A Note on Research Methodology and Benchmarking Optimization Algorithms

JASON BROWNLEE
Technical Report 070125A
Complex Intelligent Systems Laboratory, Centre for Information Technology Research,
Faculty of Information Communication Technology, Swinburne University of Technology
Melbourne, Australia
jbrownlee@ict.swin.edu.au

***Abstract*-A pervasive problem in the field of optimization algorithms is the lack of meaningful and consistent algorithm benchmarking methodology. This includes but is not limited to issues of the selection of problem instances, the selection of algorithm specifications, the algorithm configuration parameters, and interpretation of results. The intention of this paper is to summarize the literature related to benchmarking optimization algorithms, with a focus on benchmarking in the face of the "no free lunch" theorem, and useful statistical tools for interpreting results. This context for this review is biologically inspired optimization algorithms applied to continuous function optimization, although the principles extend beyond these themes.**

Keywords- No Free Lunch, Optimization Algorithm Benchmarking, Statistics, Experimental Design, Problem Classes, Sensitivity Analysis, Parameter Tuning

I. INTRODUCTION

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. Perhaps before an experimental methodology can be adopted, a researcher/practitioner may be paralyzed by the perceived pessimism of the "no free lunch" theorem that contends the futility of the benchmarking exercise. This work will provide a summary of the literature on experimental design and empirical algorithm comparison methodology. This summary will contain rules of thumb and perhaps the seeds of best practice when attempting to configure and compare optimization algorithms, specifically in the face of the no free lunch theorem.

Section II will provide a terse summary of the no free lunch theorem for search and optimization, highlighting the generalized implications and the real-world considerations related to reporting benchmark results. Section III provides a review of the literature of benchmark methodology drawn primarily from the fields of optimization heuristics and evolutionary computation. Discussed in this section are 1) a summary of common problems and concerns when experimentally evaluating and reporting on algorithm results, 2) the imperative of tuning an algorithm before benchmarking, 3) issues related to selecting benchmark problem instances, specifically from the perspective of continuous function optimization, and 4) the concerns related to selecting useful measures of performance and the use of statistical methods for testing benchmark hypotheses.

II. THE "NO FREE LUNCH" THEROEM

The so called "no free lunch" theorem (NFLT) of search and optimization[1] [11,15] is more than 10 years old and as important a contribution as it is to the field, has caused a lot of pessamisim and misunderstanding, particulary in related to the evaluation and comparison of optimization algorithms. See [40] for a precise summary of the theory, and [59,60] for a simple explanation. In simplest terms the theory indicates when searching for an extremum of a cost function, all algorithms perform the same when averaged over all possible cost functions. The implication is that the often perused general-purpose optimization algorithm is theoretically impossible.

The theory applies to stochastic and deterministic optimization algorithms, and to algorithms that learn and adjust their search strategy over time. It is invariant to the performance measure used as well as the representation selected. It indicates that random search is no better than a genetic algorithm or enumerating the search space. Further, the geometric proof implies that observed behavior of a search strategy is no indication for predicting its future behavior. Perhaps the catalyst for benchmarking cynicism [11];

"[…] comparisons reporting the performance of a particular algorithm with a particular parameter setting on a few sample problems are of limited utility".

The theorem is an important contribution to computer science, although its implications are theoretical. The original paper was produced at a time when grandiose generalizations were being made as to algorithm or configuration superiority. The practical impact of the theory is to bound claims of applicability. This advise was suggested by Wolpert and Macready; besides indicating that there is much work to do on the theory [35], they encourage effort be put into devising practical problem classes and the matching of suitable algorithms to problem classes. Further they compel the exploitation

[1] See the [36] and [58] websites for archives of NFLT publications.

of domain knowledge in optimization algorithm design [27], a axiom of modern applied optimization (and machine learning).

