

AN UPDATED PROPOSAL FOR A TESTBED OF SIMULATION-OPTIMIZATION PROBLEMS

Raghu Pasupathy

Industrial and Systems Engineering
Virginia Tech
Blacksburg, VA 24061, U.S.A.

Shane G. Henderson

School of OR and IE
Cornell University
Ithaca, NY 14853, U.S.A.

1 INTRODUCTION

We propose ideas for a testbed of simulation-optimization problems. The purpose of the testbed is to encourage development and constructive comparison of simulation-optimization techniques and algorithms. The ideas we present are revised from the copious and valuable feedback provided on an earlier presentation of the same topic at the 2006 Winter Simulation Conference (Pasupathy and Henderson 2006). Our aims in proposing this topic for the simulation workshop remain the same as those for the 2006 Winter Simulation conference: (i) to inform the simulation community of our effort; (ii) to obtain feedback from the simulation community on the revised testbed design; and (iii) to solicit SO problems for inclusion in the testbed.

Our proposed presentation for the workshop, like the organization of this abstract, will follow four broad topic areas: (i) Problem Format, (ii) Problem Taxonomy, (iii) Algorithm Performance Measures, and (iv) Problem Specification. Our ideas on some of these topics are more mature than others. For example, topic (iii) is an important area where various issues remain unresolved. So, we expect that this will be a major focus in the workshop although we remain open to suggestions in the other areas as well.

We continue this section with a formal SO problem statement, followed by a more elaborate description of our motivations behind the testbed.

1.1 SO Problem Statement

The Deterministic-Optimization (DO) problem is to

$$\begin{aligned} & \text{minimize} && g(x) \\ & \text{subject to} && h(x) \geq 0 \\ & && x \in \mathfrak{D}, \end{aligned}$$

where $g : \mathfrak{D} \rightarrow \mathbb{R}$ and $h : \mathfrak{D} \rightarrow \mathbb{R}^d$ are the real-valued objective function and vector-valued constraint functions respectively, and $\mathfrak{D} \subseteq \mathbb{R}^q$ is an underlying set of

potential solutions. In many cases h is vacuous, and the problem is then said to be unconstrained. The set \mathfrak{D} is arbitrary, so it can represent any level of constraints. Typically, however, \mathfrak{D} represents a natural and easily-defined set of potential solutions, such as the nonnegative orthant.

The Simulation-Optimization (SO) problem is a generalization of the DO problem where the objective is the same, i.e., a solution x^* is sought that minimizes g over \mathfrak{D} while also satisfying $h(x) \geq 0$. The difference is that now g and/or h are only observable through a stochastic simulation, and therefore with error. We assume estimators $G_m(x)$ of $g(x)$, and $H_m(x)$ of $h(x)$, that are consistent, in the sense that $G_m(x) \Rightarrow g(x)$ and $H_m(x) \Rightarrow h(x)$ as $m \rightarrow \infty$, for any $x \in \mathfrak{D}$. Here \Rightarrow denotes convergence in distribution, and m is some measure of simulation effort. For example, when h and g are performance measures associated with a finite-horizon simulation, m might represent the number of independent and identically distributed (i.i.d.) replications. We assume that the domain \mathfrak{D} is deterministic and known.

In many cases, the SO problem possesses structure which can be used to assist in the search for a minimum. For example, if g is differentiable and one can obtain estimates of the gradients of g , then one can employ methods that exploit gradient information.

While one would like to identify a *global* minimum, this goal is, in general, exceedingly difficult to achieve, even in the case where g can be computed without simulation error. For example, if g is completely unstructured, then to guarantee that a global minimum has been found, one must compute g at every feasible solution. Accordingly, we tailor our discussion to the goal of finding *local* minima.

1.2 Motivation

There has recently been a marked growth in the number and type of instances where SO problem formulations apply, often with only minor variations. There

has also been a corresponding increase in the number and diversity of solutions to these SO problem variants. Recognizing this, a panel discussion at the 2000 Winter Simulation Conference (Fu et al. 2000) identified the need for a testbed of SO problems which, among other things, can be used to evaluate and compare competing SO algorithms, and to identify particular problem classes for further inquiry. This need was reiterated in Fu (2002) and in comments to that article in Glynn (2002). The use of testbeds is well-established in other fields. See Jackson et al. (1991) for examples, as part of an update to a set of guidelines (Crowder et al. 1979) on reporting computational results. For early discussion in the mathematical programming community, see Mulvey (1982).

