaheuristic and Ant Colony Optimization (ACO) community is to use automated Monte Carlo and statistical procedures for sampling discretized parameter space of algorithms on benchmark problem instances [6]. Similar racing procedures have also been applied to evolutionary algorithms [44].

2.2 Problem Instances

This section focuses on issues related to the selection of function optimization test instances, but the general theme of cautiously selecting problem instances is clearly generally applicable.

Common lists of test instances include; De Jong [29], Fogel [20], and Schwefel [41]. Yao, Lui, et al. [43] list many canonical test instances as does Schaffer, Caruana, et al. [40]. Gallagher and Yuan [22] review test function generators and propose a tunable mixture of Gaussians test problem generator. Finally, McNish [31] propose using fractal based test problem generators via a standardized web interface.

The division of test problems into classes is another axiom of modern optimization algorithm research, although the issues with this methodology are the taxonomic criterion for problem classes and on the selection of problem instances for classes.

Eiben and Jelasity [17] strongly support the division of problem instances into categories and encourage the evaluation of optimization algorithm over a large number of test instances. They suggest classes could be natural (taken from the real world), or artificial (simplified or generated). In their paper on understanding the interactions of GA parameters Deb and Agrawal [16] propose four structural properties of problems for testing genetic algorithms; multi-modality, deception, isolation, and collateral noise. Yao, Lui, et al. [43] divide their large test dataset into the categories of unimodal, 'multimodal-many local optima', and 'multimodal-few local optima'.

Whitley, Rana, et al. [42] provide a detailed study on the problems of selecting test instances for genetic algorithms. They suggest that difficult problem instances should be nonlinear, non-separable, and non-symmetric.

English [18] suggests that many functions in the field of EC are selected based on structures in the response surface (as demonstrated in the above examples), and that they inherently contain a strong Euclidean bias. The implication is that the algorithms already have some a priori knowledge about the domain built into them and that results are always reported on a restricted problem set. This is a reminder that instance are selected to demonstrate algorithmic behavior , rather than performance.

2.3 Measures and Statistical Methods

There are many ways to measure the performance of an optimization algorithm for a problem instance, although the most common involves a quality (efficacy) measure of solution(s) found (see the following for lists and discussion of common performance measures [3, 5, 26, 17, 1]). Most biologically inspired optimization algorithms have a stochastic element, typically in their random starting position(s) and in the probabilistic decisions made during sampling of the domain. Thus, the measuring of performance must be repeated a number of times to account for the stochastic variance, which also could be a measure of comparison between algorithms.

Irrespective of the measures used, sound statistical experimental design requires the specification of 1) a null hypothesis (no change), 2) alternative hypotheses (difference, directional difference), and 2) acceptance or rejection criteria for the hypothesis. The null hypothesis is commonly stated as the equality between two or more central tendencies (mean or medians) of a quality measure in a typically case of comparing stochastic-based optimization algorithms on a problem instance.

Peer, Engelbrech, et al. [35] and Birattari and Dorigo [5] provide a basic introduction (suitable for an algorithm-practitioner) into the appropriateness of various statistical tests for algorithm comparisons. For a good introduction to statistics and data analysis see Peck Olson, Devore [34], for an introduction to non-parametric methods see Holander and Wolfe [24], and for a detailed presentation of parametric and nonparametric methods and their suitability of application see Sheskin [26]. For an excellent open source software package for performing statistical analysis on data see the R Project[1].

To summarize, parametric statistical methods are used for interval and ratio data (like a real-valued performance measure), and nonparametric methods are used for ordinal, categorical and rank-based data. Interval data is typically converted to ordinal data when salient constraints of desired parametric tests (such as assumed normality of distribution) are broken such that the less powerful nonparametric tests can be used. The use of nonparametric statistical tests maybe preferred as some authors [35, 12] claim the distribution of cost values are very asymmetric and or not normal. Although it is important to remember that most parametric tests degrade gracefully.

Chiarandini, Basso, et al. [12] provide an excellent case study for using the permutation test (a nonparametric statistical method) to compare stochastic optimizers by running each algorithm once per problem instance, and multiple times per problem instance. While rigorous, their method appears quite complex and their results are difficult to interpret.

