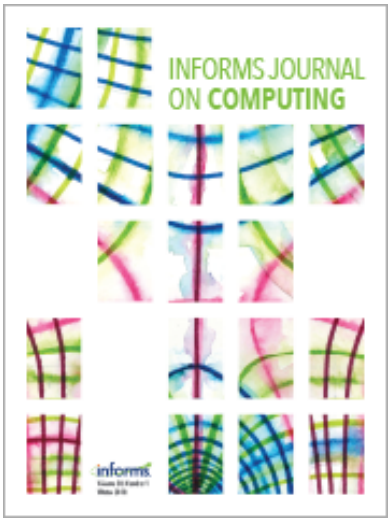


INFORMS Journal on Computing

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Diagnostic Tools for Evaluating and Comparing Simulation-Optimization Algorithms

David J. Eckman, Shane G. Henderson, Sara Shashaani

To cite this article:
David J. Eckman, Shane G. Henderson, Sara Shashaani (2023) Diagnostic Tools for Evaluating and Comparing Simulation-Optimization Algorithms. INFORMS Journal on Computing

Published online in Articles in Advance 05 Jan 2023

. https://doi.org/10.1287/ijoc.2022.1261

https://pubsonline.informs.org/journal/ijoc

INFORMS JOURNAL ON COMPUTING
Articles in Advance, pp. 1–18
ISSN 1091-9856 (print), ISSN 1526-5528 (online)

Diagnostic Tools for Evaluating and Comparing Simulation-Optimization Algorithms

David J. Eckman,[a,*] Shane G. Henderson,[b] Sara Shashaani[c]

[a] Wm Michael Barnes '64 Department of Industrial and Systems Engineering, Texas A&M University, College Station, Texas 77843; [b] School of Operations Research and Information Engineering, Cornell University, Ithaca, New York 14853; [c] Edward P. Fitts Department of Industrial and Systems Engineering, North Carolina State University, Raleigh, North Carolina 27695

*Corresponding author

Contact: eckman@tamu.edu, https://orcid.org/0000-0002-6473-6434 (DJE); sgh9@cornell.edu, https://orcid.org/0000-0003-1004-4034 (SGH); sshasha2@ncsu.edu, https://orcid.org/0000-0001-8515-5877 (SS)

Received: September 15, 2021
Revised: June 7, 2022; October 24, 2022; November 8, 2022; November 18, 2022
Accepted: November 19, 2022
Published Online in Articles in Advance: January 5, 2023

https://doi.org/10.1287/ijoc.2022.1261

Abstract. Simulation optimization involves optimizing some objective function that can only be estimated via stochastic simulation. Many important problems can be profitably viewed within this framework. Whereas many solvers—implementations of simulation-optimization algorithms—exist or are in development, comparisons among solvers are not standardized and are often limited in scope. Such comparisons help advance solver development, clarify the relative performance of solvers, and identify classes of problems that defy efficient solution, among many other uses. We develop performance measures and plots, and estimators thereof, to evaluate and compare solvers and diagnose their strengths and weaknesses on a testbed of simulation-optimization problems. We explain the need for two-level simulation in this context and provide supporting convergence theory. We also describe how to use bootstrapping to obtain error estimates for the estimators.

History: Accepted by Bruno Tuffin, area editor for simulation.
Funding: This work was supported by the National Science Foundation [Grants CMMI-2035086, CMMI-2206972, and TRIPODS+X DMS-1839346].
Supplemental Material: The software that supports the findings of this study is available within the paper and its Supplementary Information [https://pubsonline.informs.org/doi/suppl/10.1287/ijoc.2022.1261] or is available from the IJOC GitHub software repository (https://github.com/INFORMSJoC) at [http://dx.doi.org/10.5281/zenodo.7329235].

Keywords: analysis of algorithms • simulation • design of experiments • efficiency

1. Introduction and Motivation

Simulation optimization (SO) involves the optimization of some objective function over a (possibly constrained) feasible region, in which at least one of the objective and constraint functions is estimated through a stochastic simulation. The decision variables for such problems can be continuous, integer-ordered, or even categorical. Such problems are typically highly challenging because stochastic simulation yields estimators that are slow to converge; the canonical error is of stochastic order $c^{-1/2}$, where c is a measure of the computational effort devoted to the simulation (Asmussen and Glynn 2007). Moreover, many SO problems lack structure, such as smoothness, that might be exploited by specialized solvers, that is, implementations of algorithms for solving SO problems.

We consider optimization problems of the form $\{\min f(x) \mid x \in \mathcal{D}\}$, where $\mathcal{D}$ is a domain. We assume that f is estimated through stochastic simulation, and for simplicity, we assume that determining whether a point x lies in the domain $\mathcal{D}$ does not require simulation; that is, simulation is needed only in estimating f. This setup excludes the case in which constraints of the form $g(x) \geq 0$ must be satisfied, where $g(x)$ is estimated by simulation. Such problems arise in practice, but we do not yet have a recommendation on performance metrics for them. Problems with deterministic constraints, that is, those in which any constraint functions are deterministic and readily evaluated, are included.

The development of SO solvers is an active area of research. Much effort is devoted to the design of solvers with provable convergence guarantees, whether to local or global solutions, for example, Kushner and Yin (2003), Andradóttir (2006, 2015), Cooper et al. (2020), and Li and Ryzhov (2022). Such convergence analyses are valuable and insightful yet are typically most relevant in an asymptotic regime in which the computational effort becomes very large, especially in the case of results for global optimization. Because that regime can be difficult to reach in practice, there is a need to better understand the preasymptotic regime, in which solvers have yet to narrow in on a neighborhood of an optimal solution. This regime can be very difficult to explore analytically although some results are available, for example, Ghadimi and Lan (2015).

The preasymptotic regime can be investigated through the use of a testbed of SO problems and solvers; see Fu (2002) and Glynn (2002) for the germ of this idea and, for example, Chau and Fu (2015) and Dong et al. (2017) for recent examples of simulation experiments assessing the relative performance of SO solvers. See also section 5 of Amaran et al. (2016) for a discussion on SO solver comparisons and the need and utility of testbeds. Such experiments are important for a number of reasons. First, they can help with the development of new solvers by providing a testbed of problems and helping identify good choices of a solver's parameters through calibration over a set of problems. Second, they can help determine which solvers are effective when run with practically relevant computational budgets; a solver can be viewed as effective when it solves problems rapidly and reliably. Third, they can help identify problems or classes of problems that are relatively easy to solve or that defy efficient solution with existing solvers, thereby motivating further solver development and ensuring that research effort remains focused on more challenging problems. They can likewise provide insight into the structural properties of problems, such as convexity and/or pathwise smoothness, that are especially well or poorly handled by a solver.

Testbeds are of great value to research communities outside of SO (Ali et al. 2005, Gould et al. 2015, Netlib 2021, Wikipedia 2021). Compared with these other communities, the SO research community lags in the development of testbeds, but there has been recent progress. SimOpt is a library of simulation-optimization problems and solvers that is undergoing a redesign. See Pasupathy and Henderson (2006, 2011), Dong et al. (2017), and Eckman et al. (2019, 2022b) for background and recent developments and Eckman et al. (2020) for the library itself.

In parallel with the development and collection of problems and solvers, one also needs metrics for evaluating and comparing solver performance. This paper focuses on the development of such metrics and methods for estimating them. Solvers are compared on the basis of their progress in improving the objective function value as a function of completed simulation replications.

In developing metrics, we attempt to capitalize on related metrics developed in other optimization communities, namely, performance profiles (Dolan and Moré 2002, Ali et al. 2005, Gould and Scott 2016), data profiles (Moré and Wild 2009), accuracy profiles (Beiranvand et al. 2017), and log-ratio profiles (Morales 2002, Shi et al. 2021). There are special aspects of SO that prevent direct translation of the aforementioned metrics. For example, the fact that we cannot exactly (to numerical precision) evaluate an objective function and instead must estimate it through stochastic simulation means that we can never be certain that one solution is better than another or that a solution is close in objective function value to an optimal solution. Still, we can strive to make assertions with high confidence by controlling sample sizes and dependence structures through judicious use of common random numbers (CRN). In general, the estimation issues present in developing metrics and associated estimators are nontrivial.

The complexity of SO is reflected in related work in stochastic programming on assessing solution quality. Bayraksan and Morton (2006) develop new interval estimates of the optimality gap of a proposed solution that apply, at least in principle, to our setting. To compute their interval estimates, one needs to be able to obtain the globally optimal solution to a sample version of an SO problem. To that end, the examples presented therein are convex, whereas we do not assume convexity. Still, that work inspires our use of scalings relative to the objective function value of optimal or near-optimal solutions. Bayraksan and Morton (2009) review and extend Bayraksan and Morton (2006) to include bias reduction, jackknifing, randomized quasi-Monte Carlo methods, and sequential sampling. Convexity is also central to the work in Lan et al. (2012) for obtaining bounds in convex SO problems when using mirror-descent algorithms. Many of these ideas are reviewed in the survey of Homem-de Mello and Bayraksan (2014), which also discusses statistical hypothesis testing of Karush–Kuhn–Tucker conditions for a given solution. Such methods are specific to a proposed solution and, thus, not suitable for our purpose of comparing solvers.

We view the primary contributions of this paper to be the following:

- We identify important characteristics of solver performance and ways to measure them.
- We recommend an experimental design that accommodates many SO solvers. The design typically requires the use of two-level simulation, and we exploit that literature to inform the experimental design. Our experimental setup is readily implemented and has been fully implemented in SimOpt (Eckman et al. 2020, 2022b).
- We develop a variety of metrics and clarify their uses in evaluating one or more solvers on one or more problems, namely, progress curves, terminal progress plots, area and terminal scatterplots, solvability profiles, and difference profiles. Unnormalized progress curves that show the estimated objective function of the estimated best solution seen so far have been used for a long time to track solver progress on a single problem (e.g., Pasupathy and Henderson 2006) and compare multiple solvers on a single problem (e.g., Dong et al. 2017). Our metrics considerably extend these ideas, building on related ideas from deterministic-optimization testbeds. Our proposed metrics do not align perfectly with metrics used in testbeds for deterministic-optimization methods; we explain what is unique about SO to justify our choices of metrics. Progress curves and solvability profiles are

related to plots used in other research fields, for example, convergence plots and data profiles, whereas terminal progress plots, scatterplots, and difference profiles are new.

- We draw a distinction between budget-specific solvers and budget-agnostic solvers. The former use prior knowledge of the budget to set key parameters that control solver behavior. For example, Nemirovski et al. (2009) provide a stochastic approximation solver in which the step-length sequence and solution-averaging weights are both budget-specific. In contrast, budget-agnostic solvers do not exploit prior knowledge of the budget. Our focus is budget-agnostic solvers, but we also discuss the evaluation of budget-specific solvers and comparisons between solvers from the two classes.
- We make judicious use of CRN to improve various estimators.
- We describe a bootstrapping approach for assessing the estimation error in our estimated metrics.

This paper is organized as follows. Section 2 describes characteristics of SO problems and solvers in more detail and explains the need for so-called macroreplications and postreplications. Section 3 introduces progress curves as a starting point for measuring solver performance on a single problem on a single macroreplication. We describe two ways to aggregate these curves over multiple macroreplications, expanding on ideas sketched out in Pasupathy and Henderson (2006), and discuss estimators for such aggregate progress curves. These estimators use two-level, that is, nested, simulation, wherein macroreplications constitute the outer level and postreplications constitute the inner. Section 4 explores the use of the area under progress curves to efficiently summarize the performance of multiple solvers on multiple problems, also providing estimators. Section 5 proposes solvability profiles, which are closely related to data profiles as originally proposed in Moré and Wild (2009). Solvability profiles are based on the (random) time required to approximately solve a problem as opposed to progress curves, which are based on the (random) improvement in objective function value as a function of time. Solvability profiles as well as their differences, which we call "difference profiles," allow a succinct comparison of the performance of multiple solvers on multiple problems. Section 6 discusses evaluating the finite-time performance of both budget-agnostic and budget-specific solvers; unless otherwise noted, we assume budget-agnostic solvers. Section 7 provides examples of the comparative plots we advocate. Section 8 discusses convergence rates of our estimators, appealing to the literature on two-level simulation. We conclude in Section 9. The online supplement provides further details of the numerical examples in Section 7, the proof of a result given in Section 8, and more details of our use of bootstrapping to assess estimation errors in the various metrics.

