Guidelines for Designing and Reporting on Computational Experiments with Heuristic Methods

Richard S. Barr[*] Bruce L. Golden[†] James Kelly[‡]
William R. Stewart[§] Mauricio G.C. Resende[¶]

March 16, 2001

Abstract

This report studies the design of computational experiments to test heuristic methods and provides guidelines for reporting on such experimentation....

[*]Department of Computer Science and Engineering, Southern Methodist University, Dallas, TX 75275, barr@seas.smu.edu

[†]College of Business, University of Maryland, College Park, MD, bgolden@umdacc.umd.edu

[‡]School of Business, University of Colorado at Boulder, Boulder, CO, james.kelly@colorado.edu

[§]School of Business Administration, The College of William and Mary, Williamsburg, VA, wrstew@mail.wm.edu

[¶]Mathematical Sciences Research Center, ATT Bell Laboratories, Murray Hill, NJ 07974-2040, mgcr@research.att.com

The effectiveness of any proposed methodology for solving a given problems can be demonstrated by theoretical analysis and by empirical testing. This report focuses on the issues involved in designing computational experiments to test heuristic methods and gives guidelines for reporting on the experimentation.

We follow in the footsteps of those who have championed high-quality reporting on computational experiments with mathematical programming software. These efforts began in the late 1970s with [12, 8, 5], with additional writing on the subject appearing more recently in [11, 9, 2, 3].

Reporting on experimentation with heuristic methods involves many of the same concerns as with optimization algorithms, but has its own distinctive issues. While some elements of this topic have been explored elsewhere[15, 14, 19, 3], this paper takes a comprehensive view of the issues involved and provides guidelines for researchers in this important and expanding area.

1 The Role of Heuristic Methods in Problem-Solving

A *heuristic method* is a well-defined set of steps for quickly identifying a high quality solution for a given problem, where a *solution* is a set of values for the problem unknowns and "quality" is defined by a stated evaluation metric or criterion. Heuristic methods are also called *approximation algorithms*, *inexact procedures*, *incorrect algorithms*, and simply *heuristics*.

One category of heuristic methods is associated with problems for which an optimal, correct, or exact solution exists that may be computed by an *optimization*, *correct*, or *exact algorithm*. Heuristic methods are often used to identify "good," approximate solutions to such problems in less time than is required for an exact algorithm (if one exists) to uncover an exact solution.

Hence the role of heuristic methods is to identify problem solutions where no exact algorithm exists, or where time is more important than solution quality (or the knowledge of quality). A superior heuristic method is one that quickly identifies solutions, identifies high-quality solutions, and is robust, performing well across a wide range of of problems and tuning-parameter settings (and is simple to implement?). To claim superiority over other approaches, a given method should excel on one or more of these dimensions.

Superior performance of a heuristic method can be demonstrated by a theoretical analysis and, indirectly, through empirical testing. Testing is accomplished by implementing the methodology as a computer program and experimenting with the program. The sections that follow discuss how to design and execute these computational experiments, how performance can be measured, and how best to report the experimental results.

2 Computational Experiments with Heuristics

An *experiment* is "a test under controlled conditions that is made to demonstrate a known truth, examine the validity of a hypothesis, or determine the efficacy of something previously untried.[1]" "Experiments are carried out by investigators in all fields of study to discover something about a particular process or to compare the effect of several factors on some phenomena.[18]"

In computational testing of algorithms, including heuristic methods, a typical experiment consists of solving a given problem instance using a computer implementation of the algorithm. Usually the objective of the experiment is to demonstrate the efficiency or effectiveness of the software (and, by implication, the underlying methodology) in this particular instance. By running a series of such experiments and comparing the results with those obtained using other approaches, the relative effectiveness of the software on the problem set (and, implicitly, other problems) is established.

The experimenter has great latitude in selecting the problem, implementing the algorithm, choosing a computing environment, selecting performance measures, and reporting the results, and each choice made can have a significant effect on the outcome of the experiment. Therefore, to ensure that the reported information is meaningful, the researcher should use a carefully constructed experimental design and document all factors that could influence the results.

2.1 Experimental Designs (under construction)

What is meant by an experimental design? (Define) While all testing involves a design, some designs are more illuminating than others.

What are the goals of experimentation and experimental designs? (demonstrate some truth, validate a hypothesis, examine the performance of something) Relate to our situation.