The theory simplifies search [26], algorithms are considered to only visit new points in the search space (no re-sampling), and the overhead of generating samples is not considered. Further the implications of the theory arise from a uniform distribution of cost functions and algorithm behaviors that assume no *a priori* knowledge of the searched cost function. The reality of applied optimization is that we do not deal with all possible cost functions, but rather subsets of problem instances, usually with similar structural aspects [46,47]. Further, algorithms are already matched to problem classes based (on the most part) there empirically observed behaviors.

In summary; 1) bound claims of algorithm or parameter suitability to the problem instances being tested, 2) research into devising problem classes and matching suitable algorithms to classes is a good thing 3) be cautious about generalizing performance to other problem instances, and 4) be very cautious about generalizing performance to other problem classes or domains.

III. 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 [12]. 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 [24] (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. [43] 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).

Barr, Golden et al. [43] in their classical work on reporting empirical results of heuristics 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, and finally 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. [17] 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 [2] 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 [22]. For an excellent introduction to empirical research and experimental design in artificial intelligence see Cohen [42].

The theme of the classical works on algorithm testing methodology [2,24,43,44] is that there is a lack of rigor in the field. This section 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 practice to consider.

A. *Selecting Algorithm Parameters*

Optimizations algorithms are parameterized, although in the majority of cases the affect 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, the NFLT warns us to be careful 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 infeasible.

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. [20]. The best and self-evident place to start (although often ignored [2]) 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 [23]. 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.

As mentioned, an area of study involves treating the algorithm as a complex systems where problem instances may become yet another parameter of the model [3,19]. From here, sensitivity analysis can be performed in conjunction with statistical methods to discover parameters that have the greatest effect [29] and perhaps generalize model behaviors.

Francois and Lavergne [41] 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 [54,55] propose a statistical approach called Calibration and Relevance Estimation (CRE) to estimating the relevance of parameters in a genetic algorithm. Coy, Golden, et al. [48] use a statistical steepest decent method procedure for locating good parameters for metaheuristics on many different combinatorial problem instances.

Bartz-Beielstein [51] 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. [49] 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 discretised parameter space of algorithms on benchmark problem instances [38] (software available [37]). Similar racing procedures have also been applied to evolutionary algorithms [4].

B.*Problem Instances*

Continuous function optimization describes a class of problem where the goal is to locate the minimum (or maximum) of a specified and typically non-differentiable objective function, sometimes called a cost function. This constrained optimization problem can be represented in mathematical notation as follows[2]:

Given $f : \Re^n \rightarrow \Re$

find $x^* \in \Re^n$ for which $f(x^*) \leq f(x),\ \forall x \in \Re^n$

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 [30], Fogel [14], and Schwefel [21]. Yao, Lui, et al. [57] list many canonical test instances as does Schaffer, Caruana, et al. [23]. Moving beyond static instances, Spears and Potter [56] maintain a list of function generators, some with source code. Gallagher and Yuan [34] review test function generators and propose a tunable mixture of Gaussians test problem generator. Finally, McNish [6] propose using fractal based test problem generators via a standardized web interface (available [7]).

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 [2] 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 [28] propose four structural properties of problems for testing genetic algorithms; multi-modality, deception, isolation, and collateral noise. Yao, Lui, et al. [57] divide their large test dataset into the categories of unimodal, multimodal-many local optima, and multimodal-few local optima. Whitley, Rana, et al. [13] 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 [52] 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 in the context of NFLT of course is that the algorithms already have some *a priori* knowledge about the domain built into them and that results are reported on an already restricted problem set. This is a reminder that instance are selected to demonstrate algorithmic behavior on a narrow domain type.

C.*Measure 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 [2,18,32,43,49]). 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[3] to account for the stochastic variance, which also could be a measure of comparison between algorithms.

Ultimately, 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. In a typically case of comparing stochastic-based optimization algorithms on a problem instance; the null

[2] Taken from [17], easily converted to maximization.

[3] Typically > 30 according to the central limit theorem such that the underlying distribution can be meaningfully summarized

hypothesis is commonly stated as the equality between two or more central tendencies (mean or medians) of a quality measure.