2. Problems, Solvers, and Experimental Design

It is natural to use a testbed of SO problems to evaluate and compare SO solvers. In doing so, we want to exploit the literature on the use of testbeds in related areas, such as in linear programming and stochastic linear programming. SO problems and solvers, however, possess unique features that we must consider. For instance, a linear program in standard form can be fully specified by a vector of objective function coefficients, a constraint matrix, and a vector of constraint right-hand-side values. Such a succinct description allows one to readily describe problems and to write code to generate random problem instances. We find ourselves somewhat envious of this relative simplicity when we contemplate specifying SO problems. To do so, one must describe a mechanism through which sample paths can be constructed (e.g., a stochastic simulation model) and precisely define the performance measures.

Given this complexity, it is very tempting to instead use "synthetic" SO problems in which one takes a deterministic-optimization problem for which the objective and any constraints are specified using closed-form formulae and "corrupts" the problem by adding mean-zero stochastic noise. For example, one might add mean-zero Gaussian noise to the Rosenbrock (1960) function. From a testing standpoint, this offers several advantages:

- The optimal solution is the same as that of the original deterministic problem, which is often known.
- Simulation replications (i.e., function evaluations) are computationally inexpensive.
- Many high-dimensional, deterministic optimization problems exist for testing.
- Such objective functions typically have known structure.

The main disadvantage of this approach is that (solution-dependent) variances, gradient estimators, and effects of CRN are all highly artificial. For example, a perfect instantiation of CRN—in which the same additive stochastic error is applied at all feasible solutions—yields sample-path functions that resemble the original deterministic function, just shifted vertically by a common offset. Thus, a deterministic-optimization solver that ignores the stochastic nature of the problem should perform very well although it might struggle on general SO problems. We believe synthetic problems are problematic when it comes to comparing SO solvers and do not consider them further.

For each problem we study, we specify a budget of T simulation replications within which solvers must operate. We measure budgets in terms of replications because this definition is platform-independent in that performance measures do not change when run on computers with different architectures. Still, this measure has its disadvantages because it does not account for solver-

specific computation apart from simulation replications. For example, a solver may exploit additional information, such as direct gradient estimates (Fu 2006), or need to factor progressively larger matrices when using Gaussian-process predictions, but the associated computational burden is not counted. Moreover, in asynchronous parallel computing environments, it is challenging to unambiguously define which replications come first. Our choice to measure budget in simulation replications is, therefore, imperfect. As long as it is clearly understood and the results are transparently presented, we consider it to be a reasonable choice.

Solvers are tasked with finding an estimated best solution and not exceeding a budget T though we are also interested in their performance at intermediate budgets smaller than T. Budget-agnostic solvers do not tailor their parameters to the budget and, thus, can perform a single run until the budget T is exhausted, reporting the estimated best solution seen at intermediate budgets as they proceed. On the other hand, budget-specific solvers tailor their parameters to each intermediate budget. Unless otherwise stated, we focus on budget-agnostic solvers though we discuss budget-specific solvers in some depth in Section 6. We do not accommodate solvers that require a random budget, such as many ranking-and-selection algorithms that provide frequentist statistical guarantees. Our framework, thus, encompasses a diversity of algorithms, including gradient-descent, trust-region methods, direct-search methods, and evolutionary algorithms that track populations of solutions, but it is not completely general.

Simulation optimization gives rise to multiple sources of randomness. Function estimates obtained from simulation models are random, and some solvers, such as genetic algorithms, are intrinsically random. Therefore, a single run of a solver on a problem cannot yield a complete sense of the solver's potential performance. Consequently, we conduct multiple macroreplications; a macroreplication consists of a single run of a solver on a problem using up to the budget of T replications. Macroreplications collectively give a sense of the variable performance of a solver on a problem and are central to how we evaluate solvers. However, multiple macroreplications are not essential when solving an SO problem in practice.

On each macroreplication, a solver uses estimates of objective function values to guide a search for better solutions. Such a search is biased toward solutions whose estimated objective function value is lower (for minimization problems) than the true value because of random sampling (Mak et al. 1999). This phenomenon is sometimes known as optimization bias, and it means that the estimated objective function value at an estimated optimal solution is biased low. Accordingly, whenever we want to obtain unbiased estimators of the progress of SO solvers, we use a fresh set of simulation replications, which we term "postreplications," that are independent of those used in the macroreplications. Postreplications entail estimating objective function values and, thus, are conceptually simpler than macroreplications, which involve the interaction between a solver and a problem.

Our experimental design, thus, has two levels: we conduct multiple macroreplications of each solver on a given problem and obtain postreplications in a postprocessing stage. The distributional information we obtain from multiple macroreplications and postreplications then needs to be summarized in some fashion.

3. Progress Curves

In deterministic optimization, a common measure of performance of a single solver on a single problem is the time or number of function evaluations needed to find an optimal solution or a solution with objective value within some tolerance of the optimum. This form of evaluation does not easily translate to SO for the following reasons:

- Optimal solutions to SO problems are usually not known.
- In linear programming, for example, one can determine whether a given solution is optimal through a certificate of optimality, for example, complementary slackness conditions, without needing to know an optimal solution in advance. Such certificates are rare in SO problems, so even if an optimal solution were visited, usually it could not be identified as optimal.
- Suppose that an optimal solution x^* is known for an SO problem, and we want to check whether a candidate solution, x, has an objective function value, $f(x)$, within some given tolerance, ϵ, of $f(x^*)$. This determination is subject to stochastic error in the estimator $f_N(x) - f_N(x^*)$ obtained by averaging the outputs of N replications at each solution.
- The simulation error can be reduced by increasing the number of replications simulated at a solution, but this comes at a computational cost. This is not typically a feature of the deterministic-optimization setting.

For a given problem with the computational budget T, let $X(t)$ be the recommended solution when using an intermediate budget tT, $t \in [0,1]$. In a slight abuse of language, we refer to t as "time." For a fixed t, the recommended solution $X(t)$ is random, depending as it does on the outputs of the simulation replications and their influence on the solver's progress. Solvers may also deliberately inject additional randomness into this process, as with, for example, genetic or random-search algorithms. Collectively, the solutions recommended over a range of budgets are described by the stochastic process $\boldsymbol{X} := \{X(t) : t \in [0,1]\}$. Although the recommended solution can change at only discrete times $t = 0, 1/T, 2/T, \ldots, 1$, it is convenient to represent

$\boldsymbol{X}$ as a continuous-time stochastic process by assuming that, at any time t, it takes the value of the most recent recommended solution. Thus, $\boldsymbol{X}$ is piecewise constant in time, that is, once a solution is recommended, it plays the role of an incumbent solution until it is replaced by a new recommended solution.

We are especially interested in the stochastic process $f(\boldsymbol{X}) := \{f(X(t)) : t \in [0,1]\}$, where the random variable $f(X(t))$ is the true objective function value of the random recommended solution $X(t)$ obtained on a generic macroreplication. The process $f(\boldsymbol{X})$ describes the solver's progress in identifying better solutions over time and, for a given realization, that is, macroreplication, can be plotted over time. The deterministic analog of this plotted function is widely used in the deterministic-optimization community to study convergence and compare solvers on a given problem (Beiranvand et al. 2017). Such plots also frequently appear in the SO literature.

We find it useful to rescale the plot of $f(\boldsymbol{X})$, especially when we want to summarize a solver's performance over multiple problems as we do in Sections 4 and 5. Related scalings are used when reporting results for deterministic solvers on deterministic-optimization problems; see, for example, Moré and Wild (2009). To that end, let x_0 be a fixed initial solution, and let x^* be either a known optimal solution or some proxy for an optimal solution. We use x^* to set the vertical scaling in progress curves. If x^* is optimal, then progress curves are bounded below by zero. If x^* is suboptimal and some solver identifies a better solution, the effect is that the progress curve for that solver can drop below zero. Thus, it may be more useful to think of x^* as a reference solution, which, with the initial solution x_0, provides the vertical scaling in progress curves. Choosing x^* to be an optimal solution is ideal in that we can then interpret the vertical scale as a fraction of the initial optimality gap, but that interpretation is not essential.

We may choose x^* to be the best solution seen in an experiment, in which case it still has the same interpretation as a reference solution for vertical scaling with the proviso that the reference solution depends on the experiment. In this case, any curves or profiles are specific to the experiment and should not be compared with plots that result from different experiments. A potential concern with this approach is that the solver, say s, that identifies this best solution may appear to have better convergence properties than it actually does because the plot portrays convergence relative to the objective function value of this best solution. For example, solver s may appear in the progress plot to attain the truly optimal solution in finite time with positive probability. (Such performance seems unlikely, especially on problems with continuous decision variables, though it is conceivable that this could happen with discrete-decision-variable problems.) Still, our primary focus is on benchmarking, so the entire progress curve and not just its final value is of interest. So long as this context is clear, defining x^* in this manner is reasonable. If this method for defining x^* is deemed problematic, one can always obtain x^* in an alternative fashion, for example, through a macroreplication with a much larger budget. One might be concerned about optimality bias when x^* is chosen as the estimated best solution in an experiment. Certainly this choice of x^* could exhibit such bias, but because we only use x^* as a reference solution for vertical scaling, whereas it is the ordering between curves that is important, optimality bias does not cause fundamental difficulties.

The initial solution x_0 is assumed to be common across all macroreplications of a solver on a given problem. As a result, all of the variability observed in the quality of the random solution recommended at a given time, that is, $f(X(t))$, can be attributed to the simulation error in evaluating $f(\cdot)$, any internal randomness in the solver, and their effects on the solver's behavior. Whereas varying x_0 from macroreplication to macroreplication indicates how robust the solver is to starting from different initial solutions, it also leads to different rescalings of $f(X(t))$ for different macroreplications. Given these difficulties, we fix x_0 across macroreplications; different initial solutions on the same problem can be treated as distinct problems (Wild 2019).

For simplicity of exposition, we make several assumptions about the objective function values at x_0, x^* and elsewhere:

Assumption 1. *The objective function $f(x)$ is bounded above and below and attains its optimal value at $x = x^*$.*

Assumption 2. *The search does not start at an optimal solution, that is, $f(x_0) \neq f(x^*)$.*

The assumption that $f(\cdot)$ is bounded above is far stronger than needed but simplifies the discussion. It could be replaced by, for example, growth conditions on $f(\cdot)$ and conditions limiting the step size in solvers. The assumption that x^* is optimal allows us to assume a lower bound of zero on progress curves and to interpret the value of the progress curve as the relative optimality gap compared with the initial optimality gap; this assumption is not critical to the development. To normalize the plot of $f(\boldsymbol{X})$, we offset $f(X(t))$ by the optimal objective function value, $f(x^*)$, and divide by the initial optimality gap, $f(x_0) - f(x^*)$. We denote the rescaled stochastic process by $\boldsymbol{\nu} = \{\nu(t) : t \in [0,1]\}$, where

$$\nu(t) := \frac{f(X(t)) - f(x^*)}{f(x_0) - f(x^*)},$$

and refer to $\boldsymbol{\nu}$ as the "progress curve." We refer to $f(\boldsymbol{X})$ as the unnormalized progress curve. The value of the progress curve at a time $t \in [0,1]$ is a random variable giving the ratio of the optimality gap at time t to the optimality gap at time 0. Under Assumptions 1 and 2, $\nu(t)$ is finite

for all $t \in [0,1]$. For minimization problems, we expect that, with high probability, $f(x_0) \geq f(X(t)) \geq f(x^*)$ for all $t \in [0,1]$ so that $\nu(t) \in [0,1]$. On the other hand, for maximization problems, we expect that, with high probability, $f(x_0) \leq f(X(t)) \leq f(x^*)$, and again, $\nu(t) \in [0,1]$. In either case, a solver that recommends better quality solutions over time has $\nu(t)$ decrease toward zero as t increases. However, $\nu(t)$ can take values above one if the corresponding recommended solution $X(t)$ is worse than the initial solution x_0. Likewise, $\nu(t)$ can take values below zero if $X(t)$ is infeasible in a constrained optimization problem or if x^* is not optimal. Though not mathematically precise, we find it convenient to presume that $\nu(t)$ takes only values between zero and one when making general statements about the progress curve and related quantities.