Barrett, Marathe, et al. [2] provide a rigorous example of applying the parametric test Analysis of Variance (ANOVA) of three different heuristic methods on a small sample of scenarios. Reeves and Write [38, 37] also provide an example of using ANOVA in their investigation into Epistatsis on genetic algorithms. In their tutorial on the experimental investigation of heuristic methods, Rardin and Uzsoy [36] warn against the use of statistical methods, claiming their rigidity as a problem, and the meaningfulness of practical significance over that of statistical

[1] R Project is online at `http://www.r-project.org/`.

significance . They go on in the face of their objections to provide an example of using ANOVA to analyze the results of an illustrative case study.

Finally Peer, Engelbrech, et al. [35] highlight a number of case study example papers that use statistical methods inappropriately. In their OptiBench method, algorithm results are standardized and ranked according to three criteria then compared using the Wilcoxon Rank-Sum test, a non-parametric alternative to the Student-T test that is commonly used.

2.4 Other

Another pervasive problem in the field of optimization is the reproducibility (implementation) of an algorithm. An excellent solution to this problem is making source code available by creating or collaborating with open-source software projects. This behavior may result in implementation standardization, a reduction in the duplication of effort for experimentation and repeatability, and perhaps more experimental accountability [17, 35].

Peer, Engelbrech, et al. [35] stress the need to compare to the state-of-the-art implementations rather than the historic canonical implementations to give a fair and meaningful evaluation of performance.

Another area that is often neglected is that of algorithm descriptions, particularly in regard to reproducibility. Pseudocode is often used, although (in most cases) in an inconsistent manner and almost always without reference to a recognized pseudocode standard or mathematical notation. Many examples are a mix of programming languages, English descriptions and mathematical notation, making them difficult to follow, and commonly impossible to implement in software due to incompleteness and ambiguity.

An excellent tool for comparing optimization algorithms in terms of their asymptotic behavior from the field of computation complexity is the Big-O notation [14]. In addition to clarifying aspects of the algorithm, it provides a problem independent way of characterizing an algorithms space and or time complexity.

2.5 Parting Words

It is clear that there is no silver bullet to experimental design for empirically evaluating and comparing optimization algorithms, rather there are as many methods and options as there are publications on the topic. The field of optimization as of yet has not agreed upon general method of application like the field of data mining (processes such as Knowledge Discovery in Databases (KDD) [19]). Although these processes are not experimental methods for comparing machine learning algorithms, they do provide a general model to encourage the practitioner to consider important issues before application of an approach.

Finally, it is worth pointing out a paper by De Jong [28] (somewhat controversially titled) that provides a reminder that although the genetic algorithm has been shown to solve function optimization, it is not necessarily innately a function optimizer, and rather that function optimization is a demonstration of the complex adaptive systems ability to learn. It is a reminder to be careful not to link an approach too tightly with a domain, particularly if the domain was chosen for demonstration purposes.

3 Conclusions

This report provided an introduction into some of the concerns one faces when benchmarking algorithms, specifically optimization algorithms. The content for this report was based on a prior technical report by the author on benchmarking methodology [7].

References

[1] R. Barr, B. Golden, J. Kelly, M. Rescende, and W. Stewart. Designing and reporting on computational experiments with heuristic methods. *Journal of Heuristics*, 1:9–32, 1995.

[2] Christopher L. Barrett, Achla Marathe, Madhav V. Marathe, Doug Cook, Gregory Hicks, Vance Faber, Aravind Srinivasan, Yoram J. Sussmann, and Heidi Thornquist. Statistical analysis of algorithms: A case study of market-clearing mechanisms in the power industry. *Journal of Graph Algorithms and Applications*, 7(1):3–31, 2003.

[3] T. Bartz-Beielstein, K. E. Parsopoulos, and M. N. Vrahatis. Design and analysis of optimization algorithms using computational statistics. *Applied Numerical Analysis & Computational Mathematics*, 1(2):413–433, 2004.

[4] Thomas Bartz-Beielstein. Experimental analysis of evolution strategies - overview and comprehensive introduction. Technical report, Computational Intelligence, University of Dortmund, 2003.