Peer, Engelbrech, et al. [17] and Birattari and Dorigo [32] 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 [45], for an introduction to non-parametric methods see Holander and Wolfe [39], and for an excellent and detailed presentation of parametric and nonparametric methods and their suitability of application see Sheskin [16]. 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 [17,33] 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. [33] 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. [8] 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 [9,10] 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 [44] warn against the use of statistical methods, claiming their rigidity as a problem, and the meaningfulness of 'practical significance' over that of 'statistical 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. [17] highlights 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.

D. *Other*

Another pervasive problem in the field of optimization is the reproducibility (implementation) of an algorithm[4]. 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 [2,17].

Peer, Engelbrech, et al. [17] 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 such as [25]. 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.

Finally, an excellent tool for comparing optimization algorithms in terms of their asymptotic behavior from the field of computation complexity is the Big O notation [5]. In addition to clarifying aspects of the algorithm, it provides a problem independent way of characterizing an algorithms space and or time complexity.

IV. Conclusion

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) [53] and CRISP-DM [50]). Although not experimental methods for comparing machine learning algorithms, these processes 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 [31] (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.

Acknowledgments

Tim Hendtlass for providing useful feedback on drafts of this paper.

[4] Obviously related to the large and more serious problem of reproducibility of experiments

REFERENCES

[1] The R Project for Statistical Computing [Web Page]. Accessed 2007 Jan 25. Available at: http://www.r-project.org/.

[2] A. E. Eiben and M. Jelasity, "A critical note on experimental research methodology in EC," *Proceedings of the 2002 Congress on Evolutionary Computation (CEC '02),* Honolulu, HI, USA, pp. 582-587, 2002.

[3] Andrea Saltelli, Making best use of model evaluations to compute sensitivity indices *Computer Physics Communications* , vol. 145, pp. 280-297, 2002.

[4] 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.

[5] Bruno R. Preiss. *Data structures and algorithms with object-oriented design patterns in Java*, New York, USA: John Wiley, 2000.

[6] C. MacNish, "Benchmarking Evolutionary Algorithms: The Huygens Suite," *Late breaking paper at Genetic and Evolutionary Computation Conference (GECCO-2005),* Washington DC, USA, 2005.

[7] Cara MacNish . Huygens Search and Optimization Benchmarking Suite [Web Page]. 2006; Accessed 2007 Jan 22. Available at: http://gungurru.csse.uwa.edu.au/cgi-bin/WebObjects/huygensWS.

[8] 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*, vol. 7, pp. 3-31, 2003.

[9] Colin R. Reeves and Christine C. Wright, "Epistasis in Genetic Algorithms: An Experimental Design Perspective," *Proceedings of the 6th International Conference on Genetic Algorithms,* pp. 217-224, 1995.

[10] Colin Reeves and Christine Wright, "An Experimental Design Perspective on Genetic Algorithms," *Foundations of Genetic Algorithms 3,* pp. 7-22, 1995.

[11] D. H. Wolpert and W. G. Macready, No Free Lunch Theorems for Optimization *IEEE Transactions on Evolutionary Computation*, vol. 1, pp. 67-82, 1997.

[12] D. S Johnson, "A Theoreticians guide for experimental analysis of algorithms," *Proceedings of the 5th and 6th DIMACS Implementation Challenges,* pp. 215-250, 2002.

[13] Darrell Whitley, Soraya Rana, John Dzubera, and Keith E. Mathias, Evaluating evolutionary algorithms *Artificial Intelligence - Special volume on empirical methods*, vol. 85, pp. 245-276, 1996.

[14] David B. Fogel. *Evolutionary Computation : Toward a New Philosophy of Machine Intelligence*, New Jersey, USA : John Wiley & Sons, Inc. 1995.

[15] David H. Wolpert and William G. Macready, "No Free Lunch Theorems for Search," Santa Fe Institute, Sante Fe, NM, USA, Technical Report SFI-TR-95-02-010, 1995.