Remark 1. The plot of $\boldsymbol{\nu}$ depends heavily on the choice of the budget, T. If x^* is a fixed, high-quality reference solution and T is very small, a solver will not make much progress toward finding x^* and $\nu(t)$ will hover near one. If T is very large, then a solver may make substantial progress toward an optimal solution relatively early and then remain near such an optimizer for the remaining time, in which case $\nu(t)$ quickly drops to zero and remains there. When comparing multiple solvers, the choice of the budget can favor slow-and-steady solvers (by increasing T) or rapid solvers (by decreasing T). Thus, care needs to be exercised in selecting T. The budget should ideally be large enough that a near-optimal solution can be found by some solver yet also small enough that differences in solvers' finite-time performance can be detected. We advocate setting it equal to the effort used by the best performing solver to get close to an optimal solution. This recommendation is imprecise; we believe necessarily so.

The progress curve, $\boldsymbol{\nu}$, is the principal random object by which we measure a solver's performance subject to a computational budget. By running multiple independent and identically distributed (i.i.d.) macroreplications of the solver and plotting the corresponding realizations of $\boldsymbol{\nu}$, one can visualize the run-to-run variability in a solver's progress over time on a given problem. The distribution of these random progress curves offers a wealth of information on different aspects of the solver's behavior. For instance, the distribution of $\nu(t)$ for any fixed t gives a sense of the reliability of the solver, for example, how consistently it recommends high-quality solutions at an intermediate budget. Also, "flattening" progress curves can indicate that a solver has stalled or reached a local optimum. In addition, the distribution of the first time at which $\nu(t)$ drops below some fixed threshold $\alpha \in [0,1]$ indicates how much time it takes the solver to reduce the relative optimality gap to α.

3.1. Aggregate Progress Curves

When extending to multiple solvers, simultaneously plotting multiple realizations of $\boldsymbol{\nu}$ for all solvers quickly becomes too cluttered, making it hard to draw clear conclusions. For this reason, we explore ways to aggregate or summarize aspects of $\boldsymbol{\nu}$. Different manipulations of $\boldsymbol{\nu}$ lead to valuable insights into the average progress, rate of progress, and run-to-run reliability of a solver on a given problem. In Sections 4 and 5, we extend these ideas to enable comparisons of multiple solvers on a set of problems. Collectively, these ideas rely on functions that take as input a stochastic process on the interval $[0,1]$ and output either a scalar or a deterministic function on the interval $[0,1]$. An example of the former includes the expected area under $\boldsymbol{\nu}$, whereas examples of the latter include the mean and median of $\nu(t)$ as functions of t.

Studying the distribution of the stochastic process $\boldsymbol{\nu}$ at a fixed time or a fixed remaining optimality gap α offers interesting connections to other methods for evaluating solver performance. For instance, fixing a remaining optimality gap and plotting the distribution of the time at which the solver attains that level of improvement is reminiscent of data profiles (Beiranvand et al. 2017). Similarly, fixing a time and plotting the distribution of the remaining optimality gap closely resembles a rescaled accuracy profile (Beiranvand et al. 2017). However, data and accuracy profiles were originally developed for comparing deterministic-optimization solvers in which the "distribution" in question comes from a solver's performance on a fixed set of problems.

We are interested in both the typical and tail behavior of $\nu(t)$ for each $t \in [0,1]$. To do so, we introduce aggregate progress curves and present their estimated counterparts in Section 3.2. As we use the term, an aggregate progress curve plots some summary measure, for example, the mean, median, or some other quantile of $\nu(t)$, as a function of t. The mean or median is frequently reported; see, for example, Dong et al. (2017).

The mean progress at time $t \in [0,1]$ is defined as $\mu(t) := \mathbb{E}\nu(t)$, where the expectation is over $X(t)$, the random solution recommended at time t starting from the (deterministic) initial solution x_0. For a fixed $\beta \in (0,1)$, the β-quantile progress at time $t \in [0,1]$ is defined as $\chi_\beta(t) := \inf\{q : \Pr\{\nu(t) \leq q\} \geq \beta\}$. Different choices of β accommodate a user's interest in a solver's median performance ($\beta = 0.5$) or tail performance in the direction of strong (e.g., $\beta = 0.05$) or poor (e.g., $\beta = 0.95$) performance. (When no confusion can arise, we suppress β in this notation.) To estimate the mean and β-quantile progress, we can obtain M i.i.d. macroreplications of the solver on the given problem, each yielding a sequence of recommended solutions $\boldsymbol{X}_m := \{X_m(t) : t \in [0,1]\}$ for $m = 1, 2, \ldots, M$. Throughout this paper, a subscript m indicates a quantity associated with macroreplication m; for example, $\boldsymbol{\nu}_m$ is the progress curve realized on macroreplication m. For problems in which we can exactly (to

numerical precision) compute $f(\cdot)$, the mean progress at time $t \in [0,1]$ can be estimated by averaging the progress from each macroreplication:

$$\mu(t;M) := \frac{1}{M}\sum_{m=1}^{M} v_m(t) = \frac{1}{M}\sum_{m=1}^{M} \frac{f(X_m(t)) - f(x^*)}{f(x_0) - f(x^*)}.$$

The β-quantile progress at time $t \in [0,1]$ can likewise be estimated by taking the sample β-quantile,

$$\chi(t;M) := \inf\left\{ q : \frac{1}{M}\sum_{m=1}^{M} \mathbb{I}(v_m(t) \le q) \ge \beta \right\}.$$

Remark 2. It is important to understand how many macroreplications suffice to get reasonably precise estimates of $\mu(t)$ and $\chi(t)$. Understanding the error associated with estimating $\mu(t)$ and $\chi(t)$ by $\mu(t;M)$ and $\chi(t;M)$, respectively, is worthwhile, yet we defer our discussion until the end of Section 3.2, where we consider a two-level simulation setting.

3.2. Estimated (Aggregate) Progress Curves

Typically, we cannot exactly compute $f(\cdot)$ and must resort to estimating the progress curve for a given realization of $\boldsymbol{X}$ via simulation. This setting is an instance of two-level simulation: in the outer level, macroreplications of the solver produce realizations of $\boldsymbol{X}$, and in the inner level, we obtain N postreplications at each recommended solution up to the budget. For a fixed time $t \in [0,1]$ and postreplication index $n = 1,2,\ldots,N$, let $Y_n(t)$ denote the corresponding noisy observation of the objective function value $f(X(t))$. The N postreplications $Y_1(t), Y_2(t), \ldots, Y_N(t)$ are assumed to be conditionally independent of one another and of the estimated function values obtained in the course of the macroreplications, conditional on the sequence of recommended solutions, $\boldsymbol{X}$.

As a default, we assume that CRN are used to evaluate the solutions recommended at different intermediate budgets on a given macroreplication; thus, for each $n = 1,2,\ldots,N$, $\boldsymbol{Y}_n := \{Y_n(t) : t \in [0,1]\}$ is a piecewise-constant function with breakpoints corresponding to times at which the recommended solution changes. This use of CRN is intended to produce stable estimated progress curves and reduce redundant simulation of a solution that is possibly recommended at multiple times on a given macroreplication. Although $\boldsymbol{X}$ is modeled as a continuous-time stochastic process, $\boldsymbol{Y}_n$ can easily be obtained by using the same random primitives to simulate each distinct solution in $\boldsymbol{X}$ as output by the solver.

In a similar manner, we obtain L postreplications at solutions x_0 and x^*, and for each postreplication index $l = 1,2,\ldots,L$, we let Y_{0l} and Y_{*l} denote the corresponding noisy observations. Here too, we assume that CRN are used to evaluate x_0 and x^*. We allow L to differ from N because estimates of $f(x_0)$ and $f(x^*)$ are needed for all times t at which we estimate $v(t)$. We, thus, expect to use a greater run length, L, for these common terms than the standard postreplication run length, N. Moreover, we foresee using the same estimates of $f(x_0)$ and $f(x^*)$ to construct the estimated progress curves from other macroreplications of the same solver.

We use the shorthand notation $f_N(\cdot)$ and $f_L(\cdot)$ to denote the sample averages of N and L i.i.d. postreplications, respectively, that is,

$$f_N(X(t)) := \frac{1}{N}\sum_{n=1}^{N} Y_n(t) \text{ for } t \in [0,1], \quad f_L(x_0) := \frac{1}{L}\sum_{l=1}^{L} Y_{0l},$$

$$\text{and} \quad f_L(x^*) := \frac{1}{L}\sum_{l=1}^{L} Y_{*l}.$$

Based on the postreplications, the estimated progress at time $t \in [0,1]$ is defined as

$$v(t;L,N) := \frac{f_N(X(t)) - f_L(x^*)}{f_L(x_0) - f_L(x^*)}.$$

We adopt the convention that the solver recommends the initial solution x_0 at time $t = 0$. Hence, we set $v(0;L,N) = 1$ even though $f_N(x_0)$ almost certainly does not equal $f_L(x_0)$.

For this setting in which $f(\cdot)$ must be estimated from simulation postreplications, the mean and β-quantile progress curves can be estimated by aggregating the estimated progress curves from each of M i.i.d. macroreplications. Specifically, the estimated mean progress at time $t \in [0,1]$ is

$$\mu(t;L,M,N) := \frac{1}{M}\sum_{m=1}^{M} v_m(t;L,N) = \frac{1}{M}\sum_{m=1}^{M} \frac{f_N(X_m(t)) - f_L(x^*)}{f_L(x_0) - f_L(x^*)},$$

where $v_m(t;L,N)$ is the estimated progress at time t from the mth macroreplication. Similarly, the estimated β-quantile progress at time $t \in [0,1]$ is the sample β-quantile, that is,

$$\chi(t;L,M,N) := \inf\left\{ q : \frac{1}{M}\sum_{m=1}^{M} \mathbb{I}(v_m(t;L,N) \le q) \ge \beta \right\}.$$

At any given time $t \in [0,1]$, the estimated aggregate progress values $\mu(t;L,M,N)$ and $\chi(t;L,M,N)$ are computed via two-level simulation.

As with any estimation procedure, statistical error can pose challenges. For example, at any given time $t \in [0,1]$ and macroreplication m, the estimated progress $v_m(t;L,N)$ may take a negative value if the solution $X_m(t)$ is misidentified as being better than x^* because of highly noisy function values and an insufficient number of postreplications. The statistical error in the estimators $\mu(t;L,M,N)$

and $\chi(t;L,M,N)$ can be assessed in several ways. Assuming finite second moments of $Y_1(t)$, Y_{01}, and Y_{*1}, standard confidence-interval machinery yields an interval estimator of $\mu(t)$ based on $\mu(t;L,M,N)$. Additional conditions are required to establish the validity of corresponding interval estimators of $\chi(t)$ based on $\chi(t;L,M,N)$. The analysis for both error estimators is discussed in Section 8, in which we use two-level simulation analysis to assess the asymptotic order of the error. From a practical standpoint, we prescribe a general bootstrapping approach that provides error estimates for these metrics (and others); see Online Appendix C for a full description.

