Evaluating and Comparing Simulation-Optimization Algorithms

David J. Eckman
Wm Barnes '64 Department of Industrial and Systems Engineering
College Station, Texas 77843

Shane G. Henderson
School of Operations Research and Information Engineering
Cornell University
Ithaca, New York 14853

Sara Shashaani
Edward P. Fitts Department of Industrial and Systems Engineering
North Carolina State University
Raleigh, North Carolina 27695

Abstract

Simulation optimization involves the optimization of some objective function that can only be estimated via stochastic simulation. Many important problems can be profitably viewed within this framework. While 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, that can be used to evaluate and compare solvers on a testbed of simulation-optimization problems. We explain the need for two-level simulation in this context and provide supporting theory to guide the use of common random numbers to achieve efficient comparisons. We also describe how to use bootstrapping to obtain error estimates for the estimators.

1 Introduction and Motivation

Simulation optimization (SO) involves the optimization of some objective function over a (possibly constrained) feasible region, where 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. Moreover, many SO problems lack structure such as smoothness that might be exploited by specialized *solvers*, i.e., implementations of algorithms for solving SO problems.

The development of SO solvers is an active area of research. Much effort has been devoted to the design of solvers with provable convergence guarantees, whether to local or global solutions, e.g., Kushner and Yin [2003], Andradóttir [2006, 2015], Cooper et al. [2020] and Li and Ryzhov [2020]. Such convergence analyses are valuable and insightful, yet are typically most relevant in an asymptotic regime where the computational effort becomes very large, especially in the case of results for global optimization. Since that regime can be difficult to reach in practice, there is a need to better understand the pre-asymptotic regime, where algorithms 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, e.g., Ghadimi and Lan [2015].

The pre-asymptotic 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, e.g., Chau and Fu [2015] and Dong et al. [2017] for recent examples of simulation experiments assessing the relative performance of SO solvers. 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 by 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, where 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 while 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 or class of solvers.

Testbeds have been of great value to research communities outside of SO [Gould et al., 2015, Ali et al., 2005, Netlib, 2021, Wikipedia, 2021]. Compared to 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] 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. In doing so, we attempt to capitalize on related metrics developed in other optimization communities, namely, performance profiles [Dolan and Moré, 2002, Gould and Scott, 2016, Ali et al., 2005], 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, nor 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. In general, the estimation issues present in developing metrics and associated estimators are nontrivial.

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.
- We develop a variety of metrics and clarify their uses in evaluating one or more solvers on one or more problems, namely, progress curves, scatter plots, solvability profiles and difference profiles. The proposed metrics do not align perfectly with metrics that are 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, e.g., convergence plots and data profiles, while scatter plots and difference profiles are new.
- We explore the use of common random numbers (CRN) to improve various estimators.
- We describe a bootstrapping approach for assessing the estimation error in our estimated metrics.

Our experimental setup is readily implemented and has been fully implemented in SimOpt [Eckman et al., 2020].

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 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 provides examples of the comparative plots we advocate. Section 7 develops recommendations for how CRN should be employed, appealing to the literature on two-level simulation. We conclude in Section 8. The appendix provides the proof of a result given in Section 7 and

describes in more detail 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 would like 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 where 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 function [Rosenbrock, 1960]. 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 are devised 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—where the same additive stochastic error is applied at all feasible solutions—yields sample-path functions which 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. Related issues arise if one uses a multiplicative scaling by a strictly positive stochastic process with mean 1. We believe synthetic problems are problematic when it comes to comparing SO solvers and do not consider them further.

In this paper we consider optimization problems of the form $\min f(x)$ subject to $x \in \mathcal{D}$, 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, i.e., simulation is needed only in estimating f. This setup excludes the case where constraints of the form $g(x) \geq 0$ must be satisfied, where $g(x)$ is estimated by simulation. Such problems do arise in practice, but we do not yet have a recommendation on

performance metrics for them. Hence we consider solvers for single-objective optimization without stochastic constraints.

Our framework fixes a budget for each problem in terms of simulation replications, within which solvers must operate. Accordingly, we exclude fixed-confidence solvers with random running times, such as many ranking-and-selection algorithms that provide frequentist statistical guarantees. Our framework encompasses a diversity of algorithms, including gradient-descent, trust-region and direct-search methods, and evolutionary algorithms, which track populations of solutions, but it is not completely general.

Simulation optimization gives rise to multiple sources of randomness. Function estimates obtained from simulation oracles 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*, where a macroreplication is a single run of a solver on a problem that proceeds until a budget on the number of simulation replications is exhausted. Macroreplications collectively give a sense of the variable performance of a solver on a problem and are central to how we evaluate solvers. (However, 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 towards solutions whose *estimated* objective function value is lower (for minimization problems) than the true value due to 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 post-processing stage. The distributional information we obtain from multiple macroreplications and postreplications then needs to be summarized in some fashion. Our experimental setup will be made more mathematically precise in Section 3.

3 Progress Curves

Consider evaluating the performance of a single solver on a single problem. In deterministic optimization, a common measure of performance 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*, e.g., 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 simulating 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.

We advocate fixing a problem-specific budget of simulation replications and tracking the solution that the solver would recommend as the best—as a function of the number of simulation replications used in the search—until the budget is expended. We measure computational effort in terms of simulation replications, partly because this metric is somewhat universal across computing architectures. Importantly, we do not measure effort in terms of iterations, since the number of replications per iteration can change as the search proceeds. The recommended solution would typically be the one that has the best estimated objective function value amongst those simulated thus far, but not always; e.g., with Polyak-Ruppert averaging one recommends the average of all solutions visited, which itself may not have been visited. Our approach has a number of strengths that we will discuss shortly, but like any approach it has shortcomings. First, the recommended solution at an intermediate budget may not be easily defined for some solvers. For example, many ranking-and-selection algorithms are designed to run until a stopping condition is met, at which point they terminate and propose a solution as optimal. Such algorithms do not typically offer an intermediate recommended solution. On a related note, for solvers that use parallel computation to simultaneously simulate solutions, the point at which a certain intermediate budget of simulation replications has been expended is not well defined, since it depends on the scheduling of processors. Second, some solvers use the budget when specifying parameters that control, for example, an exploration-versus-exploitation tradeoff. The quality of an intermediate recommended solution will therefore not necessarily reflect how well the same solver would do if given a smaller or larger budget.

For a given problem, let T be the computational budget—as measured in simulation replications—and let $X(t)$ be the recommended solution after a fraction $t \in [0, 1]$ of the budget has been expended. 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, e.g., genetic algorithms or random-search algorithms. Collectively, the solutions recommended throughout the budget are described by the stochastic process $\boldsymbol{X} := \{X(t) \colon t \in [0, 1]\}$. Although the recommended solution can change at only discrete times $t = 0, 1/T, 2/T, \ldots, 1$, we find it mathematically convenient to represent $\boldsymbol{X}$ as a continuous-time stochastic process.