[5] M. Birattari and M. Dorigo. How to assess and report the performance of a stochastic algorithm on a benchmark problem: Mean or best result on a number of runs? Technical report, IRIDIA, Universite Libre de Bruxelles, Brussels, Belgium, 2005.

[6] M. Birattari, T. Stützle, L. Paquete, and K. Varrentrapp. A racing algorithm for configuring metaheuristics. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 11–18. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002.

[7] Jason Brownlee. A note on research methodology and benchmarking optimization algorithms. Technical Report 070125, Complex Intelligent Systems Laboratory (CIS), Centre for Information Technology Research (CITR), Faculty of Information and Communication Technologies (ICT), Swinburne University of Technology, 2007.

[8] Jason Brownlee. The clever algorithms project: Overview. Technical Report CA-TR-20100105-1, The Clever Algorithms Project http://www.CleverAlgorithms.com, January 2010.

[9] Jason Brownlee. A template for standardized algorithm descriptions. Technical Report CA-TR-20100107-1, The Clever Algorithms Project http://www.CleverAlgorithms.com, January 2010.

[10] F. Campolongo, A. Saltelli, and S. Tarantola. Sensitivity anaysis as an ingredient of modeling. *A Review Journal of The Institute of Mathematical Statistics.*, 15(4):377–395, 2000.

[11] K Chan, A Saltelli, and S Tarantola. Sensitivity analysis of model output: variance-based methods make the difference. In *Proceedings of the 29th conference on Winter simulation (Winter Simulation Conference)*, pages 261–268. ACM Press, New York, NY, USA, 1997.

[12] M. Chiarandini, D. Basso, and T. Stützle. Statistical methods for the comparison of stochastic optimizers. In Michel Gendreau, Peter Greistorfer, Walter J. Gutjahr, Richard F. Hartl, and Marc Reimann, editors, *MIC2005: Proceedings of the 6th Metaheuristics International Conference*, pages 189–196, 2005.

[13] Paul R. Cohen. *Emperical Methods for Artificial Intelligence*. The MIT Press, Cambridge, Massachusetts, USA; London, England, 1995.

[14] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press and McGraw-Hill, 2001.

[15] Steven P. Coy, Bruce L. Golden, George C. Runger, and Edward A. Wasil. Using experimental design to find effective parameter settings for heuristics. *Journal of Heuristics*, 7(1):77–97, 2001.

[16] K. Deb and S. Agrawal. Understanding interactions among genetic algorithm parameters. In Colin R. Reeves, editor, *Proceedings of the Fifth Workshop on Foundations of Genetic Algorithms (FOGA)*, pages 265–286. Morgan Kaufmann, 1999.

[17] A. E. Eiben and M. Jelasity. A critical note on experimental research methodology in ec. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC '02)*, volume 1, pages 582–587. IEEE Press, USA, 2002.

[18] Thomas M. English. Evaluation of evolutionary and genetic optimizers: No free lunch. In *Evolutionary Programming V: Proceedings of the Fifth Annual Conference on Evolutionary Programming*, pages 163–169. MIT Press, USA, 1996.

[19] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. The kdd process for extracting useful knowledge from volumes of data. *Communications of the ACM*, 39(11):27–34, 1996.

[20] David B. Fogel. *Evolutionary computation: toward a new philosophy of machine intelligence.* IEEE Press, 1995.

[21] O. Francois and C. Lavergne. Design of evolutionary algorithms - a statistical perspective. *IEEE Transactions on Evolutionary Computation*, 5(2):129–148, April 2001.

[22] M. Gallagher and Bo Yuan. A general-purpose tunable landscape generator. *IEEE Transactions on Evolutionary Computation*, 10(5):590–603, October 2006.

[23] I. Gent and T. Walsh. How not to do it. In *Presented at the AAAI Workshop on Experimental Evaluation of Reasoning and Search Methods*, pages –. July 1994.

[24] Myles Hollander and Douglas A. Wolfe. *Nonparametric Statistical Methods.* John Wiley & Sons, Inc., Canada, 1999.

[25] J. N. Hooker. Testing heuristics: We have it all wrong. *Journal of Heuristics*, 1(1):33–42, September 1995.