Whereas aggregate progress curves summarize the typical and tail behavior of $v(t)$ for each $t \in [0,1]$, they cannot be easily extended to evaluate the performance of one or more solvers on multiple problems. For this reason, we introduce metrics that each reduce a collection of progress curves to a scalar rather than a function in the next two sections. Section 4 focuses on monitoring each solver's performance on each problem in a set of problems, whereas Section 5 provides a metric to compare the overall performance of the solvers.

4. Area Under Progress Curves

The area under the progress curve is given by the random variable

$$A := \int_0^1 v(t)\,dt = \int_0^1 \frac{f(X(t)) - f(x^*)}{f(x_0) - f(x^*)}\,dt.$$

Provided a solver is unlikely to recommend solutions worse than x_0 or better than x^*, A takes values in $[0,1]$ with high probability. Our rescaling of $f(X(t))$ addresses several shortcomings of a related metric proposed by Pasupathy and Henderson (2006): the area under the curve $f(\boldsymbol{X})$.

The random variable A can be interpreted as the solver's time-average relative optimality gap up to the budget T on a given macroreplication. Smaller values of A indicate better performance in the sense that the solver recommended better solutions on average throughout a given macroreplication. Although the area under the progress curve summarizes a solver's time-average progress, it does not capture all aspects of a solver's behavior given a fixed budget. For example, two progress curves can have the same area under them but exhibit very different solver behaviors with one showing steady progress over time and the other showing early rapid progress before plateauing. Although imperfect, distributional properties of A can be used for comparisons over multiple problems.

Let $\mu_A := \mathbb{E}A$ and $\sigma_A := \sqrt{\mathrm{Var}A}$ be the expectation and standard deviation of the area under the progress curve. The quantities μ_A and σ_A measure the average and variability of the time-average relative optimality gap with smaller values indicating better average and less-variable run-to-run performance, respectively. Under Assumptions 1 and 2, μ_A is also the area under the mean progress curve:

$$\mu_A = \mathbb{E}A = \mathbb{E}\left[\int_0^1 v(t)\,dt\right] = \int_0^1 \mathbb{E}v(t)\,dt = \int_0^1 \mu(t)\,dt.$$

The area under the mean progress curve, μ_A, combines the typical as well as especially poor or strong behavior of $v(t)$ but, otherwise, offers limited information about the distribution of $v(t)$. The standard deviation σ_A provides some information on reliability. It measures the across-macroreplication variability in how a solver's sequence of recommended solutions evolves depending on the data it observes.

We can estimate μ_A and σ_A by obtaining M i.i.d. macroreplications of the solver and calculating the sample mean and standard deviation of the areas under the realized progress curves. For the typical case in which $f(\cdot)$ must be estimated from postreplications, we can estimate μ_A and σ_A from the realized estimated progress curves:

$$\mu_A(L,M,N) := \frac{1}{M}\sum_{m=1}^{M} A_m(L,N) = \frac{1}{M}\sum_{m=1}^{M}\int_0^1 v_m(t;L,N)\,dt,$$

and

$$\sigma_A(L,M,N) := \sqrt{\frac{1}{M-1}\sum_{m=1}^{M}\left(A_m(L,N) - \mu_A(L,M,N)\right)^2}.$$

Let $\mu_A^{p,s}(L,M,N)$ and $\sigma_A^{p,s}(L,M,N)$ denote the estimated mean and standard deviation of the area under the progress curve for a given solver, s, and a given problem, p. By our scaling of the progress curves, $\mu_A^{p,s}(L,M,N)$ and $\sigma_A^{p,s}(L,M,N)$ should take values in the intervals $[0,1]$ and $[0,1/2]$, respectively, with high probability. The performance of a solver s over a set of problems $\mathcal{P}$ can be depicted in a scatterplot of $\{(\mu_A^{p,s}(L,M,N), \sigma_A^{p,s}(L,M,N)) : p \in \mathcal{P}\}$. Problems for which $(\mu_A^{p}(L,M,N), \sigma_A^{p}(L,M,N))$ lies in the lower left quadrant of $[0,1] \times [0,1/2]$ are those on which the solver makes rapid, reliable progress. Comparing superimposed scatterplots for different solvers can give a rough sense of their relative performance though, when comparing more than a handful of solvers, it may be necessary to produce separate plots.

The appearance of these scatterplots depends on the budget, T. The sample mean $\mu_A^{p,s}(L,M,N)$ should decrease as T increases for solvers that continue to make progress or at least do no worse than the previous time average during the additional time. The effect of T on $\sigma_A^{p,s}(L,M,N)$ is more intricate. Roughly speaking, σ_A is related to the variability in the height of the progress curve. For many solvers, we observe that the interquartile ranges of $f(X(t))$ first increase as the different trajectories of a solver move away from x_0 and later decrease as they converge to x^*. (This is, of course, an oversimplification:

if different trajectories converge to different local optimal solutions, the variance of $f(X(t))$ may remain high for large t.) Thus, increasing the budget should typically introduce extensions of the progress curves whose heights are less variable, translating to less variability in the area under the curve, that is, smaller values of $\sigma_A^{p,s}(L,M,N)$, but this is not universally the case. The choice of budget can also influence the appearance of the scatterplot as a result of solver characteristics. For example, a scatterplot depicting two solvers appears quite different at different budgets T if one solver requires more setup than another but benefits from that setup in the long run.

5. Solvability Profiles

Another way to compare solvers on multiple problems is through profiling. In this section, we explore related ideas for simulation optimization, in which the variable performance of a solver across macroreplications must also be addressed.

5.1. A Single Problem–Solver Pair

We previously define the progress curve as a stochastic process $\boldsymbol{\nu} = \{\nu(t) : t \in [0,1]\}$, where $\nu(t) := (f(X(t)) - f(x^*))/(f(x_0) - f(x^*))$ reports the optimality gap as a function of time. Let $\tau(\alpha)$ be the (random) time required to reduce the optimality gap to a fraction $\alpha \in [0,1]$ of its initial value, that is,

$$\begin{aligned}\tau(\alpha) &:= \inf\{t \in [0,1] : \nu(t) \le \alpha\} \\ &= \inf\{t \in [0,1] : f(X(t)) \le \alpha(f(x_0) - f(x^*)) + f(x^*)\},\end{aligned}$$

where the second equality only applies for a minimization problem. (A similar second equality applies for a maximization problem.) We take the infimum of the empty set to be ∞, so $\tau(\alpha)$ takes values in $[0,1] \cup \{\infty\}$. (This is still true even if $\nu(t)$ can take values outside of $[0,1]$, e.g., if the reference solution x^* is suboptimal and a solver finds a strictly better solution.) We refer to $\tau(\alpha)$ as the α-solve time. The corresponding stochastic process $\boldsymbol{\tau} = \{\tau(\alpha) : \alpha \in [0,1]\}$ can be thought of as the inverse of $\boldsymbol{\nu}$ though this is not exact because $\boldsymbol{\nu}$ need not be monotone.

The choice of α reflects a user's preferred reduction in the initial optimality gap. Values of α such as 0.5 or 0.3 reflect a relatively modest improvement, whereas values such as $\alpha = 0.05$ represent a quite strict requirement. When no confusion can arise, we fix α and suppress it in the notation. As with the area under the progress curve, the budget T plays a significant role in determining the solve time τ.

A plot of the cumulative distribution function of τ yields detailed information about how rapidly and reliably a single solver α-solves a single problem. Summary statistics might be useful, but because τ is extended-valued, moments are typically infinite. Instead, we look at β-quantiles of τ defined as $\pi = \pi_\beta := \inf\{q : \Pr\{\tau \le q\} \ge \beta\}$. Assuming we can exactly compute f, we can estimate π by the sample quantile over M macroreplications,

$$\pi(M) := \inf\left\{q : \frac{1}{M}\sum_{m=1}^{M} \mathbb{I}(\tau_m \le q) \ge \beta\right\},$$

where $\tau_m = \inf\{t \in [0,1] : \nu_m(t) \le \alpha\}$ is the α-solve time from the mth macroreplication. In other words, $\pi(M)$ is the smallest time at which at least a fraction β of the macroreplications α-solve the problem. This quantity can be extended-valued, particularly when α is small. If f cannot be computed exactly, then we use the sample β quantile of $(\tau_m(L,N) : m = 1,2,\ldots,M)$, where $\tau_m(L,N) = \inf\{t \in [0,1] : \nu_m(t;L,N) \le \alpha\}$ is the estimated α-solve time from the mth macroreplication. Perhaps reasonable values of β are 0.5 or 0.9. The corresponding estimators, denoted by $\pi_{0.5}(L,M,N)$ and $\pi_{0.9}(L,M,N)$, represent the median and "fairly sure" fractions of the budget required to α-solve the problem with 0.9 being the stricter requirement. Bootstrapping can be used to provide error estimates for these estimators.

5.2. Multiple Solvers and Problems

Let $\tau^{p,s}$ denote the α-solve time of solver s on problem p given some fixed $\alpha \in [0,1]$ and problem-specific budget T^p. Consider the average probability that solver s solves problem p within a fraction $t \in [0,1]$ of its budget, averaged across a set of problems $p \in \mathcal{P}$, that is,

$$\rho^s(t) := \frac{1}{|\mathcal{P}|}\sum_{p\in\mathcal{P}} \Pr\{\tau^{p,s} \le t\}.$$

We call $\boldsymbol{\rho}^s = \{\rho^s(t) : t \in [0,1]\}$ the *cdf*-solvability profile of solver s. (Here, "cdf" stands for cumulative distribution function and reflects the fact that $\boldsymbol{\rho}^s$ is an average of the cdfs of $\tau^{p,s}$ over problems.) Notice that $\rho^s(1) < 1$ if solver s cannot solve all problems $p \in \mathcal{P}$ within their budgets with probability one. Assuming we can compute f exactly, we can estimate $\rho^s(t)$ for $t \in [0,1]$ by the sample proportion from M i.i.d. macroreplications of solver s on each problem $p \in \mathcal{P}$:

$$\rho^s(t;M) := \frac{1}{|\mathcal{P}|}\sum_{p\in\mathcal{P}} \frac{1}{M}\sum_{m=1}^{M} \mathbb{I}(\tau_m^{p,s} \le t).$$

If we cannot compute f exactly, then we replace $\tau_m^{p,s}$ by its two-level estimate $\tau_m^{p,s}(L,N)$ to obtain the estimator $\rho^s(t;L,M,N)$.

The cdf-solvability profile of a solver at time $t \in [0,1]$ gives the probability that a problem, selected uniformly at random from $\mathcal{P}$, is α-solved by time t on a single macroreplication of the solver. A different form of solvability profile returns the fraction of problems in $\mathcal{P}$ that a given solver α-solves by time t with probability exceeding β. More precisely, we define

$$\rho_\beta^s(t) := \frac{1}{|\mathcal{P}|}\sum_{p\in\mathcal{P}} \mathbb{I}(\Pr\{\tau^{p,s} \le t\} \ge \beta) = \frac{1}{|\mathcal{P}|}\sum_{p\in\mathcal{P}} \mathbb{I}(\pi_\beta^{p,s} \le t),$$

and call $\boldsymbol{\rho}^s_\beta = \{\rho^s_\beta(t) : t \in [0,1]\}$ the β-quantile-α-solvability profile of solver s. Here, $\pi^{p,s}_\beta$ is the β-quantile of the α-solve time $\tau^{p,s}$ of solver s on problem p. For $\beta = 0.9$, for example, $\rho^s_\beta(t)$ gives the fraction of problems we are fairly sure solver s α-solves by time t. Natural one- and two-level estimators of $\rho^s_\beta(t)$ are

$$\rho^s_\beta(t;M) := \frac{1}{|\mathcal{P}|}\sum_{p\in\mathcal{P}} \mathbb{I}(\pi^{p,s}_\beta(M) \le t) \text{ and}$$

$$\rho^s_\beta(t;L,M,N) := \frac{1}{|\mathcal{P}|}\sum_{p\in\mathcal{P}} \mathbb{I}(\pi^{p,s}_\beta(L,M,N) \le t),$$

respectively. Quantile-solvability profiles are perhaps more intuitive than cdf-solvability profiles because they depict a fraction of problems as opposed to a fraction of macroreplications. They summarize the progress curves of problem–solver pairs based on quantiles at time t, thereby diminishing the effect of macroreplications with particularly poor or strong performance. Because quantile-solvability profiles are piecewise-constant and increasing in t with jumps of size $|\mathcal{P}|^{-1}$, they are coarsely quantized for small problem sets.