We are especially interested in the stochastic process $f(\boldsymbol{X}) := \{f(X(t)) \colon t \in [0, 1]\}$ where the random variable $f(X(t))$ is the *true* objective function value of the *random* recom-

mended 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, i.e., 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 (standardize) 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. To that end, we rescale the vertical axis to represent the *relative optimality gap*, which we define as the proportion of the initial optimality gap that has yet to be eliminated. Related scalings have been used when reporting results for deterministic solvers on deterministic-optimization problems; see, e.g., Moré and Wild [2009].

To that end, let x_0 denote a fixed initial solution and let x^* denote a fixed optimal solution. As previously mentioned, usually we do not know the optimal solution to a given simulation-optimization problem; however, in our analysis, we treat x^* as fixed, acknowledging that this creates an imperfection when x^* is in fact estimated from the same macroreplications we are analyzing. Henceforth, we let x^* represent an optimizer, or its proxy, and ignore variability in that term; our analysis is therefore conditional on x^*.

Remark: In our presentation, 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, i.e., $f(X(t))$, 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. While varying x_0 from macroreplication to macroreplication would indicate how robust the solver is to starting from different initial solutions, it would also lead to different rescalings of $f(X(t))$ for different macroreplications. For simplicity, we fix x_0 across macroreplications, but point out that different initial solutions on the same problem could be treated as distinct problems [Wild, 2019].

Let $f(x_0)$ and $f(x^*)$ denote the objective function values at x_0 and x^*, respectively. In the typical situation where an optimal solution is unknown, we can replace $f(x^*)$ with an estimate from the results of previous experiments, perhaps using a different solver on the same problem, or with a bound—a lower bound in the case of minimization problems or an upper bound in the case of maximization problems. We make several assumptions about the objective function values at x_0, x^* and elsewhere:

A1 $f(x)$ is bounded above and below and attains its optimal value at x^*.

A2 $f(x_0) \neq f(x^*)$, i.e., the search does not start at an optimal solution.

To standardize 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*. 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 A1 and A2, $\nu(t)$ 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 $f(x_0) \leq f(X(t)) \leq f(x^*)$, so that both numerator and denominator are negative and again $\nu(t) \in [0,1]$. In either case, a solver that recommends better quality solutions over time will have $\nu(t)$ decrease towards 0 as t increases. However, $\nu(t)$ could take values above 1 if the corresponding recommended solution $X(t)$ is worse than the initial solution x_0. Likewise, $\nu(t)$ could take values below 0 if $X(t)$ is infeasible in a constrained optimization problem. Though not mathematically precise, we find it convenient to presume that $\nu(t)$ takes only values between 0 and 1 when making general statements about the progress curve and functionals thereof.

Remark: The plot of $\boldsymbol{\nu}$ depends heavily on the choice of the budget, T. If T is very small, a solver will not make much progress toward finding x^* and $\nu(t)$ will hover near 1. If T is very large, then a solver will have conceivably made substantial progress towards an optimal solution relatively early and then remained near such an optimizer for the remaining time, in which case $\nu(t)$ will quickly drop to 0 and remain 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 most solvers have a chance to stabilize, 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.

Remark: Since the optimal solution to a simulation-optimization problem is often unknown, the reference solution x^* may be estimated from running one or more solvers on the problem and may therefore indirectly depend on T. We ignore this dependence.

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 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, e.g., how consistently it recommends high-quality solutions at an intermediate budget. Alternatively, 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. We build upon these different perspectives to devise summary statistics for a solver's performance on one or more problems.

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 (e.g., convergence) 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

can be framed as instances of stochastic functionals: 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 $[0, 1]$. Examples of the former include the expected area under $\boldsymbol{\nu}$ and the expected time at which $\boldsymbol{\nu}$ first drops below some value $\alpha \in [0, 1]$, while examples of the latter include the mean, median and quantile of $\nu(t)$ as functions of t.

Studying the distribution of the stochastic process $\boldsymbol{\nu}$ at a fixed time or at 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 profiles 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 behavior and tail behavior of $\nu(t)$ for each $t \in [0, 1]$, which is a lot of information to summarize. 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, e.g., the mean, median or some other quantile of $\nu(t)$ as a function of t. The mean or median is frequently reported; see, e.g., Dong et al. [2017]. Selected quantiles provide some indication of the variability in a solver's performance across macroreplications. The mean also provides useful information, averaging as it does across macroreplications. The mean could be especially affected by macroreplications with very poor performance, so the mean is best viewed as summarizing some combination of "typical" and "especially poor or strong" behavior.

The *mean progress at time* $t \in [0, 1]$, i.e., after a fraction t of the budget has been expended, 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 $\chi_\beta(t) := \inf\{q\colon \Pr\{\nu(t) \leq q\} \geq \beta\}$. 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. (Where no confusion can arise, we suppress β in this notation.) The mean and β-quantile progress will generally take values between 0 and 1 for all $t \in [0, 1]$, though it is possible for them to takes values outside of this interval.

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 $\boldsymbol{X}_m := \{X_m(t)\colon t \in [0, 1]\}$ for $m = 1, 2, \ldots, M$. Throughout this paper, a subscript m indicates a quantity associated with macroreplication m; e.g., $\boldsymbol{\nu}_m$ 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:

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

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

quantile, i.e., the $\lceil M\beta \rceil$th smallest value in the list of the progress values $\nu_1(t), \nu_2(t), \ldots, \nu_M(t)$:

$$\chi(t; M) := \inf \left\{ q \colon \frac{1}{M} \sum_{m=1}^{M} \mathbb{I}(\nu_m(t) \leq q) \geq \beta \right\}.$$

Remark: The choice of M affects the precision of the estimators $\mu(t; M)$ and $\chi(t; M)$ for any $t \in [0, 1]$. Specifically, in aggregating the progress curves $\boldsymbol{\nu}_1, \boldsymbol{\nu}_2, \ldots, \boldsymbol{\nu}_M$, an outlier progress curve can significantly shift the aggregated estimates when M is small. Hence, 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. To obtain precise estimators, we may need a different M for each problem-solver pair; for simplicity, we instead choose a single common value for M.

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 an outer level we obtain i.i.d. macroreplications to estimate the distribution of ν, and in an inner level we obtain replications from the simulation model to estimate $f(X(t))$ for a given t and realization of $X(t)$. To be precise, a macroreplication of the solver produces a realization of $\boldsymbol{X}$ and we then obtain N postreplications at each solution recommended throughout 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 observations $Y_1(t), Y_2(t), \ldots, Y_N(t)$ are importantly not function evaluations taken by the solver during the macroreplication that produced $X(t)$, as these would likely yield an optimistic estimator of $f(X(t))$ due to optimization bias, as explored in, e.g., Mak et al. [1999]. Instead, the N postreplications, 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 $n = 1, 2, \ldots, N$, $\boldsymbol{Y}_n := \{Y_n(t) \colon t \in [0, 1]\}$ 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 while reducing redundant simulation of a solution that is possibly recommended 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 $\nu(t)$. We thus expect to use a greater runlength, L, for these common terms than the standard postreplication runlength, 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, i.e.,

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

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

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

We adopt the convention that the solver recommends the initial solution x_0 at time $t = 0$. Hence we set $\nu(0; L, N) = 1$, even though $f_N(x_0)$ almost certainly will 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

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

where $\nu_m(t; L, N)$ 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 of $\nu_1(t; L, N), \nu_2(t; L, N), \ldots, \nu_M(t; L, N)$, i.e.,

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

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.