In the physical sciences, engineering, and medicine, approved experimental practices are highly developed and well-defined. Standards for empirical testing in the computing and mathematical sciences remain much less rigorous, and there is a broad range of accepted practice. Despite regular exhortations for high standards of reporting, demonstration or proof-of-concept studies with minimal or ad hoc testing is more the norm than carefully constructed experimental designs with statistically validated conclusions.

What constitutes a good experiment and experimental design? Good experiment: (define) Good design: unbiased, achieves experimentation goals, clearly demonstrates the performance of the tested process, uncovers the reasons for performance, has a justifiable rationale, generates supportable conclusions, is repeatable.

Best: statistical experimental design methods[18, 16]. See [14, 19, 2] for examples of their use in testing optimization and heuristic algorithms. Not always an option when problem factors cannot be controlled by the experimenter.

Helpful: results on standard benchmark problems and problems of practical interest.

We encourage greater rigor and the use of statistical design principles wherever possible.

2.2 Measuring Performance

The measurement of performance is a critical issue when reporting computational results obtained through the use of a heuristic method. The literature contains numerous attempts to demonstrate the quality of heuristic performance. Essentially, most researchers and practitioners are interested in answering the following questions:

1. What is the best solution found?
2. How long does it take to determine the best solution?
3. How quickly does the algorithm find good solutions?

4. How robust is the method?

The first two questions can be directly answered. Readers may also be interested in the total time that the heuristic is run as well as the setup time needed to begin a run. The third question can be conveniently addressed by a graph of performance as a function of time. The shape of this graph clearly demonstrates the performance of a heuristic and is useful to practitioners as well as researchers.

The question of robustness is more difficult to address. Clearly, a heuristic that can only obtain an excellent solution for one instance of a problem is not robust and arguably not very interesting. Generally, robustness is based on the ability of a heuristic to perform well over a wide range of test problems. Furthermore, heuristic strategies and parameters should either remain constant over the set of test problems or should be set based on individual test problem attributes. Robustness should be demonstrated prior to fine-tuning a heuristic for a single problem instance.

Finally, authors should be encouraged to report negative results. For example, if a heuristic performs well over a wide range of problems but fails on a specific type, then the authors should report all of the results.

2.3 Test Problems

Clearly, the set of test problems used to evaluate a heuristic can affect performance results. If possible, benchmark problems taken from the literature should be used. This allows for a direct comparison with other heuristics. If problems are generated by the developer, then the generation process should be clearly described and the newly generated problems should be archived so that other researchers can use them in the future. Generated problem instances should be representative of problems likely to found in the field. Problem instances generated from real data are preferred and should be included in the analysis whenever possible. In general, the more test problems evaluated the better the analysis. Large numbers of results are best displayed graphically.

Heuristics typically contain multiple strategies and multiple parameters that control these strategies. Readers are not only interested in the final results but are also interested in the relative contributions of the various strategies. Computational testing should be included to clearly demonstrate the contribution of each strategy within a complex heuristic. This approach identifies innovative ideas that may be used in other problems. Actual parameter settings should be clearly defined. Furthermore, the process used to determine these settings should also be described. Parameter sensitivity analyses are useful in evaluating the robustness of a heuristic.

2.4 Comparing Heuristic Methods

It is desirable and often necessary to compare a heuristic to other approaches. If well-known or published techniques exist, then the new heuristic should be compared against them. It is preferable to obtain the competing method software and make comparisons on the same platform. If other methods do not exist, then a general method, such as one based linear or integer programming, should be developed for comparisons. The comparisons should be made based on the analyses described above. The performance / time graph is an excellent way to compare different heuristics. Use statistics See upcoming JOC article by McGeogh "Statistics in the Analysis of Algorithms"

3 Reporting on Computational Experiments with Heuristic Methods

Heuristics or approximate algorithms fall into two categories, finite algorithms that produce a feasible (usually suboptimal) solution to a problem in number of iterations that is proportionate to the size of the problem and unbounded algorithms that essentially search for better and better solutions until that reach some arbitrary stopping point. Greedy algorithms that stop when these are no further improvements possible are examples of the former, and metaheuristics, such as simulated annealing, tabu search and genetic algorithms, heuristics, which drive local search algorithms are examples of the latter. The reporting requirements for these two categories of heuristics differ in several ways. In particular, most metaheuristic techniques require one or more parameters be set tune the heuristic and tell it when to stop. With a finite heuristic, the search stops occurred or when no further improvements can be made. In the material that follows, reporting guidelines will be discussed for heuristic algorithms that fall in each of these categories.