Because $\rho^s(t;M)$ and $\rho^s_\beta(t;M)$ are bounded, their second moments are finite, and hence, we can obtain a confidence interval for each $t \in [0,1]$ using asymptotic normality, bootstrapping, or other mechanisms for bounded random variables (Diouf and Dufour 2005, Learned-Miller and Thomas 2019). As for $\rho^s(t;L,M,N)$ and $\rho^s_\beta(t;L,M,N)$, one can construct confidence intervals using the bootstrapping procedure outlined in Online Appendix C.

The (estimated) cdf- or quantile-solvability profiles for a set of solvers can be plotted on the same graph with higher curves indicating better performance on the set of problems. Whereas area-under-the-progress-curve scatterplots summarize the overall performance of solvers for different problems, solvability profiles provide comparisons of the solvers' performance at different times, aggregated over all problems. As a result, solvability profiles provide insight about the rate of progress for different solvers.

5.3. Difference Profiles

So far, we have discussed how to compute solvability profiles for each solver in a set of solvers, $\mathcal{S}$. However, when comparing any two solvers, sharper comparisons can be obtained through paired differences. As we see, a difference plot shows how the performance of each solver compares with that of a fixed benchmark solver, s_0. The benchmark solver s_0 can be one that has exhibited robust performance across a range of problems, such as the Nelder–Mead algorithm as tested in Dong et al. (2017), or it can be a newly proposed solver. The benchmark solver can also be used to determine a reasonable budget for each problem $p \in \mathcal{P}$ by running s_0 until an acceptable optimality gap in the problem is achieved.

For solver s, define

$$\delta^s(t) := \rho^s(t) - \rho^{s_0}(t) \text{ for } t \in [0,1],$$

the difference between the cdf-solvability profiles of solvers s and s_0 at time t. The quantity $\delta^s(t)$ is deterministic, ranges between –1 and 1, and is similar to the continuously ranked probability score (Matheson and Winkler 1976) between two distributions. We define the *cdf-solvability difference profile*, henceforth cdf-difference profile, of solver s as $\boldsymbol{\delta}^s = \{\delta^s(t) : t \in [0,1]\}$. The cdf-difference profile represents the difference between the probabilities of solvers s and s_0 solving a problem chosen uniformly at random from $\mathcal{P}$ within a fraction $t \in [0,1]$ of its associated budget. An analogous definition yields the β-quantile-α-solvability difference profile or, in short, quantile-difference profile of solver s: $\boldsymbol{\delta}^s_\beta = \{\delta^s_\beta(t) : t \in [0,1]\}$, where $\delta^s_\beta(t) := \rho^s_\beta(t) - \rho^{s_0}_\beta(t)$ for $t \in [0,1]$. If f can be computed exactly, we can estimate $\delta^s(t)$ and $\delta^s_\beta(t)$ from M i.i.d. macroreplications by $\delta^s(t;M) := \rho^s(t;M) - \rho^{s_0}(t;M)$ and $\delta^s_\beta(t;M) := \rho^s_\beta(t;M) - \rho^{s_0}_\beta(t;M)$, respectively. When f cannot be computed exactly, the corresponding two-level estimators are given by $\delta^s(t;L,M,N) := \rho^s(t;L,M,N) - \rho^{s_0}(t;L,M,N)$ and $\delta^s_\beta(t;L,M,N) := \rho^s_\beta(t;L,M,N) - \rho^{s_0}_\beta(t;L,M,N)$.

The ordering of solvers in difference profiles is the same as that of the solvability profiles, but comparisons with the benchmark s_0 are accentuated. Moreover, difference profiles can take advantage of CRN, much as one can use paired difference estimators to estimate a difference of means in classical statistics. Difference profiles for multiple solvers can be exhibited in a single plot by pairing all solvers against a benchmark solver s_0. In such a plot, solver s overperforms (underperforms) solver s_0 at a time $t \in [0,1]$ if the difference profile lies above (below) zero at time t. Pointwise confidence intervals can be constructed via bootstrapping or other methods mentioned in the previous section.

6. Budget-Specific Solvers

Our focus to this point has been on solvers that do not explicitly consider the budget T in setting their parameters. Such budget-agnostic solvers can report estimated best solutions at intermediate budgets as they proceed on a single macroreplication. On the other hand, measuring and interpreting the progress of budget-specific solvers is more nuanced. One can run a budget-specific solver with a single budget T on a given problem and, for any fixed t, define $X(t)$ to be the (random) recommended solution upon expending an intermediate budget tT. This approach yields a stochastic process $X_T = \{X_T(t) : 0 \le t \le 1\}$, where the subscript T explicitly indicates the dependence on the given budget. One can then generate the performance measures we define for budget-agnostic solvers as before. This approach allows

time-dependent comparisons with budget-agnostic solvers, but it is not entirely satisfactory because the stochastic process X_T is specific to the budget T. Consequently, under this approach, the progress of a budget-specific solver at an intermediate budget tT no longer signifies how the solver performs if given a budget of tT at the outset.

It may appear that these issues can be resolved by running separate macroreplications of a budget-specific solver at multiple intermediate budgets $T_i = t_i T$, $i = 1, 2, \ldots, k$, and defining each $X(t_i)$ as the solution recommended at termination. However, even though the marginal distributions of $X(t_i)$ are well-defined, it is unclear how to precisely characterize the joint distribution of $X(t_1), X(t_2), \ldots, X(t_k)$ for budget-specific solvers. This joint distribution depends, for example, on whether one uses CRN across the macroreplications run with different budgets. Moreover, if the solver uses an internal source of randomness, then how is that source coordinated across those same macroreplications? The fact that this joint distribution is not well-defined for budget-specific solvers affects several of our performance measures. Progress curves, which depict the scaled value of $f(X(t))$ as a function of time over a single macroreplication, are no longer well-defined because they rely on the joint behavior of $f(X(t))$ at multiple values of t. Likewise, neither the time-average optimality gap A nor the solve time $\tau(\alpha)$ are well-defined, meaning that area scatterplots and solvability profiles are not well-defined.

Nevertheless, the progress at time t, $\nu(t)$, as well as its mean $\mu(t)$ and quantile $\chi(t)$, are well-defined because they rely on the distribution of $f(X(t))$ for only a single t. One can, therefore, still run multiple macroreplications of budget-specific solvers at multiple budgets but redefine our performance measures to depend on the marginal distributions of $f(X(t_i))$ for $i = 1, 2, \ldots, k$. For example, one can estimate the mean progress and quantile progress at times $t_1, t_2, \ldots, t_k$ and produce estimated mean and quantile progress curves. Both the appearance of these curves—which now have jumps at a common set of breakpoints—and their interpretation differ from those of budget-agnostic solvers. Area scatterplots are harder to generalize because they depend on the joint distribution of the heights of progress curves at multiple time points, and this joint distribution is not well-defined. Generalizations are possible but seem hard to justify in a principled manner. This approach of specifying a finite set of times on which to focus, thus, has fundamental difficulties, and even if we ignore those difficulties, the chosen times are necessarily arbitrary. Moreover, from a practical standpoint, this approach is computationally very demanding; multiple macroreplications are required at multiple budgets, in contrast to budget-agnostic solvers that do not require new macroreplications at each budget. Given these flaws, we elect not to pursue this approach either.

The approach we instead advocate is to evaluate a budget-specific solver at a single budget tT through the distribution of $f(X(t))$, the true objective function value at the random recommended solution $X(t)$, scaling to $\nu(t) = [f(X(t)) - f(x^*)]/[f(x_0) - f(x^*)]$ to improve interpretability. The natural choice of a single budget is T, which arises when $t = 1$. We can then compare both budget-specific and budget-agnostic solvers at this budget (for a single problem) through distributional properties of $\nu(1)$. To summarize the results of multiple solvers tackling a single problem, we use so-called terminal-progress comparative violin plots (or boxplots) of $\nu(1)$, plotting one distribution per solver. With multiple solvers and problems, one might still use comparative violin plots or boxplots, but with each violin (box) representing a mixture distribution of a single solver's performance across multiple problems. Such plots are likely hard to interpret, however, so we instead recommend a scatterplot in which each problem–solver combination is represented by a single point with coordinates given by the mean and standard deviation of $\nu(1)$ for that problem–solver combination. We refer to these plots as terminal progress scatterplots.

7. Examples

We present examples of the aforementioned plots for an experiment conducted with problems and solvers from the SimOpt testbed (Eckman et al. 2020). Our problem set consists of 20 instances of three problems—`SSCONT-1`, `IRONORECONT-1`, and `SAN-1`—for a total of 60 problem instances. `SSCONT-1` is an (s, S) inventory problem with continuous demand and order quantities in which the objective is to identify the reorder and order-up-to levels that minimize the expected per-period total cost: the sum of back-order, order, and holding costs. `IRONORECONT-1` is another production-planning problem in which a manufacturer of iron ore seeks to determine the price levels at which to start and stop production or sell all stock so as to maximize expected profit. `SAN-1` is a stochastic activity network problem in which the objective is to choose the mean task durations in the network that minimize the expected length of the longest path plus a penalty paid for reducing the mean task durations. The dimensions of the three problems are 2, 3, and 13, respectively, and all problems are formulated with continuous decision variables and nonnegativity constraints. More detailed problem descriptions can be found in the documentation for SimOpt (Eckman et al. 2020), and Online Appendix A lists specific parameter settings we used. All code and data for these experiments can be found in an accompanying GitHub repository (Eckman et al. 2022a).

We test four classes of solvers:

`Random Search` randomly samples solutions from the feasible region and takes a fixed number of replications at each solution. New solutions are generated until the budget is exhausted. We test two versions of `Random Search` with 10 (`RS10`) and 50 (`RS50`)

replications per solution. See, for example, Chia and Glynn (2013) for basic theory on random search.

ASTRO-DF is a stochastic, derivative-free, trust-region method that uses adaptive sampling at each visited solution and interpolation to build local models (Shashaani et al. 2018).

STRONG is another stochastic, derivative-free, trust-region method that uses design of experiments to sample neighboring solutions and construct a first or second order approximation of the objective function (Chang et al. 2013, Chang 2014).

Nelder-Mead is a heuristic algorithm that maintains a simplex of solutions and performs a variety of geometric operations to alter the structure of the simplex and traverse the feasible region (Nelder and Mead 1965, Barton and Ivey 1996).