At any given time $t \in [0,1]$ and macroreplication m, the estimated progress $\nu_m(t; L, N)$ may take a negative value if the solution $X_m(t)$ is misidentified as being better than x^* due to 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 7, where we use Taylor's theorem and the literature of two-level simulation to assess the order of the error when CRN are used not just at different times t, but (optionally) also across macroreplications. From a practical standpoint, we prescribe a general bootstrapping approach that provides error estimates for these metrics (and others); see Appendix B for a full description.

4 Area under Progress Curves

While aggregate progress curves summarize the typical and tail behavior of $\nu(t)$ for each $t \in [0, 1]$, they cannot be easily extended to help 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. The area under the progress curve is given by the random variable

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

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* over a fixed 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 could 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. While imperfect, distributional properties of A can be used for comparisons over multiple problems.

Let $\mu_A := \mathbb{E}A$ and $\sigma_A := \sqrt{\text{Var}A}$ 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 A1 and A2, μ_A is also the area under the mean progress curve:

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

The area under the mean progress curve, μ_A, combines the typical as well as especially poor or strong behavior of $\nu(t)$ but otherwise offers limited information about the distribution of $\nu(t)$. The standard deviation σ_A provides some information on variability. It does not measure the *within-macroreplication* variability of the objective function values across different recommended solutions. Instead, it reflects 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 sample standard deviation of the areas under the realized progress curves:

$$\mu_A(M) := \frac{1}{M}\sum_{m=1}^{M} A_m = \frac{1}{M}\sum_{m=1}^{M}\int_0^1 \nu_m(t)dt, \quad \text{and} \quad \sigma_A(M) := \sqrt{\frac{1}{M-1}\sum_{m=1}^{M}(A_m - \mu_A(M))^2}.$$

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:

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

and

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

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 construction 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 scatter plot of $\{(\mu_A^{p,s}(L, M, N), \sigma_A^{p,s}(L, M, N)) \colon p \in \mathcal{P}\}$. 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 scatter plots 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.

How sensitive are the metrics $\mu_A^{p,s}(L, M, N)$ and $\sigma_A^{p,s}(L, M, N)$, and their relative ordering across solvers s, to a problem's budget, T? When x^* is assumed to be fixed (and independent of T), increasing T changes the horizontal scaling of the progress curve—compressing it towards the left—while leaving the vertical scaling unchanged. As long as a solver proceeds to recommend solutions with better objective function values than the previous time-average on a given macroreplication m, increasing T will cause $A_m(L, N)$ to decrease. Consequently, 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 inter-quartile 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, i.e., 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 scatter plot due to solver characteristics. For example, a scatter plot depicting two solvers will appear 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. As we have discussed, performance, data and accuracy profiles have been used extensively to compare

deterministic-optimization solvers. In this section, we explore related ideas for simulation optimization, where the variable performance of a solver across macroreplications must also be addressed. To introduce key ideas, we first consider a single problem-solver pair.

5.1 A Single Problem-Solver Pair

We previously defined 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, i.e.,

$$\tau(\alpha) := \inf\{t \in [0,1] \colon \nu(t) \leq \alpha\} = \inf\{t \in [0,1] \colon f(X(t)) - f(x^*) \leq \alpha(f(x_0) - f(x^*))\},$$

where the second equality 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 $[0,1] \cup \{\infty\}$. We refer to $\tau(\alpha)$ as the *progress curve α-solve time*. The corresponding stochastic process $\boldsymbol{\tau} = \{\tau(\alpha) \colon \alpha \in [0,1]\}$ can be thought of as the inverse of $\boldsymbol{\nu}$, though this is not exact since $\boldsymbol{\nu}$ need not be monotone. This metric is another way, besides the area under the progress curves, to reduce a solver's performance from a function of time to a scalar for easier solver comparisons.

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, while values such as $\alpha = 0.05$ represent a quite strict requirement. Where 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 since τ is extended-valued moments will typically be infinite. Instead, we look at β-quantiles of τ defined as $\pi = \pi_\beta := \inf\{q \colon \Pr\{\tau \leq q\} \geq \beta\}$. Assuming we can exactly compute f, we can estimate π by the sample quantile over M macroreplications,

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

where $\tau_m = \inf\{t \in [0,1] \colon \nu_m(t) \leq \alpha\}$ 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 also be extended-valued, particularly when α is small. If f cannot be computed exactly, then we use two-level simulation:

$$\pi(L,M,N) := \inf\left\{q \colon \frac{1}{M}\sum_{m=1}^{M} \mathbb{I}(\tau_m(L,N) \leq q) \geq \beta\right\},$$

where $\tau_m(L,N) = \inf\{t \in [0,1] \colon \nu_m(t;L,N) \leq \alpha\}$ 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 $p \in \mathcal{P}$, i.e.,

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

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 $p \in \mathcal{P}$:

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

If we cannot compute f exactly, then the corresponding two-level estimator is

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

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

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

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

$$\rho_\beta^s(t; M) := \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \mathbb{I}(\pi_\beta^{p,s}(M) \leq t) \text{ and } \rho_\beta^s(t; L, M, N) := \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \mathbb{I}(\pi_\beta^{p,s}(L, M, N) \leq t),$$

respectively. Quantile-solvability profiles are perhaps more intuitive than cdf-solvability profiles since 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 and smoother for larger problem sets.

Since $\rho^s(t; M)$ 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 Appendix B.

The (estimated) cdf- or quantile-solvability profiles for all solvers can be plotted on the same graph, with higher curves indicating better performance on the set of problems. While area-under-the-progress-curve scatter plots 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. For example, one can contrast the performance of solvers at a range of budgets. Furthermore, one can compare multiple solvers on a single problem but with different initial solutions, treating each initial solution as a distinct problem.

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 shall see, a difference plot shows how the performance of each solver compares to that of a fixed benchmark solver, s_0. The benchmark solver s_0 could 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 could 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 δ^s(t) is deterministic and ranges between -1 and 1. It is also known as the continuously ranked probability score [Matheson and Winkler, 1976]. We define the *cdf-solvability difference profile*, henceforth cdf-difference profile, of solver s as $\boldsymbol{\delta}^s = \{\delta^s(t) \colon t \in [0, 1]\}$. 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: $\boldsymbol{\delta}^s_\beta = \{\delta^s_\beta(t) \colon t \in [0, 1]\}$ 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 $\delta^s(t; L, M, N) := \rho^s(t; M) - \rho^{s_0}(t; M)$ and $\delta^s_\beta(t; L, M, N) := \rho^s_\beta(t; M) - \rho^{s_0}_\beta(t; M)$.