[26] Evan J. Hughes. Assessing robustness of optimisation performance for problems with expensive evaluation functions. In *IEEE Congress on Evolutionary Computation (CEC 2006)*, pages 2920–2927. IEEE Press, USA, 2006.

[27] D. S Johnson. A theoreticians guide for experimental analysis of algorithms. In D. S. Johnson and C. C. McGeoch, editors, *Proceedings of the 5th and 6th DIMACS Implementation Challenges*, pages 215–250. American Mathematical Society, 2002.

[28] Kenneth A. De Jong. Genetic algorithms are NOT function optimizers. In *Proceedings of the Second Workshop on Foundations of Genetic Algorithms*, pages 5–17. Morgan Kaufmann, 1992.

[29] Kenneth Alan De Jong. *An analysis of the behavior of a class of genetic adaptive systems.* PhD thesis, University of Michigan Ann Arbor, MI, USA, 1975.

[30] Fernando G. Lobo, Claudio F. Lima, and Zbigniew Michalewicz. *Parameter Setting in Evolutionary Algorithms.* Springer, 2007.

[31] Cara MacNish. Benchmarking evolutionary algorithms: The huygens suite. In Franz Rothlauf, editor, *Late breaking paper at Genetic and Evolutionary Computation Conference (GECCO'2005)*, Washington, D.C., USA, 25-29 June 2005.

[32] Volker Nannen and A.E. Eiben. A method for parameter calibration and relevance estimation in evolutionary algorithms. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 183–190. ACM Press, New York, NY, USA, 2006.

[33] Volker Nannen and A.E. Eiben. Relevance estimation and value calibration of evolutionary algorithm parameters. In *Joint International Conference for Artificial Intelligence (IJCAI)*, pages 975–980. 2007.

[34] Roxy Peck, Chris Olsen, and Jay Devore. *Introduction to Statistics and Data Analysis*. Duxbury PublishingS, USA, 2005.

[35] E. S. Peer, A. P. Engelbrecht, and F. van den Bergh. Cirgup optibench: a statistically sound framework for benchmarking optimisation algorithms. In *The 2003 Congress on Evolutionary Computation, (CEC '03)*, volume 4, pages 2386–2392. IEEE Press, USA, 2003.

[36] R. L. Rardin and R. Uzsoy. Experimental evaluation of heuristic optimization algorithms: A tutorial. *Journal of Heuristics*, 7(3):261–304, May 2001.

[37] Colin Reeves and Christine Wright. An experimental design perspective on genetic algorithms. In Michael D. Vose, editor, *Foundations of Genetic Algorithms 3*, pages 7–22. Morgan Kaufmann, San Francisco, CA, USA, 1995.

[38] Colin R. Reeves and Christine C. Wright. Epistasis in genetic algorithms: An experimental design perspective. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 217–224. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1995.

[39] Andrea Saltelli. Making best use of model evaluations to compute sensitivity indices. *Computer Physics Communications*, 145(2):280–297, 2002.

[40] J. David Schaffer, Richard A. Caruana, Larry J. Eshelman, and Rajarshi Das. A study of control parameters affecting online performance of genetic algorithms for function optimization. In *Proceedings of the third international conference on Genetic algorithms*, pages 51–60. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1989.

[41] Hans-Paul Schwefel. *Evolution and optimum seeking*. Wiley, New York, USA, 1995.

[42] D. Whitley, S. Rana, J. Dzubera, and K E. Mathias. Evaluating evolutionary algorithms. *Artificial Intelligence - Special volume on empirical methods*, 85(1-2):245–276, 1996.

[43] Xin Yao, Yong Liu, and Guangming Lin. Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation*, 3(2):82–102, 1999.

[44] Bo Yuan and Marcus Gallagher. Statistical racing techniques for improved empirical evaluation of evolutionary algorithms. In Edmund K. Burke, Jos Antonio Lozano, Jim Smith, Juan J. Merelo Guervs, John A. Bullinaria, Jonathan E. Rowe, Peter Tio, Ata Kabn, and Hans-Paul Schwefel, editors, *Lecture Notes in Computer Science*, pages 161–171. Springer, 2004.