All solvers are given a budget of $T = 1,000$ replications for each instance of SSCONT-1 and IRONORECONT-1 and a budget of $T = 10,000$ for each instance of SAN-1. We run $M = 10$ macroreplications of each solver on each problem instance with all solvers starting from a common initial solution on all macroreplications. In a post-processing stage, we take $N = 100$ postreplications at all recommended solutions. One might consider using different numbers of postreplications for those recommended solutions that may be noisier than others, but using a fixed N for all solutions has its merits, including simplicity and transparency. For each problem instance, the proxy optimal solution, x^*, is taken to be the recommended solution with the best postreplicated estimate $f_N(\cdot)$ over all macroreplications of all solvers. We take $L = 200$ postreplications at x_0 and x^* for each problem instance. Ideally, the values M, N, and L would all be chosen very large to ensure smooth plots with small uncertainty ranges (as indicated by the shading in the plots from bootstrapping). Computational considerations necessitate compromise. Our choices are simply plausible values that yield reasonable results, and certainly other choices could be made. In general, all three parameters must be increased to reduce the uncertainty ranges in the plots; see Section 8.

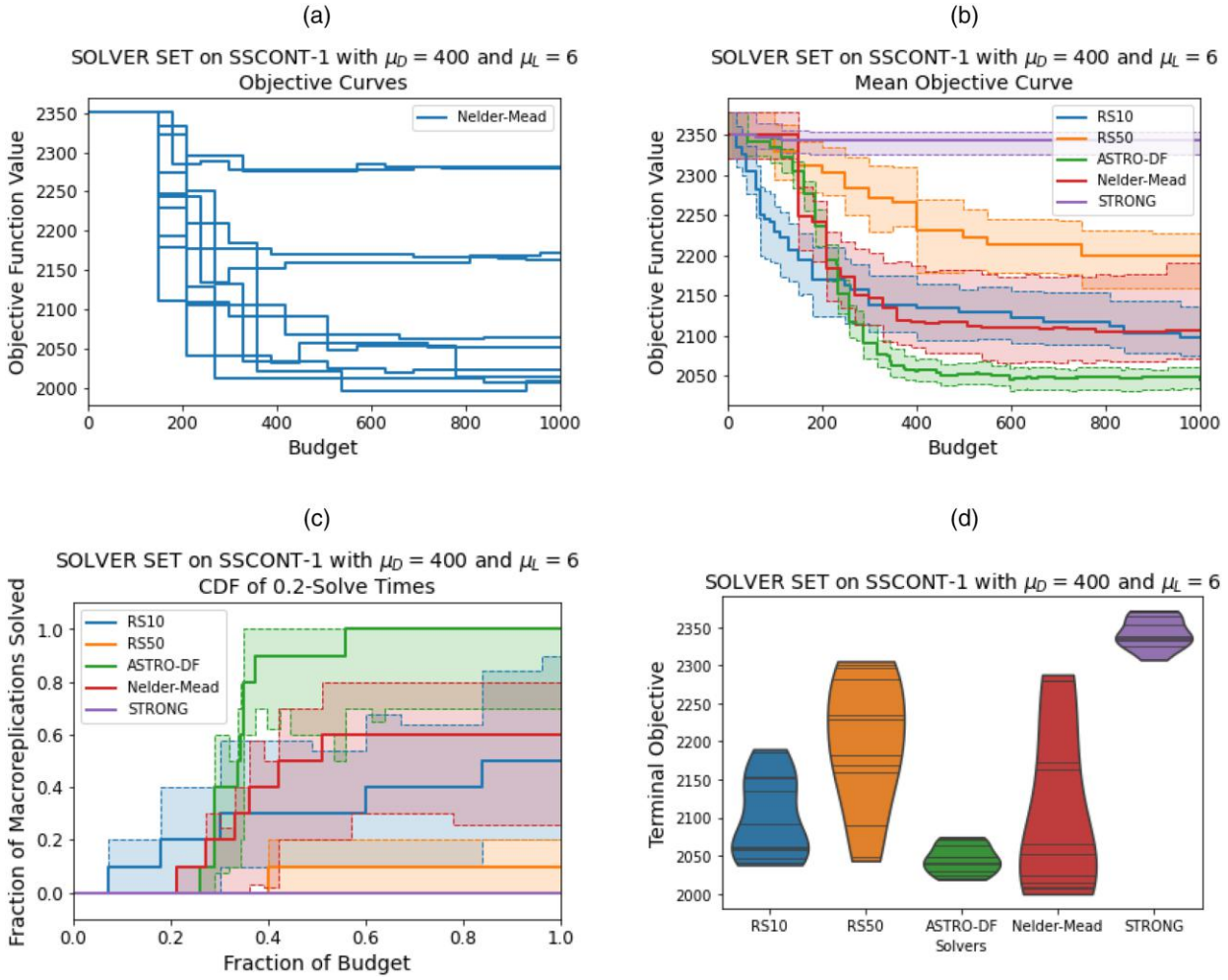
In Figure 1, we present plots for a problem instance of SSCONT-1 with a mean demand of $\mu_D = 400$ units and a mean lead time of $\mu_L = 6$ days. Because the comparison is made over a single problem instance, one can use the unnormalized progress curves for interpretability. The confidence intervals in these plots, constructed via bootstrapping, depict the error in estimating solver performance on this problem instance.

Figure 1(a) and (b) shows that progress curves are highly variable because of the high variance of the lead time. The mean progress curves in Figure 1(b) illuminate the average performance of different solvers over time and offer a clear ordering of the solvers based on their empirical performance. Figure 1(c) depicts the cdf of the α-solve times for $\alpha = 0.20$, that is, the first times at which each solver recommends a solution within 20% of optimal (relative to the original optimality gap) on any given macroreplication. Figure 1(c) shows the same ranking of solvers on the problem instance as in Figure 1(b). The terminal-progress violin plots in Figure 1(d) show that ASTRO-DF consistently recommends high-quality solutions upon termination, whereas STRONG struggles to improve upon the initial solution. A possible explanation for the latter is that we implemented a version of STRONG that increases the sample size whenever the estimated gradient is close to zero, which may arise given how the experiments were performed using CRN across solutions. The shape of the violin plots also indicates that, although Nelder-Mead, RS10, and RS50 can recommend competitive solutions, the quality of their final recommendations is more variable.

Next, we examine the tools we propose to compare solvers over a testbed of problems. Area and terminal-progress scatterplots for the 20 instances of SAN-1 appear in Figure 2(a) and (b), respectively, with the corresponding plots for the 20 instances of IRONORECONT-1 appearing in Figure 2(c) and (d). We omit plots for SSCONT-1 and all 60 instances combined for space reasons. We also suppressed the horizontal and vertical bars that indicate bootstrapped confidence intervals to reduce clutter, but they can optionally be added. On SAN-1, there is very clear clustering of solver performances. Both random search solvers fail to make any progress as indicated by the points at the bottom right of the plots. The performance of the other three solvers is similar. ASTRO-DF has more variable time-averaged performance than both Nelder-Mead and STRONG, but its superior terminal progress makes it clearly preferable to STRONG. Both Nelder-Mead and ASTRO-DF are left-shifted in the terminal-progress scatterplot relative to the area scatterplot as one might expect from solvers making steady progress over time. For the problem IRONORECONT-1, STRONG has the best terminal performance among the solvers though all solvers demonstrate mixed performance, suggesting some of these instances are difficult to solve. The variability in the terminal progress for some of the solvers appears to be greater than that of the time-average progress as evidenced by the wider vertical spread of the points in Figure 2(d).

Figure 3(a) and (c), shows cdf- and quantile-solvability profiles of the five solvers over the full set of problem instances. The cdf-solvability profile pertains to the time required to reduce the optimality gap (of any problem) to a fifth of its initial value on each macroreplication of each problem instance. The quantile-solvability profiles are presented for the median performance of solvers; these plots thereby discount the effect of extreme solve times encountered on certain macroreplications. The right endpoints of the cdf- and quantile-solvability profiles show that our problem set includes some hard

Figure 1. (Color online) Results for SSCONT-1 with Mean Demand $\mu_D = 400$ Units and Mean Lead Time $\mu_L = 6$ Days



Notes. (a) Estimated progress curves. (b) Unnormalized mean progress curves. (c) Cdfs of 0.2-solve times. (d) Terminal progress violin plots.

problems that all solvers fail to 0.2-solve within the budget of simulation replications. Of the solvers, RS10 can quickly 0.2-solve about one third of the problem instances, which is to be expected because the cheap sampling of solutions (particularly for low-dimensional problems, such as SSCONT-1 and IRONORECONT-1) allows RS10 to quickly visit improving solutions. RS10's superior initial performance relative to ASTRO-DF quickly disappears as the fraction of the budget increases and ASTRO-DF then dominates. The cdf-solvability and quantile solvability profiles also show that the high-level ordering of solvers changes depending on the fraction of the budget expended. Finally, although Nelder-Mead is slow to 0.2-solve problems, it makes steady progress throughout.

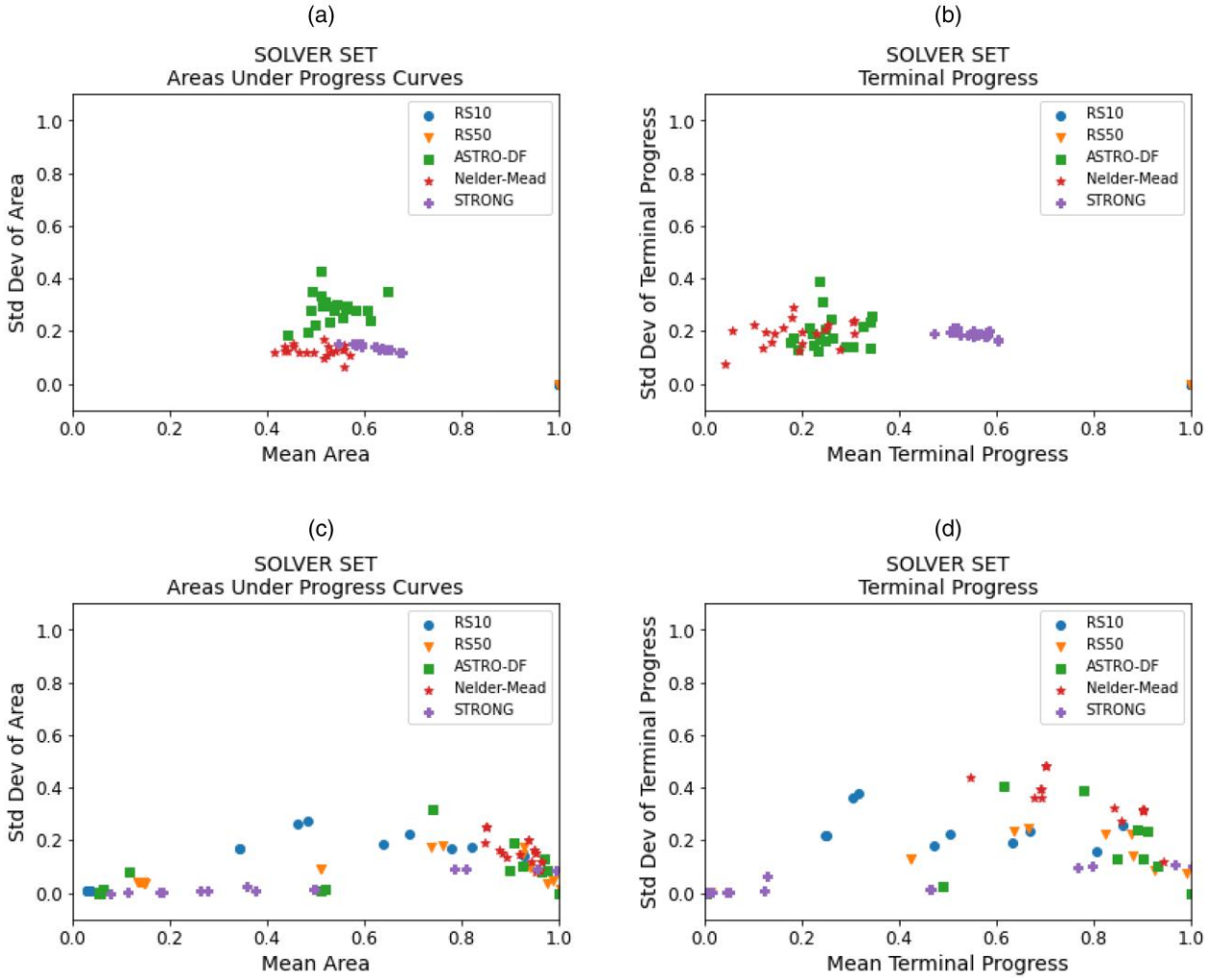
The difference profiles in Figure 3(b) and (d), depict the performance gaps between ASTRO-DF and the other solvers and further clarify the perceived superiority of ASTRO-DF's performance over the problem set. An advantage of the difference profiles is that they exploit CRN, so we see statistically significant differences between the solvers.