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 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 25 instances of `SSCONT-1`—an (s, S) inventory problem with continuous demand and order quantities. Under an (s, S) inventory policy, when the on-hand inventory position drops to s (or below), an order is placed to bring the inventory position up to S. To avoid a notational clash where we denote solvers by s, we denote the "little s" in such inventory problems by y. The demand in each period is assumed to be exponentially distributed with mean μ_D, independent across periods, and the order lead time is assumed to be Poisson distributed with mean μ_L, also independent across periods. (A lead time of zero corresponds to receiving the ordered quantity at the start of the following period). The objective is to select the thresholds y and S to minimize the expected per-period total cost—the sum of back-order, order and holding costs. The back-order cost per unit is \$4, the holding cost per unit per period is \$1, the fixed order cost is \$36 and the variable order cost per unit is \$2. The system starts with an initial inventory of y at time 0 and is simulated for a total of 120 periods with the first 20 periods treated as warm-up. The 25 problem instances comprise all combinations of $\mu_D \in \{25, 50, 100, 200, 400\}$ and $\mu_L \in \{1, 3, 6, 9, 12\}$. These values are chosen to span a range of signal-to-noise ratios; higher values of μ_D and μ_L result in more variable inventory fluctuations, hence more variable costs. We re-parameterize the problem and place non-negativity constraints on the two continuous decision variables: y and $Q := S - y$.

We test two classes of solvers:

- `Random Search` randomly samples solutions from the feasible region—y and Q are each drawn independently from an exponential distribution with a mean of 200 units—and takes a fixed number of replications at each solution. New solutions are generated until the budget is exhausted. We test three versions of `Random Search` with 10, 50 and 100 replications taken per solution.
- `ASTRO-DF` is a stochastic derivative-free trust-region method that uses adaptive sampling at each visited solution [Shashaani et al., 2018].

All solvers are given a budget of $T = 1000$ replications for each problem instance with equal warm-up and run-length periods. It would be more appropriate to use different warm-up and run-length periods, as well as different numbers of postreplications, for those problem instances that may be harder and noisier than others, but this setup suffices for our demonstration purposes. We run $M = 10$ macroreplications of each solver on each problem instance, with all solvers starting from a common initial solution $x_0 = (y_0, Q_0) := (600, 600)$ on all macroreplications. In a post-processing stage, we take $N = 100$ postreplications at all recommended solutions. For each problem instance, the proxy optimal solution, x^*, is taken to

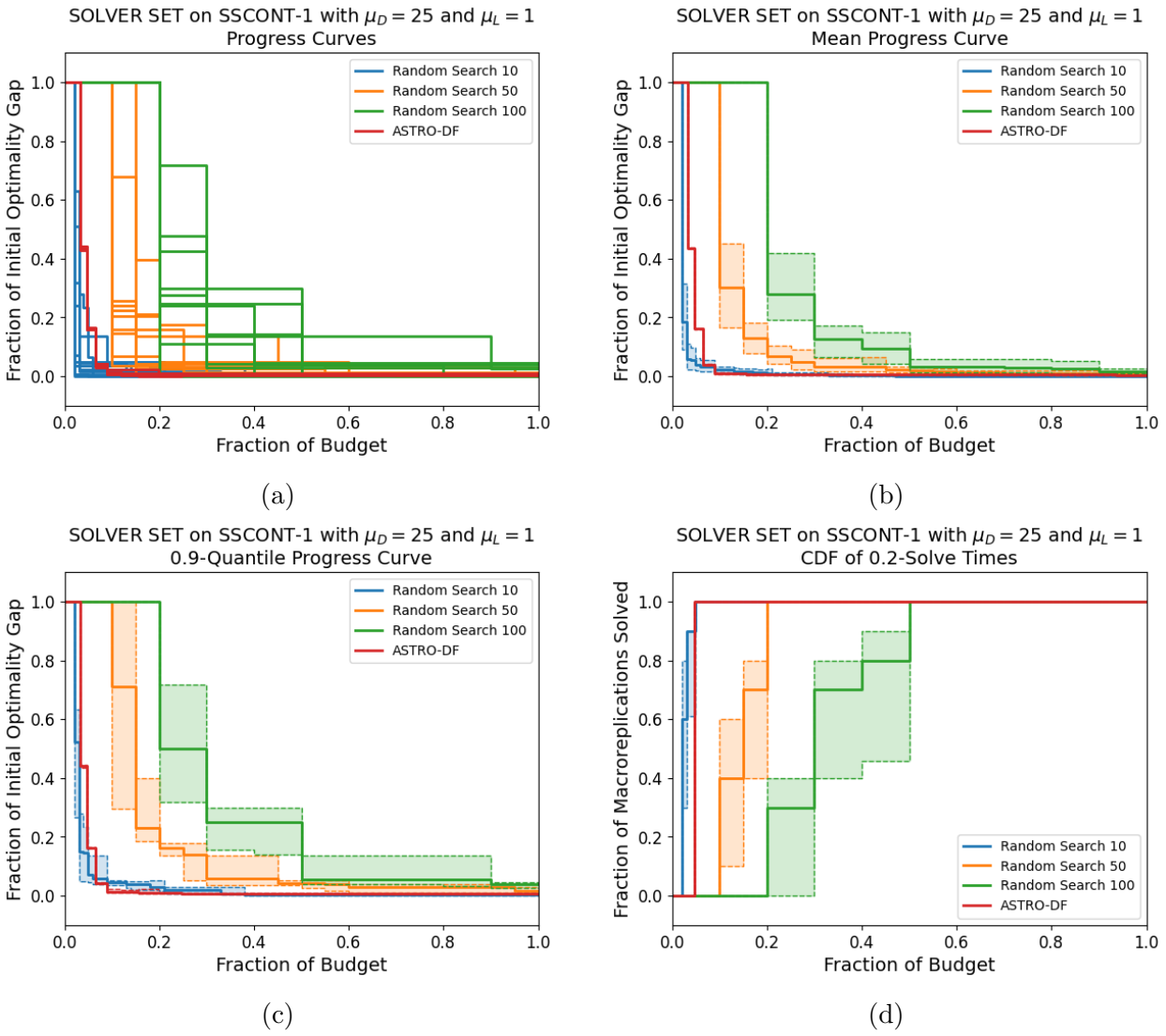

Figure 1: Results for `SSCONT-1`with $\mu_D = 25$ and $\mu_L = 1$: estimated progress curves (1a), mean progress curves (1b), 0.9-quantile progress curves (1c) and cdfs of 0.2-solve times (1d).

be the recommended solution with the best postreplicated estimate $f_N(\cdot)$ over all macroreplications of all solvers. To normalize the progress curves, we take $L = 200$ postreplications at x_0 and x^* for each problem instance.