3.1 What should be reported?

In an ideal world, competing algorithms would be coded by the same expert programmer and run on the same test problems on the identical computer configuration. The results of these runs in terms of time to solution and quality of the solution produced on each problem instance would be directly comparable for the two heuristics. Absent this ideal situation, the conscientious researcher should do what is possible to help the reader understand the results and enable that reader to do the comparisons.

Fundamental to this end is that all computational results be accompanied with detail of the machine configuration on which the heuristic being tested was run (e.g. brand, model, size of memory, CPU speed, and operating system). In addition, the versions of relevant software such as compilers and operating systems along with the settings used (e.g. optimize should be reported. With this information in hand, the reader can hopefully draw some conclusions regarding the speed of the algorithm being distributed. In terms of problems solved, all problem instances in the open literature should be treated and the results included, even if some of these are failure of the heuristic on particular problems.

Besides the differentiation as to whether the heuristic procedure is a finite or potentially infinite, there are three reasons why a new heuristic could be of interest to the research community, it produces higher quality solutions (i.e. solutions closer to the optimal solution) than its competitors, it produces high quality solutions faster than its competitors, or it introduces a new and innovative heuristic technique that is creative in its own right and may in time lead to better and faster heuristics. Whichever the accomplishment of the heuristic being reported, some computational experimentation will be necessary to demonstrate that procedure is effective in doing what the author claims it will do.

For all heuristics whether they be finite or infinite, or whether the author is preparing a new way of looking at problems, a number of standard practices should be adhered to in presenting the results.

3.2 Reproducibility

The heuristic should be described in sufficient detail so that it could be replicated by another researcher and the author's results.

3.3 Environment

The machine environment and versions of software used in computational tests should be fully reported.

3.4 Test problems

For many standard problems (i.e. traveling salesman, bin packing, etc.) there exists a bank of well-studied test problems that are available in the literature and generally in electronic form (e.g. TSPLIG ref [ ]). A new heuristic should be tested on all available problems for which it was designed. Where an author generates his/her own test problems, some effort should be given to the description of how these problems were generated, and whether they are generally representative of this class of problem. In some cases, problem characteristics make problems easy to solve by just about any heuristic. For this reason, where test problems have been generated to test a new heuristic, some effort must be given to demonstrating that these test problems are indeed representative of the class of problem for which the heuristic was designed.

3.5 Quality of solutions

When testing an algorithm that finds an optimal solution to a given problem, the important issues are speed and rate of convergence to the optimal solution. For heuristics, the additional consideration of how close the heuristic solution comes to the optimal is generally the primary concern of the researcher. Where possible, when optimal solutions to a problem exist, the heuristic solutions obtained should be compared to those optimal solutions. Generally the percent deviation from optimal is reported. When the optimal solution is unavailable, the percent deviation from a tight lower (upper) bound can be an effective measure of the quality of solution (see [13]). However, a gap of 20-30% between the bound and heuristic solution is probably not tight enough to convey useful information.

For most standard problems heuristic results exist in the open literature and direct comparison of a heuristic algorithm's performance to earlier heuristic solutions should be made in much the same way as comparisons are made to optimal solutions.

3.6 Timing

In most cases, absent the ideal environment described at the beginning of this section, timing comparisons across platforms and programmers are imprecise at best. Where the complexity of the heuristic can be analyzed in terms of the order (0 [ ]) operator, this should be done to give the reader an asymptotic estimate of the computational efficiency of the heuristic. However, for practical purposes most researchers must in the end run their heuristics on standard test problems and report the results in terms of quality of solution and time to solution. In terms of the timings, no guidelines can hope to anticipate all the possible forms that computational testing can take on, nor should they presume to set impossible standards that must be adhered to an all occasions. Rather, what follows are a set of standards that should reasonably be followed in reporting the timing of heuristic testing.

1. Time to solution. The time it takes the heuristic to find and report the solution the author is using in assessing the quality of the solution produced by that heuristic. This timing should include all processing involved (e.g. computation of distance matrices, etc.) along with an preprocessing. Where the heuristic is composite (i.e. initial solution, improved solution) the timing of each phase and the quality of solution at the end of each phase should also be reported. For open ended heuristics such as tabu search and simulated annealing, the time for the total run, not just the time to the best solution, should also be reported.