A SO testbed is primarily motivated by the following considerations:

1. a testbed is useful in comparing the performance of competing algorithms through execution on the same problem instances;
2. a testbed helps to identify particular SO problem classes that have defied efficient solution and in the process stimulates algorithm development for these classes;
3. a testbed increases the visibility of SO problem tools among researchers and practitioners thereby leading to their increased use.

It is worth emphasizing that the testbed we propose is intended not as an archive of “unsolved” problems in SO, but instead as a forum that presents a range of SO problems to facilitate research and application. Furthermore, to keep the scope of the proposed endeavor manageable, we do not plan to include problems with recourse, where successive decisions are made over time. So we do not plan to include stochastic linear programs, nor the more general stochastic dynamic programs (also known as Markov decision problems). In general, these problems cannot be formulated as SO problems. Having said that, versions of these problems can sometimes be formulated within our framework, as occurs when one has a finite parametrization of a class of policies. In that case, we will not specifically exclude it.

We recognize that a potential consequence of a testbed is the emergence of algorithms tailored for solving the particular instances appearing in the testbed. This negative consequence can be mitigated by ensuring that particular problem classes are not overly represented, that all problem classes contain a rich set of problems, and through the careful choice of performance measures for reporting algorithm performance. For further discussion of the merits and potential pitfalls of testbeds, see Jackson et al. (1991) and the references therein.

2 PROBLEM FORMAT

We propose three general classes of problem formats to cover the gamut of SO problems.

1. High-level Description (HD): In this format, the SO problem is specified through a detailed, verbal description of the problem. For example, the famous newsvendor problem as an SO problem in HD format is:

A newsvendor orders a fixed quantity x of newspapers to be sold each day. The cost to the newsvendor, and the selling price respectively, of each newspaper is c and s . Unsold newspapers are salvaged each day at the unit price w . The daily demand D is Poisson distributed with mean λ . What is the quantity x that maximizes the expected profit for the newsvendor?

Sets of values for λ, c, s and w would then be specified, corresponding to particular problem instances. An important feature of this problem is that the optimal solution is known. Another important feature is that x can be viewed as an integer-ordered variable as in the newspaper example, or as a continuous quantity if, for example, the problem relates to stocks of a perishable chemical.

2. Simulation Blackbox (SB): In SB format, one has some form of black box that, given a design point x and simulation runlength m (broadly interpreted), returns estimates of $g(x)$ and $h(x)$. If available, the blackbox also returns estimates of the gradients, $\nabla g(x)$ and $\nabla h(x)$, at the design point x . The black box could be, for example, code in some general-purpose programming language, or a model encoded in a specific simulation language.

The SB format may be useful in contexts where the model complexity defies easy problem description. As an example, in the newsvendor problem described above, suppose that the daily demand is not known to be Poisson but is instead the output of another module. Then the problem in SB format is a program that, for a given design x , uses the demand generation module to generate a random demand, and then delivers the corresponding realized profit for each day.

An important weakness of this approach is that it is highly dependent on the black box being implementable on different platforms. This is especially important in view of the fast pace at which hardware and software platforms are evolving. Therefore, while we believe it is important to include such descriptions, they are likely to date rather quickly.