We present plots from two problem instances: one with $\mu_D = 25$ units and $\mu_L = 1$ period (Figure 1) and the other with $\mu_D = 400$ units and $\mu_L = 6$ periods (Figure 2). The confidence intervals in these plots, constructed via bootstrapping, indicate the error in estimating solver performance.

Figures (1a)–(1c) and (2a)–(2c) show how progress curves are useful for evaluating and comparing solver performance on individual problems. Recall that progress curves depict improvement relative to the initial optimality gap; hence all y-axes are, for the most part, between zero and one. For the problem instance with $\mu_D = 25$ and $\mu_L = 1$, the objective function estimates obtained during macroreplications are relatively precise and all solvers

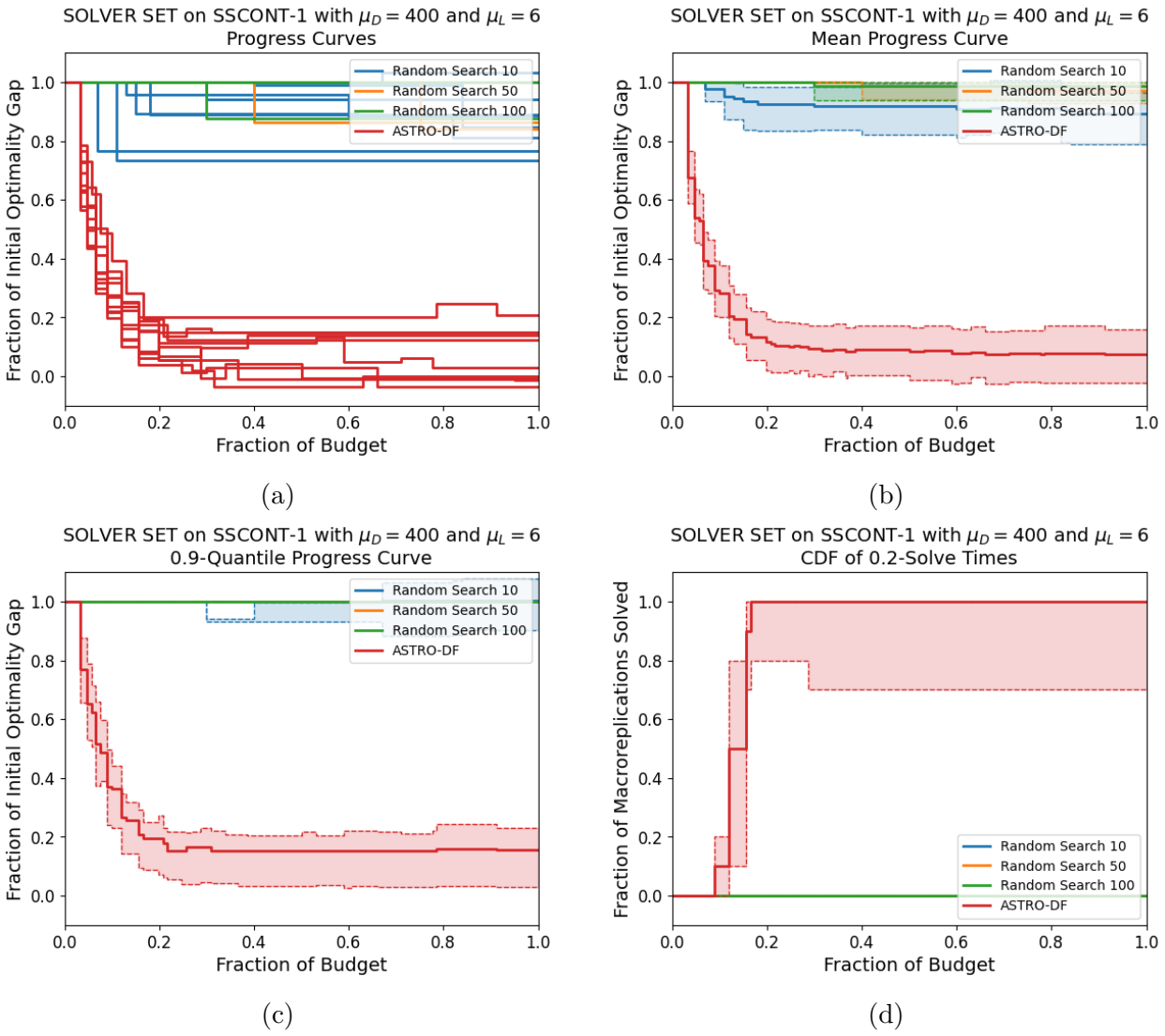

Figure 2: Results for `SSCONT-1`with $\mu_D = 400$ and $\mu_L = 25$: estimated progress curves (2a), mean progress curves (2b), 0.9-quantile progress curves (2c) and cdfs of 0.2-solve times (2d).

show reliable progress over time (Figure (1a)), with `ASTRO-DF` showing more reliable performance than the others. For the problem instance with $\mu_D = 400$ and $\mu_L = 6$, the objective function estimates are highly variable, particularly due to the high variance of the lead time. Accordingly, the solvers' performances are either more variable from macroreplication to macroreplication or are consistently poor (Figure (2a)). On this more difficult problem instance, the `Random Search` solvers appear to struggle more than `ASTRO-DF`, perhaps indicating that the noise in the objective function estimates overwhelms the true differences in the objectives, leading to highly random recommended solutions.

Figures (1b)–(1c) and (2b)–(2c) show the aggregate (mean and 0.9-quantile) progress curves for all four solvers. These curves illuminate the average and variable performance of different solvers over time and offer a clearer ordering of the solvers based on their empirical performance.