8. Convergence of the Estimators

The number of macroreplications, M, and postreplications L, N affects the properties of our estimators. Here, we explore the impact of these parameters on the convergence of the estimators, exploiting the theory of two-level simulation. For simplicity, throughout this section, we fix the time t and consider the pointwise error in the estimated progress curves at that fixed time. Ideally, we would consider multiple values of t simultaneously because we are interested in the entire aggregate progress curve, but such an analysis appears to be quite involved. The analysis for a fixed t shows what we believe to be the main points.

We assume that the postreplications used to estimate $f(X_i(t))$ are independent of those used to estimate $f(X_j(t))$ for $i \neq j$, that is, that the postreplications used to

Figure 2. (Color online) Scatterplots of the Mean and Standard Deviation of Areas Under Estimated Progress Curves ((a) and (c)) and Terminal Progress ((b) and (d)) for `SAN-1` and `IRONORECONT-1`, Respectively



evaluate recommended solutions from different macroreplications are mutually independent. Moreover, we assume that the L postreplications used to estimate $f(x_0)$ and $f(x^*)$ are independent of those used to estimate $f(X_j(t))$, $j = 1, 2, \ldots, M$. We allow the L postreplications at x_0 and x^* to be statistically dependent.

Remark 3. It is not helpful to use CRN in the postreplications used to evaluate solutions from different macroreplications. Doing so creates a dependence across macroreplications that slows down the convergence of the estimators.

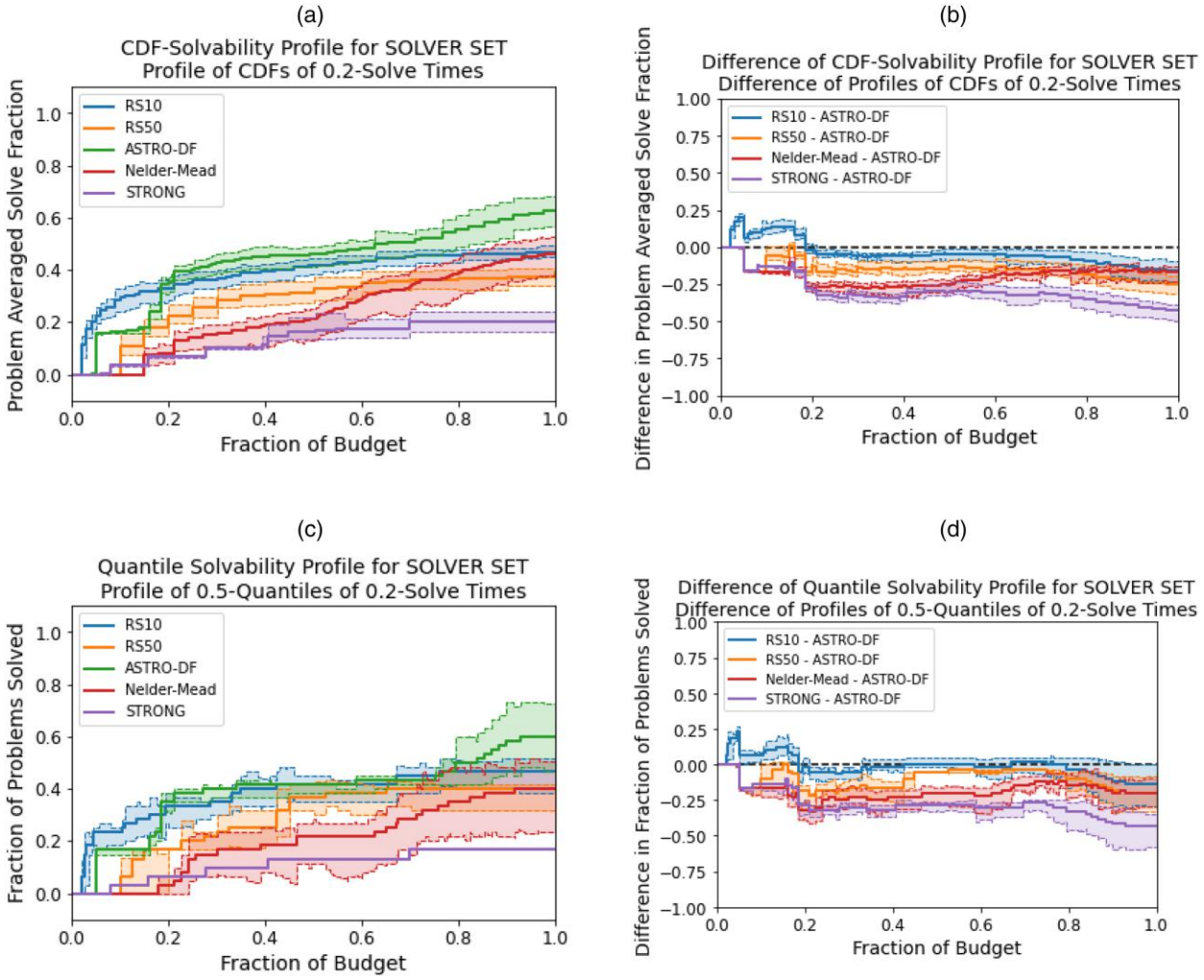
Throughout this section, we assume x^*, the optimal solution, is deterministic and given. In practice it often needs to be estimated. One might extend the ideas presented here under some assumption about the behavior (as a function of L, M, and N) of an estimator of x^* that replaces x^*, but such methodology depends heavily on the nature of f and the manner in which x^* is estimated.

We shall, in some detail, analyze mean progress curves and discuss quantile progress curves to a lesser degree.

8.1. Mean Progress Curves

How accurate is the estimator $\mu(t; L, M, N)$ of $\mu(t)$? We state our observations in terms of an overall computational budget of simulation replications used to run the entire experiment, including all macroreplications and postreplications. We denote this overall budget by c and assume, for simplicity, that the cost of running a replication (likewise postreplication) is uniform in x. Thus, we regard $L = L(c)$, $M = M(c)$, and $N = N(c)$ as functions of c, which we assume are bounded below by one to avoid trivialities, and we suppress the dependence on c for notational simplicity. We first run M macroreplications with per-macroreplication cost (average number of simulation replications) Tt and then complete the postreplications at cost $2L + MN$ replications, yielding the

Figure 3. (Color online) Profiles over All Problem Instances



Notes. (a) Cdf solvability profiles. (b) Difference of cdf solvability profiles. (c) Quantile solvability profiles. (d) Difference of quantile solvability profiles.

estimator $\mu(t;L,M,N)$. Thus, $c = TtM + 2L + MN$, where $Tt \gg 1$ represents the number of simulation replications needed for a single macroreplication out to time Tt, MN is the number of postreplications at recommended solutions $X_1(t), X_2(t), \ldots, X_M(t)$, and L postreplications are spent at each of solutions x_0 and x^*. (For simplicity, we ignore rounding effects associated with the need for L, M, and N to all be integers.)

To proceed, we require additional assumptions.

Assumption 3. *Simulation replications at any solution x are unbiased, that is, $\mathbb{E}Y_1(x) = f(x)$ for all x.*

Assumption 4. *Simulation replications at any solution x have bounded (in x) nonzero variance, that is, $\sigma^2(x) := \mathrm{var}Y_1(x)$ is positive and bounded in x.*

Assumptions 3 and 4 (and Assumption 1 stated earlier) permit a transparent analysis but are restrictive. For example, relaxing Assumption 3 might be useful in the context of steady-state simulation. We expect the conclusions of Theorem 1 to hold under relaxed assumptions, but we do not attempt to relax the assumptions because we do not think additional insight would be obtained.

We say that a family of random variables $X(c)$ is $O_p(h(c))$ if the family $\{X(c)/h(c) : c \geq c_0\}$ is tight for some $c_0 > 0$. The proof of the following result appears in Online Appendix B.

Theorem 1. *Suppose that Assumptions 1–4 hold and $\min\{L,M\} \to \infty$ as $c \to \infty$. Then, $\mu(t;L,M,N) \to \mu(t)$ in probability as $c \to \infty$. Moreover,*

$$\mu(t;L,M,N) = \tilde{\mu}(t;L,M,N) + O_p(L^{-1} + M^{-1}),$$

where the random variable $\tilde{\mu}(t;L,M,N)$ has mean $\mu(t)$ and variance

$$\varsigma^2(L,M,N) = \frac{a_1}{MN} + \frac{a_2}{M} + \frac{a_3}{L},$$

for appropriate constants a_1, a_2 and a_3.

Recall that $c = TtM + MN + 2L$, representing the total number of simulation replications. For large budgets c, the variance of $\tilde{\mu}(t;L,M,N)$ is minimized by taking the number of macroreplications M to be linear in c, the number of postreplications L to be linear in c, and the number of postreplications N to be constant in c as can be derived by standard calculus arguments that relax the constraint that these quantities be integers. In that case, the variance of $\tilde{\mu}(t;L,M,N)$ is of order c^{-1}, which is the usual canonical rate in Monte Carlo, and the need for two-level simulation does not result in a deterioration in the convergence rate.

The conclusion that the estimator $\mu(t;L,M,N)$ converges at the canonical rate (asymptotic variance of order c^{-1}) when L, M, and N are chosen appropriately is in line with the observations in Sun et al. (2011) for two-level simulations. The purpose of Theorem 1 is not to help identify optimal choices of the parameters L, M, and N because such choices depend on parameters that are difficult to compute. Rather, we present the result to emphasize the convergence rate.

One might be tempted to use Theorem 1 to develop confidence intervals on values of the mean progress curve. Doing so requires developing estimators of the various constants appearing in the result. It seems to be far more practical to use bootstrapping to obtain error estimates as discussed in Online Appendix C.

8.2. Quantile Progress Curves

The quantile progress estimator $\chi(t;L,M,N)$ can be analyzed using techniques similar to those we use for $\mu(t;L,M,N)$. However, quantile estimation for two-level simulation poses an additional challenge. Quantile estimators are analyzed in Lee (1998), Lee and Glynn (2003), and Gordy and Juneja (2010), in which asymptotic theory is developed in the case in which the number of postreplications N is the same at all macroreplication solutions. It is natural, however, not to seek high accuracy in estimating the true objective function value of recommended solutions with relative optimality gaps that are far from that of the true quantile $\chi_\beta(t)$. Gordy and Juneja (2010) and Broadie et al. (2011) exploit this observation, analyzing estimators that carefully vary the second level sample sizes, achieving a faster convergence rate than the common-N estimator. Extensions are explored in Broadie et al. (2015) and Hong et al. (2017). In what follows, we adopt a common number of postreplications, N, for all macroreplication solutions.

To rigorously state convergence results for $\chi(t;L,M,N)$, which is a quantile estimator using two-level simulation, requires a great deal of associated notation and regularity assumptions as is clear from Lee (1998) and Gordy and Juneja (2010). Accordingly, we choose not to state such results, but rather indicate what one can expect in general in our setting given the results in the aforementioned literature.