[16] David J. Sheskin. *Handbook of Parametric and Nonparametric Statistical Procedures*, USA: Chapman & Hall/CRC, 2000.

[17] E. S. Peer, A. P. Engelbrecht, and F. van den Bergh, "CIRG@UP OptiBench: a statistically sound framework for benchmarking optimisation algorithms," *The 2003 Congress on Evolutionary Computation, (CEC '03),* pp. 2386-2392, 2003.

[18] Evan J. Hughes, "Assessing Robustness of Optimisation Performance for Problems With Expensive Evaluation Functions," *IEEE Congress on Evolutionary Computation (CEC 2006),* Canada, pp. 2920-2927, 2006.

[19] F. Campolongo, A. Saltelli, and S. Tarantola, Sensitivity Anaysis as an Ingredient of Modeling *A Review Journal of The Institute of Mathematical Statistics.*, vol. 15, pp. 377–3952000.

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

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

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

[23] 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," *Proceedings of the third international conference on Genetic algorithms,* George Mason University, United States, pp. 51-60, 1989.

[24] J. N. Hooker, Testing heuristics: We have it all wrong *Journal of Heuristics*, vol. 1, pp. 33-42, Sep, 1995.

[25] John Dalbey. Pseudocode Standard [Web Page]. Accessed 2007 Jan 22. Available at: http://www.csc.calpoly.edu/~jdalbey/SWE/pdl_std.html.

[26] John R. Woodward and James R. Neil, "No Free lunch, program induction and combinatorial problems," *EuroGP : European conference on genetic programming,* Essex, UK, pp. 475-484, 2003.

[27] Joseph C. Culberson, "On the Futility of Blind Search," University of Alberta, Edmonton, Alberta, Canada, Technical Report TR96-18, 1996.

[28] Kalyanmoy Deb and Samir Agrawal, "Understanding Interactions among Genetic Algorithm Parameters," *Proceedings of the Fifth Workshop on Foundations of Genetic Algorithms (FOGA),* Madison, WI, USA, pp. 265-286, 1999.

[29] Karen Chan, Andrea Saltelli, and Stefano Tarantola, "Sensitivity analysis of model output: variance-based methods make the difference," *Proceedings of the 29th conference on Winter simulation (Winter Simulation Conference),* Atlanta, Georgia, United States, pp. 261-268, 1997.

[30] Kenneth A. De Jong, An Analysis of the Behavior of a Class of Genetic Adaptive Systems 1975. University of Michigan.

[31] Kenneth A. De Jong, "Genetic Algorithms are NOT Function Optimizers," *Proceedings of the Second Workshop on Foundations of Genetic Algorithms (FOGA),* Vail, Colorado, USA, pp. 5-17, 1992.

[32] 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?," IRIDIA, Université Libre de Bruxelles, Brussels, Belgium, TR/IRIDIA/2005-007, 2005.

[33] M. Chiarandini, D. Basso, and T. Stützle, "Statistical methods for the comparison of stochastic optimizers," *MIC2005: Proceedings of the 6th Metaheuristics International Conference,* Vienna, Austria, pp. 189-196, 2005.

[34] M. Gallagher and Bo Yuan, A general-purpose tunable landscape generator *IEEE Transactions on Evolutionary Computation*, vol. 10, pp. 590-603, Oct, 2006.

[35] M. Koppen, D. H. Wolpert, and W. G. Macready, Remarks on a recent paper on the "no free lunch" theorems *IEEE Transactions on Evolutionary Computation*, vol. 5, pp. 295-296, Jun, 2001.

[36] Martin Sewell, Web Master. No Free Lunch Theorems [Web Page]. Accessed 2007 Jan 22. Available at: http://www.no-free-lunch.org/.

[37] Mauro Birattari. race: Racing methods for the selection of the best [Web Page]. Accessed 2007 Jan 22. Available at: http://cran.r-project.org/src/contrib/Descriptions/race.html.

[38] 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.