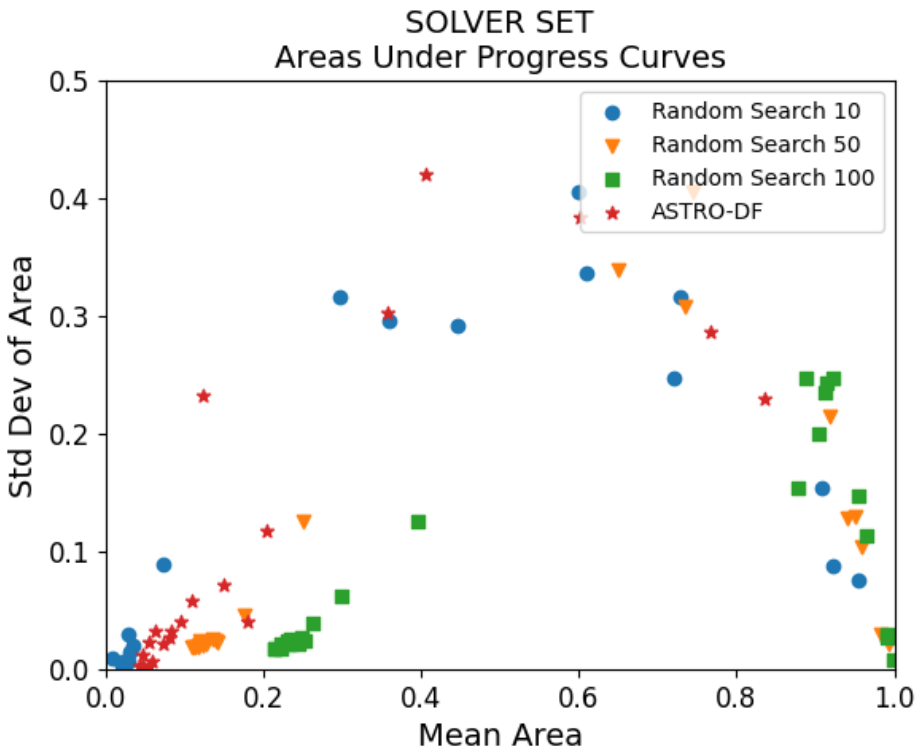

Figure 3: Scatter plot of the mean and standard deviation of areas under estimated progress curves for all problem instances.

Figures (1d) and (2d) offer a different perspective of the solvers' performances on these two problem instances. They depict the cdf of the α-solve times for $\alpha = 0.20$, i.e., 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 (1d) shows the same ranking of solvers on the problem instance with $\mu_D = 25$ units and $\mu_L = 1$ period, while Figure (2d) makes it apparent that the `Random Search` solvers never 0.2-solve the problem instance with $\mu_D = 400$ units and $\mu_L = 6$ periods.

Next, we examine the tools we have proposed to compare solvers over a testbed of problems. Figure 3 is a scatter plot where the coordinates of each point provide the mean and standard deviation of the areas under the estimated progress curves (over macroreplications) for a given problem-solver pair. When the number of problem-solver pairs is small, it may be beneficial to also show the bootstrapped confidence intervals of the estimated means and standard deviations with horizontal and vertical bars, respectively; in this plot, we suppress that information to avoid clutter. In the lower-left quadrant of the plot, which corresponds to solvers making rapid, reliable progress, there appears to be a clear ordering among the solvers. Performances are more scattered in the high-mean area, which corresponds to poor solver performance, especially among the `Random Search` solvers. From inspecting the high-mean area, it appears that `ASTRO-DF` struggles on only five problem instances. `Random Search 10` performs mostly well, but there are a number of problems where it fails to make much progress, as evidenced by points in the lower-right quadrant. Indeed, when solvers are unable

to solve a problem, the area under a progress curve is about 1 in most macroreplications, leading to a high mean and low standard deviation of the area under the progress curves. From this plot, `Random Search 50` and `Random Search 100` appear to be dominated by the other two solvers.

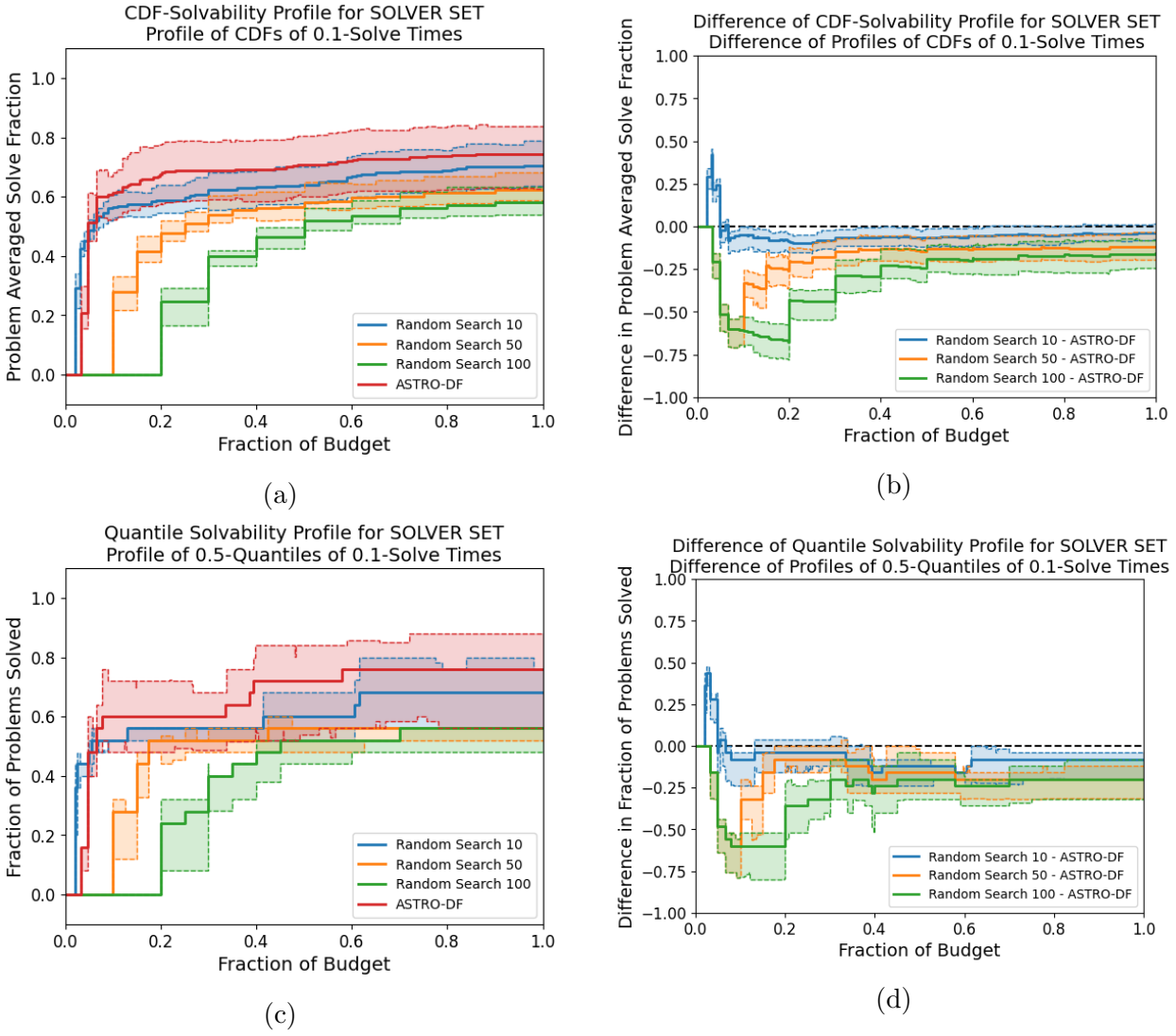

Figure 4: Profiles over all problem instances: cdf solvability profiles (4a), difference of cdf solvability profiles (4b), quantile solvability profiles (4c) and difference of quantile solvability profiles (4d).