2. Empirical growth function. While the complexity operator can give some idea of how solution times might grow with problem size, an empirical estimate of this growth can be quite useful in assessing how a heuristic will perform on large instances of a problem.

3. Benchmark testing. Where possible, some idea of the speed of the machine and software being used should be given the reader, if only the execution of some standard algorithm, such as a greedy heuristic, for the problem at hand should help in assessing the power of the machine used.

3.7 Experimental Design

When comparing the results of one heuristic to another, it is incumbent on the author to demonstrate that his/her approach had made a contribution. When the area of contribution is superior quality of solution or faster time to solution, some thought should be given to a statistical comparison of the author's results to established results from the literature. Many non parametric tests exist that can help in this endeavor (sign test, runs test, Wilcoxon test).

Generally a paper with statistically significant results will be superior to one where the reader is left to compare pairs of numbers from two columns in a table. The empirical growth function mentioned above is an example of a statistically fit curve that answers an empirical question for the reader.

3.8 Metaheuristics

Perhaps the most ambiguous arena for reporting on the performance of a metaheuristic algorithm that employs some version of a local search heuristic and drives beyond a single solution to explore many local optimal solutions. Metaheuristics like simulated annealing and tabu search are examples of heuristics that are essentially unbounded in the time they can use to solve a problem. With these heuristics in particular, special care must be given to reporting the results. In particular, most metaheuristics involve parameters who values must be properly set for the heuristic to operate effectively. Since the performance of most metaheuristics is tied to the parameter settings, much of the research effort can be spent in determining the appropriate values for these settings.

In reporting heuristic results for metaheuristics, the following points should be addressed.

1. Values of any parameters employed in by the heuristic should be reported. Where these values are problem dependents, the rules for establishing appropriate values should be specified. In general, if there should be a reliable way of setting effective parameter values for a new instance of the problem the heuristic is designed to solve.

2. The process employed to determine appropriate parameter settings should be reported. This may involve some sampling and statistical

analysis. Heuristics which perform well over a range of parameter values (robust) are generally superior to heuristics which require unique settings for every problem instance.

3. Computational Results. Given that most metaheuristics can be run for an indefinite period of time, special care must be given to reporting the time to the best solution. In particular, the time required to produce the reported solution and the total length of the run that produced that solution should be reported. All reported solutions should be for either a single set of parameter values or a specific rule should be used to establish the parameter values for each instance. When the best solution is reported from a set of runs with differing parameter values, this should be clearly reported.

4. Robustness. Where parameter values are chosen, some measure of the robustness of the heuristic performance to small changes in parameter settings should be offered for the readers (see [21]).

5. Speed of convergence. While heuristics that produce superior solutions are important, the speed with which these heuristics converge to a solution close to that "best found" solution needs to be reported. A graphical display of the quality of solution versus the time the algorithm such as shown in [21] would be helpful in this respect. Additionally some measure of how fast the heuristic converges to such a good solution is needed.

   As an example of such a measure, report the ratio of time to produce a solution within 5 percent of the best found to the time to produce that "best found" solution.

$$r_{0.05} = \frac{time\ to\ within\ 5\%\ of\ best}{time\ to\ best\ found}$$

4 Guidelines for Testing and Reporting on Heuristics

We propose the following guidelines for researchers to use in reporting on computational experiments with heuristic methods (based on [3]).

- Thoroughly document the process.
  - Describe the code being tested. This includes the heuristic or algorithm on which it is based, including any modifications; the overall design; the data structures used; and the available tuning parameters.
  - Document the computing environment for the experimentation. Report all pertinent characteristics of the machine used, including the manufacturer, model, types and number of processors, interprocessor communication schemes, size of memories, and configuration.
  - Describe the testing environment and methodology. State how times were measured. Report all values of tuning parameters.
- Use a well-considered experimental design
  - Focus on the real time and cost required to solve difficult problems.
  - Try to identify those factors that contribute to the results presented, and their effects. This includes the impacts of problem characteristics and tuning-parameter strategy.
  - Provide points of reference. If possible, use well-known codes and problems to determine reference values, even if testing must be performed on different machines.
  - Perform final, reported testing on a dedicated or lightly loaded system.
  - Employ statistical experimental design techniques. This powerful, often neglected, methodology can highlight those factors that contributed to the results, as well as those that did not.
- Provide a comprehensive report of the results
  - Use the graph of solution quality versus computational effort expended (elapsed time, number of iterations, etc.) where possible for describing your results.
  - For summary measures, use measures of central tendency, variability, and cost-effectiveness.