First, in the case when the solution space is discrete, the results in Lee and Glynn (2003) indicate that the mean-squared error of the estimator $\chi(t;L,M,N)$ is typically minimized when the number of postreplications L is of order c, the number of macroreplications M is of order c and the number of postreplications N is of order $\ln c$, in which case the mean squared error is of order $\ln c/c$. This is slower than the canonical rate $1/c$, but only by a logarithmic factor. Second, in the case when the solution space is continuous, the mean squared error of the estimator $\chi(t;L,M,N)$ is typically minimized when the number of postreplications L is of order $c^{2/3}$, the number of macroreplications M is of order $c^{2/3}$, and the number of postreplications N is of order $c^{1/3}$, in which case the mean squared error is of order $c^{-2/3}$; see section 3.1.2 of Lee (1998).

9. Conclusions

Current practice in computational comparisons of SO solvers is inconsistent. We provide a two-level experimental design consisting of macroreplications and postreplications with systematic CRN implementation and error estimation. Postreplications to remove optimization bias are rarely performed in existing literature.

We develop and demonstrate a range of plots for use in empirical evaluation of SO solvers on a testbed of problems. Progress curves are closely related to curves that have frequently been used to date, indicating the objective function value of the most recently recommended solution as a function of time, but when progress curves appear, they are usually based only on the original macroreplications, not on postreplications. Moreover, our plots differ in the way they are scaled with the x-axis reflecting the fraction of the computational budget expended and the y-axis reflecting the fraction of the initial optimality gap that remains. Progress curves and closely related plots giving the cdf of the α-solve time for varying α provide a great deal of information about the performance of a single solver operating on a single problem. Yet these plots are less useful when one wishes to explore the performance of a solver on multiple problems or to compare the performance of multiple solvers on multiple problems. Area scatterplots and solvability profiles can prove useful in this more information-rich setting by providing a high-level view of overall performance.

We provide some examples of these plots that clarify both their nature and usefulness. We believe these examples provide a "proof of concept" that demonstrates the potential in such comparisons. The plots generated here were obtained using the very recently upgraded SimOpt testbed (Eckman et al. 2020), which is now available for general use. The new version of SimOpt was designed to be useful not just in the simulation-optimization setting we explore here, but also in other settings such as in data

farming (Eckman et al. 2022c). Those design improvements will be reported elsewhere.

Acknowledgments

The authors thank the reviewers, associate editor, and area editor for their suggestions for improving the paper.

References

Ali MM, Khompatraporn C, Zabinsky ZB (2005) A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. *J. Global Optim.* 31(4):635–672.

Amaran S, Sahinidis NV, Sharda B, Bury SJ (2016) Simulation optimization: A review of algorithms and applications. *Ann. Oper. Res.* 240(1):351–380.

Andradóttir S (2006) An overview of simulation optimization via random search. Henderson SG, Nelson BL, eds. *Simulation*, Handbooks in Operations Research and Management Science, vol. 13 (Elsevier, North Holland), 617–631.

Andradóttir S (2015) A review of random search methods. Fu MC, ed. *Handbook of Simulation Optimization*, International Series in Operations Research & Management Science, vol. 216 (Springer, New York), 277–292.

Asmussen S, Glynn PW (2007) *Stochastic Simulation: Algorithms and Analysis*, Stochastic Modeling and Applied Probability, vol. 57 (Springer, New York).

Barton RR, Ivey JS Jr (1996) Nelder-Mead simplex modifications for simulation optimization. *Management Sci.* 42(7):954–973.

Bayraksan G, Morton DP (2006) Assessing solution quality in stochastic programs. *Math. Programming* 108(2):495–514.

Bayraksan G, Morton DP (2009) Assessing solution quality in stochastic programs via sampling. Informs TutORials in Operations Research, 102–122.

Beiranvand V, Hare W, Lucet Y (2017) Best practices for comparing optimization algorithms. *Optim. Engrg.* 18(4):815–848.

Broadie M, Du Y, Moallemi CC (2011) Efficient risk estimation via nested sequential simulation. *Management Sci.* 57(6):1172–1194.

Broadie M, Du Y, Moallemi CC (2015) Risk estimation via regression. *Oper. Res.* 63(5):1077–1097.

Chang KH (2014) Improving the efficiency and efficacy of stochastic trust-region response-surface method for simulation optimization. *IEEE Trans. Automatic Control* 60(5):1235–1243.

Chang KH, Hong LJ, Wan H (2013) Stochastic trust-region response-surface method (STRONG)—A new response-surface framework for simulation optimization. *INFORMS J. Comput.* 25(2):230–243.

Chau M, Fu MC (2015) An overview of stochastic approximation. Fu MC, ed. *Handbook of Simulation Optimization*, International Series in Operations Research & Management Science, vol. 216 (Springer, New York), 149–178.

Chia YL, Glynn PW (2013) Limit theorems for simulation-based optimization via random search. *ACM Trans. Model. Comput. Simulation* 23(3):1–18.

Cooper K, Hunter SR, Nagaraj K (2020) Biobjective simulation optimization on integer lattices using the epsilon-constraint method in retrospective approximation framework. *INFORMS J. Comput.* 32(4):1080–1100.

Diouf MA, Dufour JM (2005) Improved nonparametric inference for the mean of a bounded random variable with application to poverty measures. Technical report, Université de Montréal, QC, Canada.

Dolan ED, Moré JJ (2002) Benchmarking optimization software with performance profiles. *Math. Programming* 91(2):201–213.

Dong N, Eckman DJ, Zhao X, Poloczek M, Henderson SG (2017) Empirically comparing the finite-time performance of simulation-optimization algorithms. Chan WKV, D'Ambrogio A, Zacharewicz G, Mustafee N, Wainer G, Page E, eds. *Proc. 2017 Winter Simulation Conf.* (Institute of Electrical and Electronics Engineers, Inc., Piscataway, NJ), 2206–2217.

Eckman DJ, Henderson SG, Pasupathy R (2019) Redesigning a testbed of simulation-optimization problems and solvers for experimental comparisons. Mustafee N, Bae KHG, Lazarova-Molnar S, Rabe M, Szabo C, Haas P, Son YJ, eds. *Proc. 2019 Winter Simulation Conf.* (Institute of Electrical and Electronics Engineers, Inc., Piscataway, NJ), 3457–3467.

Eckman DJ, Henderson SG, Shashaani S (2022a) Diagnostic tools for evaluating and comparing simulation-optimization algorithms. Accessed November 18, 2022, http://dx.doi.org/10.5281/zenodo.7329235.

Eckman DJ, Henderson SG, Shashaani S (2022b) SimOpt: A testbed for simulation-optimization experiments.

Eckman DJ, Shashaani S, Sanchez SM (2022c) Data farming for simulation optimization. Working paper, Texas A&M University, College Station, TX.

Eckman DJ, Henderson SG, Shashaani S, Pasupathy R (2020) Simulation optimization library. Accessed May 1, 2020, http://github.com/simopt-admin/simopt.

Fu MC (2002) Optimization for simulation: Theory vs. practice. *INFORMS J. Comput.* 14(3):192–215.

Fu MC (2006) Gradient estimation. Henderson SG, Nelson BL, eds. *Simulation, Handbooks in Operations Research and Management Science* (Elsevier, North Holland), 575–616.

Ghadimi S, Lan G (2015) Stochastic approximation methods and their finite-time convergence properties. Fu MC, ed. *Handbook of Simulation Optimization*, International Series in Operations Research & Management Science, vol. 216 (Springer, New York), 179–206.

Glynn PW (2002) Additional perspectives on simulation for optimization. *INFORMS J. Comput.* 14(3):220–222.

Gordy MB, Juneja S (2010) Nested simulation in portfolio risk measurement. *Management Sci.* 56(10):1833–1848.

Gould N, Scott J (2016) A note on performance profiles for benchmarking software. *ACM Trans. Math. Software* 43(2):1–5.

Gould NI, Orban D, Toint PL (2015) CUTEst: A constrained and unconstrained testing environment with safe threads for mathematical optimization. *Comput. Optim. Appl.* 60(3):545–557.

Homem-de Mello T, Bayraksan G (2014) Monte Carlo sampling-based methods for stochastic optimization. *Surveys Oper. Res. Management Sci.* 19(1):56–85.

Hong LJ, Juneja S, Liu G (2017) Kernel smoothing for nested estimation with application to portfolio risk measurement. *Oper. Res.* 65(3):657–673.

Kushner HJ, Yin GG (2003) *Stochastic Approximation and Recursive Algorithms and Applications*, 2nd ed. (Springer-Verlag, New York).

Lan G, Nemirovski A, Shapiro A (2012) Validation analysis of mirror descent stochastic approximation method. *Math. Programming* 134(2):425–458.

Learned-Miller E, Thomas PS (2019) A new confidence interval for the mean of a bounded random variable. Preprint, submitted May 15, https://arxiv.org/abs/1905.06208v2.

Lee SH (1998) Monte Carlo computation of conditional expectation quantiles. Unpublished PhD thesis, Stanford University, Stanford, CA.

Lee SH, Glynn PW (2003) Computing the distribution function of a conditional expectation via Monte Carlo: Discrete conditioning spaces. *ACM Trans. Model. Comput. Simulation* 13(3):238–258.

Li J, Ryzhov IO (2022) Convergence rates of epsilon-greedy global optimization under radial basis function interpolation. *Stochastic Systems*, ePub ahead of print August 2, https://doi.org/10.1287/stsy.2022.0096.

Mak WK, Morton DP, Wood RK (1999) Monte Carlo bounding techniques for determining solution quality in stochastic programs. *Oper. Res. Lett.* 24(1):47–56.

Matheson JE, Winkler RL (1976) Scoring rules for continuous probability distributions. *Management Sci.* 22(10):1087–1096.

Morales JL (2002) A numerical study of limited memory BFGS methods. *Appl. Math. Lett.* 15(4):481–487.

Moré JJ, Wild SM (2009) Benchmarking derivative-free optimization algorithms. *SIAM J. Optim.* 20(1):172–191.

Nelder JA, Mead R (1965) A simplex method for function minimization. *Comput. J.* 7(4):308–313.

Nemirovski A, Juditsky A, Lan G, Shapiro A (2009) Robust stochastic approximation approach to stochastic programming. *SIAM J. Optim.* 19(4):1574–1609.

Netlib (2021) Accessed March 3, 2021, http://netlib.org.

Pasupathy R, Henderson SG (2006) A testbed of simulation-optimization problems. Perrone LF, Wieland FP, Liu J, Lawson BG, Nicol DM, Fujimoto RM, eds. *Proc. 2006 Winter Simulation Conf.* (Institute of Electrical and Electronics Engineers, Inc., Piscataway, NJ), 255–263.

Pasupathy R, Henderson SG (2011) SimOpt: A library of simulation optimization problems. Jain S, Creasey RR, Himmelspach J, White KP, Fu M, eds. *Proc. 2011 Winter Simulation Conf.*, 4075–4085 (Institute of Electrical and Electronics Engineers, Inc., Piscataway, NJ).

Rosenbrock HH (1960) An automatic method for finding the greatest or least value of a function. *Comput. J.* 3(3): 175–184.

Shashaani S, Hashemi FS, Pasupathy R (2018) ASTRO-DF: A class of adaptive sampling trust-region algorithms for derivative-free stochastic optimization. *SIAM J. Optim.* 28(4):3145–3176.

Shi HJM, Xuan MQ, Oztoprak F, Nocedal J (2021) On the numerical performance of derivative-free optimization methods based on finite-difference approximations. Preprint, submitted February 19, https://arxiv.org/abs/arXiv:2102.09762v1.

Sun Y, Apley DW, Staum J (2011) Efficient nested simulation for estimating the variance of a conditional expectation. *Oper. Res.* 59(4):998–1007.

Wikipedia (2021) Test functions for optimization. Accessed March 3, 2021, https://en.wikipedia.org/wiki/Test_functions_for_optimization.

Wild S (2019) Personal communication with authors, October.