Figures (4a) and (4c) show cdf- and quantile-solvability profiles of the four solvers over the set of problem instances. The cdf-solvability profile pertains to the time required to reduce the optimality gap (of any problem) to a tenth 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 problems that all solvers fail to 0.1-solve

within the budget of 1000 replications. `Random Search 10` is the fastest of the three `Random Search` solvers to 0.1-solve the problems, which is to be expected since the cheaper sampling of solutions allows `Random Search 10` to visit better solutions sooner than `Random Search 50` and `Random Search 100`. `Random Search 10`'s superior initial performance relative to `ASTRO-DF` quickly disappears as the fraction of the budget increases and `ASTRO-DF` then dominates. Unlike `Random Search`, the times at which `ASTRO-DF` identifies improved solutions do not belong to a fixed set; this explains the relative smoothness of its cdf-solvability profile.

The difference profiles in Figures (4b) and (4d) depict the performance gaps between `ASTRO-DF` and each variant of `Random Search` and further clarify the ordering of their performance over the problem set. An advantage of the difference profiles is that they exploit common random numbers, so we see statistically significant differences between the solvers.

7 Should we use Common Random Numbers in Postreplications?

The plots in Section 6 were obtained through prototype software that makes judicious and automated use of CRN. The decision of where to use CRN is delicate; CRN can lead to smoother plots due to the introduction of dependence in plot heights at different points, but as we shall see herein, if not applied with care it can also be counterproductive. We focus on whether CRN should be used *across macroreplications* in performing postreplications. Using CRN in this fashion may be an appealing way to reduce any perceived difference in performance from one macroreplication to another due to independent statistical noise. For example, one might take this approach to attempt to better identify which of the recommended solutions from the macroreplications is best. Ultimately, we will not recommend using CRN in this manner because it slows down convergence rates of the estimators.

It is important to draw a distinction with our recommended use of CRN for evaluating solutions encountered on a *single* macroreplication across different times. Doing so will typically produce smoother estimated progress curves. The discussion here concerns coupling the evaluations of solutions recommended on *different* macroreplications.

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. Nevertheless, the analysis for a fixed t suffices to make our point that CRN across macroreplications is undesirable.

We consider two possible dependence structures in postreplications. In both cases we allow the L postreplications at x_0 and x^* to be statistically dependent. The difference is instead whether the postreplications at different recommended solutions across macroreplications are coupled through CRN.

Independence: Here we assume that the postreplications used to estimate $f(X_i(t))$ are independent of those used to estimate $f(X_j(t))$ for $i \neq j$, i.e., that the postreplications used to 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 $f(X_j(t))$, $j = 1, 2, \ldots, M$.

CRN: Here we assume that CRN are used in the postreplications to evaluate the solutions from different macroreplications, so that $f_N(X_i(t))$ and $f_N(X_j(t))$ are dependent for $i \neq j$. For simplicity, we further assume that the L postreplications used to estimate $f(x_0)$ and $f(x^*)$ are independent of the N postreplications used to evaluate the recommended solutions from different macroreplications.

Throughout this section, we take x^*, the optimal solution, to be deterministic and given. In practice it will often need 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 will depend 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. This should make it clear that using CRN across macroreplications is not advisable.

7.1 Mean Progress Curves

How accurate is the estimator $\mu(t; L, M, N)$ of $\mu(t)$? This question is important, not just in providing some sense of accuracy when the plots are produced, but also in aiding in the choice of L, M and N.

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, M and N as functions of c, which we assume are bounded below by 1 to avoid trivialities. We first run $M(c)$ macroreplications with some per-macroreplication cost (average number of simulation replications) Tt and then complete the postreplications at cost $2L(c)+M(c)N(c)$ replications, yielding the estimator $\mu(t; L, M, N)$. Thus, $c = TtM(c) + 2L(c) + M(c)N(c)$. (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:

A3 Simulation replications at any solution x are unbiased, i.e., $\mathbb{E}Y_1(x) = f(x)$ for all x.

A4 Simulation replications at any solution x have bounded (in x) non-zero variance, i.e., $\sigma^2(x) := \operatorname{var} Y_1(x)$ is positive and bounded in x.

The uniformity in Assumption A4 (and in A1 stated earlier) permits a transparent analysis, which is restrictive but could be relaxed with effort. We do not pursue that effort because the conclusion is clear as is. Assumption A3 could be relaxed, as might be needed in the context of steady-state simulation, for example. We expect the conclusions of Theorem 1 to hold under relaxed assumptions, but establishing the result under such conditions would require further effort with, we believe, no change to the overall conclusion.

For notational simplicity we suppress the dependence of L, M and N on c. We say that a family of random variables $X(c)$ is $O_p(h(c))$ if the family $\{X(c)/h(c) \colon c \geq c_0\}$ is tight for some $c_0 > 0$. The proof of the following result appears in Appendix A.

Theorem 1 *Suppose that Assumptions A1–A4 hold.*

1. *Suppose in addition that* $\min(L, M) \to \infty$ *as* $c \to \infty$. *In the* Independence *case* $\mu(t; L, M, N) \to \mu(t)$ *in probability as* $c \to \infty$. *In the* CRN *case the same conclusion holds provided that, in addition,* $N \to \infty$ *as* $c \to \infty$.

2. *Consider the* Independence *case. If* $\min(L, M) \to \infty$ *as* $c \to \infty$ *then*

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

where the random variable $\widetilde{\mu}(t; L, M, N)$ *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.

3. *Consider the* CRN *case. If* $\min(L, M, N) \to \infty$ *as* $c \to \infty$, *then*

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

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

$$\kappa^2(L, M, N) = \frac{a_4}{L} + \frac{a_5}{M} + \frac{a_6}{N},$$

for appropriate constants a_4, a_5 *and* a_6.

Part 1 of Theorem 1 is implied by Parts 2 and 3. Nevertheless, we separated that result because it showcases the fact that for consistency we require the number of postreplications to grow without bound in the CRN case, but not in the Independence case. The difference between the two cases is further highlighted in Parts 2 and 3 of the theorem that provide rates of convergence and allow a more precise comparison. Indeed, for given values of L, M and N, $c = TtM + MN + 2L$, representing the total number of simulation replications, 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 large budgets c, in the Independence case, the variance of $\widetilde{\mu}(t; L, M, N)$ 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. On the other hand, in the CRN case, the number of postreplications N must increase without bound. Given that the total number of simulation replications is at least MN, we conclude that the convergence rate of the estimator $\mu(t; L, M, N)$ is necessarily slower in the CRN case than in the Independence case.

The conclusion that in the Independence case 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, since such choices will depend on parameters that are difficult to compute. Rather, we present

the result to draw forth the key conclusion that CRN will deteriorate the convergence rate of the estimator relative to the Independence case.

One might be tempted to use Theorem 1 to develop confidence intervals on values of the mean progress curve. Doing so would require 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 Appendix B.