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

[40] Nicholas J. Radcliffe and Patrick D. Surry, Fundamental Limitations on Search Algorithms: Evolutionary Computing in Perspective *Computer Science Today: Recent Trends and Developments. Lecture Notes in Computer Science*, vol. 1000, pp. 275-291, 1995.

[41] O. Francois and C. Lavergne , Design of evolutionary algorithms - A statistical perspective *IEEE Transactions on Evolutionary Computation*, vol. 5, pp. 129-148, Apr, 2001.

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

[43] R. Barr, B. Golden, J. Kelly, M. Rescende, and W. Stewart, Designing and Reporting on Computational Experiments with Heuristic Methods *Journal of Heuristics*, vol. 1, pp. 9-32, 1995.

[44] R. L. Rardin and R. Uzsoy, Experimental Evaluation of Heuristic Optimization Algorithms: A Tutorial *Journal of Heuristics*, vol. 7, pp. 261-304, May, 2001.

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

[46] Stefan Droste, Thomas Jansen, and Ingo Wegener, Optimization with Randomized Search Heuristics - The (A)NFL Theorem, Realistic Scenarios, and Difficult Functions *Theoretical Computer Science*, vol. 287 , pp. 131-144, Sep, 2002.

[47] Stefan Droste Thomas Jansen and Ingo Wegener, "Perhaps Not a

Free Lunch But At Least a Free Appetizer," *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '99),* pp. 833-839, 1999.

[48] 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*, vol. 7, pp. 77-97, 2001.

[49] T. Bartz-Beielstein, K. E. Parsopoulos, and M. N. Vrahatis, Design and Analysis of Optimization Algorithms Using Computational Statistics *Applied Numerical Analysis & Computational Mathematics*, vol. 1, pp. 413-433, 2004.

[50] The CRISP-DM consortium. CRoss Industry Standard Process for Data Mining [Web Page]. Accessed 2007 Jan 22. Available at: http://www.crisp-dm.org/.

[51] Thomas Bartz-Beielstein, "Experimental Analysis of Evolution Strategies - Overview and Comprehensive Introduction," Computational Intelligence. University of Dortmund, Technical Report 157, Nov 2003.

[52] Thomas M. English, "Evaluation of Evolutionary and Genetic Optimizers: No Free Lunch," *Evolutionary Programming V: Proceedings of the Fifth Annual Conference on Evolutionary Programming,* San Diego, CA, USA, pp. 163-169, 1996.

[53] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth, The KDD process for extracting useful knowledge from volumes of data *Communications of the ACM*, vol. 39, pp. 27-34, 1996.

[54] Volker Nannen and A.E. Eiben, "Relevance Estimation and Value Calibration of Evolutionary Algorithm Parameters," *Joint International Conference for Artificial Intelligence (IJCAI),* pp. 975-980.

[55] Volker Nannen and A.E. Eiben, "A method for parameter calibration and relevance estimation in evolutionary algorithms," *Proceedings of the 8th annual conference on Genetic and evolutionary computation,* Seattle, Washington, USA, pp. 183-190 , 2006.

[56] William M. Spears and Mitchell A. Potter . Genetic Algorithms (Evolutionary Algorithms): Repository of Test Problem Generators. 99. 2007.

[57] Xin Yao, Yong Liu, and Guangming Lin, Evolutionary programming made faster *IEEE Transactions on Evolutionary Computation*, vol. 3, pp. 82-102, Jul, 1999.

[58] Yin-Yang. Yin-Yang: No-Free-Lunch Theorems for Search . 99. 2007.

[59] Yu-Chi Ho and D.L. Pepyne, "Simple explanation of the no free lunch theorem of optimization," *Proceedings of the 40th IEEE Conference on Decision and Control,* Orlando, FL, USA, pp. 4409-4414, 2001.

[60] Yu-Chi Ho and D.L. Pepyne, Simple explanation of the no free lunch theorem of optimization *Cybernetics and Systems Analysis*, vol. 38, pp. 292-298, 2002.