3. **Objective Function–Feasible Space–Error (OFE):** In the OFE format, the SO problem representation is more explicit and specified through an objective function, a set of inequalities that constitute the feasible space, and the error distribution, all provided in closed form. For example,

$$\begin{aligned} g(x) &= 2x_1^2 + x_2^2, \\ x_1x_2 &\geq 2, \\ x_1 &\geq 0, \\ x_2 &\geq 0, \\ G_m(x) &= g(x) + N(0, (x_1^2 + x_2^2)/m), \end{aligned}$$

where $N(a, b)$ denotes a normal random variable with mean a and variance b . Here we have not specified the correlation structure of the error for different x s, or for different runlengths m . This is important, for example, in the setting of common random numbers. One such structure assumes that the errors are independent for different x s, and that for a given x , $G_m(x)$ is a sample average of m i.i.d. $N(g(x), x_1^2 + x_2^2)$ random variables, with the samples used for different runlengths being mutually independent. It is difficult to specify more general dependence structures in this format, and this is a weakness of the approach. However, problem descriptions such as this are needed in the library, because they represent problems for which much is known about the problem. Therefore, they can be used to identify particular strengths and weaknesses of algorithms.

Each problem also includes a brief description of what may be assumed about the problem. For example, in the newsvendor problem, one would clarify whether x should be viewed as continuous or integer-ordered. Furthermore, newsvendor problems are known to have certain convexity properties. One would specify whether this information can be assumed or not.

3 PROBLEM TAXONOMY

Algorithms are usually tailored to particular features of SO problems. For example, problems with discrete variables usually require quite different algorithms than problems involving continuous variables and a differentiable objective function. Accordingly, we provide a taxonomy of SO problems, similar in many respects to that provided in Barton and Meckesheimer (2006), based on important and identifiable properties of the feasible region, objective function and constraints. We plan to use the taxonomy to classify problems in the testbed.

In Figure 1 we first determine whether the set \mathcal{D} consists of categorical, integer-ordered, or continuous variables. Categorical variables are those that cannot be ordered in an appropriate way, such as variables corresponding to a choice of queue discipline. Integer-ordered variables usually represent a count of discrete entities, such as agents in a call center. Continuous variables take values in \mathbb{R} . If a problem contains a mix of variable types, then we classify it as categorical if it has categorical variables, or integer-ordered otherwise.

Integer-ordered problems and continuous problems are further classified as to whether they are constrained (h is non null) or unconstrained. Continuous-variable problems are also classified as to whether their objective function and constraint functions are known to be smooth (continuously differentiable) or non-smooth. In cases where it is not known whether the functions are smooth or non-smooth, the problems are classified as non-smooth.

4 MEASURING ALGORITHM PERFORMANCE

The literature on simulation optimization is replete with proofs of convergence of algorithms. Such proofs are widely viewed as being an important feature of algorithms: given enough computational effort, one would hope that an algorithm converges to at least a locally optimal solution. Unfortunately, it is often the case, or perhaps usually the case, that the computational effort required to reach a point where such asymptotic results are informative is too large to be practical. In that case, the asymptotic results are not as useful as one might hope. In order to compare algorithms on a more appropriate time scale, it seems that we need to evaluate their finite-time performance.

Finite-time performance evaluation of algorithms is an important sub-topic that we address in this research. We categorize our discussion of finite-time performance measures according to whether a user (i) operates on a fixed computational budget, or (ii) seeks a solution that satisfies a specified tolerance criterion. Such a categorization brings forth several important questions for which we have answers to varying extents at the present time. For example,

- (1) how do we measure the computational budget, especially considering the fact that the total computational burden placed on an algorithm comes from potentially very different sources such as simulation calls for objective function or constraint estimates, simulation calls for gradient estimates, and actual processor computation?
- (2) what is an ideal way to depict the performance of

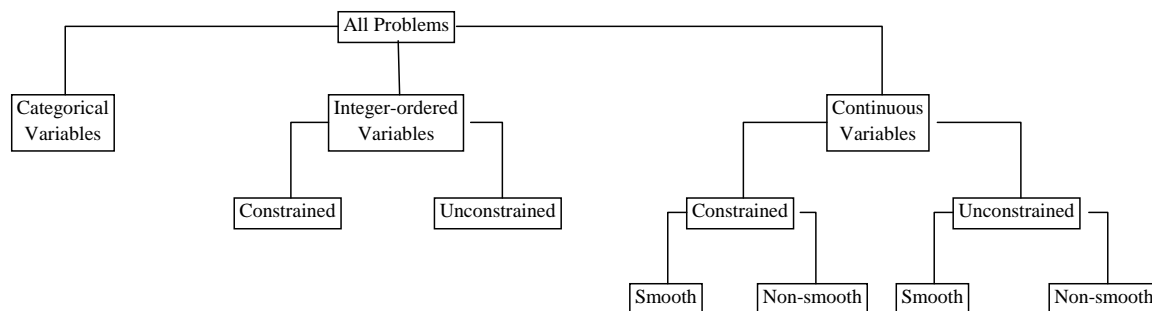


Figure 1: A SO Problem Taxonomy

an algorithm for various computational budgets, as reflected across users, and how do we compute this depiction?