  - Use graphics where possible and when informative.
  - Provide as much detail as possible. If a journal will not publish all pertinent data—perhaps due to space limitations—make them available in a research report.
  - Describe the sensitivity of the code to changes in the tuning strategy.
  - Be courageous and include your “failures,” since they provide insight also.

These standards should be viewed as guidelines, not as a fixed set of norms that cannot be violated.

5 Conclusions and Future Directions

- An alternative approach to algorithm analysis and testing is being investigated: algorithm modeling[17, 10].
- Larger, more accessible, computer-based archives of benchmark problems and test beds.
- Support for the experimentation process by the *Journal of Heuristics*.

References

[1] *The American Heritage Dictionary of the English Language* (3rd ed.), 1992, Houghton Mifflin, Boston, MA.

[2] Amini, M.M., and R.S. Barr, 1990. Network Reoptimization Algorithms: A Statistically Designed Comparison. *ORSA Journal on Computing* .

[3] Barr, R., and B. Hickman, 1993. Reporting Computational Experiments with Parallel Algorithms: Issues, Measures, and Experts' Opinions (feature article), *ORSA Journal on Computing* 5:1, 2-18.

[4] Cormen, T.H., C.E. Leiserson, and R.L. Rivest, 1990. *Introduction to Algorithms*, MIT Press, Cambridge, MA.

[5] Crowder, H.P., R.S. Dembo, and J.M. Mulvey, 1980. On Reporting Computational Experiments with Mathematical Software.*ACM Transactions on Mathematical Software* 5, 193-203.

[6] Feo, T.A., M.G.C. Resende, and S.H. Smith, 1994. A Greedy Randomized Adaptive Search Procedure for Maximum Independent Set, *Operations Research* 42, 860-878.

[7] Garey, M.R., and D.S. Johnson, 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman.

[8] Gilsinn, J., K. Hoffman, R.H.F. Jackson, E. Leyendecker, P. Saunders, and D. Shier (1977). Methodology and Analysis for Comparing Discrete Linear L1 Approximation Codes. *Communications in Statistics 136*, 399-413.

[9] Greenberg, H.J., 1990. Computational Testing: Why, How and How Much. *ORSA Journal on Computing 2*, 7-11.

[10] Hooker, J., 1993. Needed: An Empirical Science of Algorithms, *Operations Research* 42:2.

[11] Jackson, R.H.F., P.T. Boggs, S.G. Nash, and S. Powell, 1990. Report of the Ad Hoc Committee to Revise the Guidelines for Reporting Computational Experiments in Mathematical Programming. *Mathematical Programming 49,* 413-425.

[12] Jackson, R.H.F., and J.M. Mulvey, 1978. A Critical Review of Comparisons of Mathematical Programming Algorithms and Software (1953-1977). *J. Research of the National Bureau of Standards 83,* 563-584.

[13] Johnson, David. ???

[14] Lin, B.W. and R.L. Rardin, 1979. Controlled Experimental Design for Statistical Comparison of Integer Programming Algorithms, *Management Science* 25:12, 33-43.

[15] Lin, S. and B.W. Kernigan, 1973. An Effective Heuristic Algorithm for the Traveling-Salesman Problem, *Operations Research* 21:2, 498-516.

[16] Mason, R.L., R.F. Gunst, and J.L. Hess, 1989. *Statistical Design and Analysis of Experiments.* John Wiley & Sons, New York.

[17] McGeoch, C.C., 1995. Toward an Experimental Method for Algorithm Simulation, *INFORMS Journal on Computing*, to appear.

[18] Montgomery, D.C., 1984. *Design and Analysis of Experiments*, John Wiley, New York.

[19] Nance, R.E., R.L. Moose, Jr., and R.V. Foutz, 1987. A Statistical Technique for Comparing Heuristics: An Example from Capacity Assignment Strategies in Computer Network Design, *Communications of the ACM* 30:5, 430-442.

[20] Purdom, P.E., Jr. and C. A. Brown, 1985. *The Analysis of Algorithms.* Holt, Rinehart and Winston, New York.

[21] Stewart, W.R., J. Kelly, and M. Laguna. ???