- (3) what is a reasonable and computationally feasible way of defining the tolerance criterion mentioned in (ii)?
- (4) how does one modify answers to (2) and (3) when the constraints occurring in the SO problem have to be estimated in addition to the objective function, thereby making the feasibility of the returned solution uncertain?
- (5) what ideal and estimable summary statistics best illustrate the performance of an algorithm on a testbed as opposed to a single problem?
- (6) how can the summary statistics in (5) be used to compare competing algorithms on a testbed?

5 PROBLEM SPECIFICATION

It is expected that a simulation-optimization problem appearing in the testbed is specified through the following characteristics:

- (1) problem statement (in one of the recommended formats);
- (2) recommended parameter settings;
- (3) recommended measurement of time;
- (4) a method for obtaining a starting solution for algorithms that require a single starting point, and a method for generating a collection of starting solutions for algorithms that require a family of solutions, e.g., genetic algorithms;
- (5) any information about the optimal solution(s);
- (6) comments on any known structure in the problem;
- (7) recommended parameters to be used when reporting algorithm performance.

ACKNOWLEDGMENTS

We would like to thank Michael Saunders for helpful discussions. This work was partially supported by National Science Foundation Grant DMI 0400287.

REFERENCES

- Barton, R. R., and M. Meckesheimer. 2006. Metamodel-based simulation optimization. In *Handbook of Simulation*, ed. S. G. Henderson and B. L. Nelson, 535–574. Elsevier.
- Crowder, H. P., R. S. Dembo, and J. M. Mulvey. 1979. On reporting computational experiments with mathematical software. *ACM Transactions on Mathematical Software* 5:193–203.
- Fu, M. C. 2002. Optimization for simulation: theory vs. practice. *INFORMS Journal on Computing* 14:192–215.
- Fu, M. C., S. Andradóttir, J. S. Carson, F. G. J. P. Kelly, and S. M. Robinson. 2000. Integrating optimization and simulation: research and practice. In *Proceedings of the 2000 Winter Simulation Conference*, ed. J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, 610–616. Piscataway NJ: IEEE.
- Glynn, P. W. 2002. Additional perspectives on simulation for optimization. *INFORMS Journal on Computing* 14:220–222.
- Jackson, R. H. F., P. T. Boggs, S. G. Nash, and S. Powell. 1991. Guidelines for reporting results of computational experiments. report of the ad hoc committee. *Mathematical Programming* 49 (3): 413–425.
- Mulvey, J. M. (Ed.) 1982. *Evaluating mathematical programming techniques*, Volume 199 of *Lecture Notes in Economics and Mathematical Systems*. New York: Springer-Verlag.
- Pasupathy, R. and S. G. Henderson. 2006. A testbed of simulation-optimization problems. In *Proceedings of the 2006 Winter Simulation Conference*, ed. L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, 255–263. Piscataway NJ:

IEEE.

AUTHOR BIOGRAPHIES

RAGHU PASUPATHY is an assistant professor in the Industrial and Systems Engineering department at Virginia Tech. His research interests lie in general simulation methodology, with a current focus on simulation optimization and stochastic root finding. He is a member of INFORMS and IIE.

SHANE G. HENDERSON is an associate professor in the School of Operations Research and Industrial Engineering at Cornell University. He has previously held positions at the University of Michigan (Ann Arbor) and the University of Auckland. He is the simulation area editor at *Operations Research*, and an associate editor for the *ACM Transactions on Modeling and Computer Simulation* and *Operations Research Letters*. He likes cats but is allergic to them. His research interests include discrete-event simulation and simulation optimization.