7.2 Quantile Progress Curves

The quantile progress estimator $\chi(t; L, M, N)$ can be analyzed using techniques similar to those we used 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], where asymptotic theory is developed in the case where the number of postreplications N is the same at all macroreplication solutions. It is natural, however, to not 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.

Assume that we do not use CRN in the postreplications. 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].

Should we use CRN in the postreplications? In general, the answer is no, since that will slow down the convergence rate of the estimator $\chi(t; L, M, N)$. To see why this is plausible, note that the path to establishing the asymptotic mean-squared error of the quantile estimator requires, first, the analysis of the empirical cdf of $f_N(X_1), f_N(X_2), \ldots, f_N(X_M)$, which can be written as

$$\widehat{F}(\cdot; M, N) = \frac{1}{M} \sum_{m=1}^{M} \mathbb{I}(f_N(X_m) \leq \cdot).$$

Here the macroreplication solutions $X_1, X_2, \ldots, X_M$ are independent, but we assume that $f_N(x)$ and $f_N(x')$ are dependent for any solutions x and x', consistent with the use of CRN. In this case, the empirical cdf $\widehat{F}(\cdot; M, N)$ is an average of dependent terms, and we expect that these terms are positively correlated. Thus, the empirical cdf will have larger variance than if the postreplications were independent, leading to slower convergence.

8 Conclusions

We have developed and demonstrated 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. They differ in the way they are scaled, with the x-axis reflecting the fraction of the computational budget expended, and with 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 scatter plots and solvability profiles can prove useful in this more information-rich setting, by providing a high-level view of overall performance.

We provided some examples of these plots that clarify both their nature and their 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 have explored here, but also in other settings such as in data farming [Eckman et al., 2021]. Those design improvements will be reported elsewhere.

Acknowledgments

This work was supported in part by National Science Foundation grants TRIPODS+X DMS-1839346 and CMMI-2035086.

References

M. Montaz Ali, Charoenchai Khompatraporn, and Zelda B. Zabinsky. A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. *Journal of Global Optimization*, 31(4):635–672, 2005.

S. Andradóttir. An overview of simulation optimization via random search. In S. G. Henderson and B. L. Nelson, editors, *Simulation*, volume 13 of *Handbooks in Operations Research and Management Science*, chapter 20, pages 617–631. Elsevier, North Holland, 2006.

S. Andradóttir. A review of random search methods. In Michael C. Fu, editor, *Handbook of Simulation Optimization*, volume 216 of *International Series in Operations Research & Management Science*, chapter 10, pages 277–292. Springer, New York, 2015.

Vahid Beiranvand, Warren Hare, and Yves Lucet. Best practices for comparing optimization algorithms. *Optimization and Engineering*, 18(4):815–848, 2017.

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

Mark Broadie, Yiping Du, and Ciamac C. Moallemi. Risk estimation via regression. *Operations Research*, 63(5):1077–1097, 2015.

Marie Chau and Michael C. Fu. An overview of stochastic approximation. In Michael C. Fu, editor, *Handbook of Simulation Optimization*, volume 216 of *International Series in Operations Research & Management Science*, chapter 6, pages 149–178. Springer, New York, 2015.

Kyle Cooper, Susan R. Hunter, and Kalyani Nagaraj. Biobjective simulation optimization on integer lattices using the epsilon-constraint method in retrospective approximation framework. *INFORMS Journal on Computing*, 32(4):1080–1100, 2020.

Mame Astou Diouf and Jean Marie Dufour. Improved nonparametric inference for the mean of a bounded random variable with application to poverty measures. Technical report, Département de Sciences Économiques, Université de Montréal, 2005.

Elizabeth D Dolan and Jorge J Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002.

Naijia Dong, David J. Eckman, Xueqi Zhao, Matthias Poloczek, and Shane G. Henderson. Empirically comparing the finite-time performance of simulation-optimization algorithms. In W. K. V. Chan, A. D'Ambrogio, G. Zacharewicz, N. Mustafee, G. Wainer, and E. Page, editors, *Proceedings of the 2017 Winter Simulation Conference*, pages 2206–2217, Piscataway, New Jersey, 2017. Institute of Electrical and Electronics Engineers, Inc.

D. J. Eckman, S. G. Henderson, R. Pasupathy, and S. Shashaani. Simulation optimization library. `http://www.simopt.org`, 2020. [Online; Accessed May 1, 2020].

David J. Eckman, Shane G. Henderson, and Raghu Pasupathy. Redesigning a testbed of simulation-optimization problems and solvers for experimental comparisons. In N. Mustafee, K.-H. G. Bae, S. Lazarova-Molnar, M. Rabe, C. Szabo, P. Haas, and Y.-J. Son, editors, *Proceedings of the 2019 Winter Simulation Conference*, pages 3457–3467, Piscataway, New Jersey, 2019. Institute of Electrical and Electronics Engineers, Inc.

David J. Eckman, Sara Shashaani, and Susan M. Sanchez. Data farming for simulation optimization. Working paper, 2021.

Bradley Efron. Nonparametric standard errors and confidence intervals. *The Canadian Journal of Statistics*, 9(2):139–158, 1981.

M. C. Fu. Optimization for simulation: theory vs. practice. *INFORMS Journal on Computing*, 14(3):192–215, 2002.

Saeed Ghadimi and Guanghui Lan. Stochastic approximation methods and their finite-time convergence properties. In Michael C. Fu, editor, *Handbook of Simulation Optimization*, volume 216 of *International Series in Operations Research & Management Science*, chapter 7, pages 179–206. Springer, New York, 2015.

P. W. Glynn. Additional perspectives on simulation for optimization. *INFORMS Journal on Computing*, 14(3):220–222, 2002.

Michael B. Gordy and Sandeep Juneja. Nested simulation in portfolio risk measurement. *Management Science*, 56(10):1833–1848, 2010.

Nicholas Gould and Jennifer Scott. A note on performance profiles for benchmarking software. *ACM Transactions on Mathematical Software (TOMS)*, 43(2):1–5, 2016.

Nicholas IM Gould, Dominique Orban, and Philippe L Toint. CUTEst: a constrained and unconstrained testing environment with safe threads for mathematical optimization. *Computational Optimization and Applications*, 60(3):545–557, 2015.

L. Jeff Hong, Sandeep Juneja, and Guangwu Liu. Kernel smoothing for nested estimation with application to portfolio risk measurement. *Operations Research*, 65(3):657–673, 2017.

H. J. Kushner and G. G. Yin. *Stochastic Approximation and Recursive Algorithms and Applications*. Springer-Verlag, New York, 2nd edition, 2003.

Erik Learned-Miller and Philip S Thomas. A new confidence interval for the mean of a bounded random variable. *ArXiv e-prints*, 2019. Preprint arXiv:1905.06208v2, `http://arxiv.org/abs/1905.06208v2`.

S. H. Lee. *Monte Carlo Computation of Conditional Expectation Quantiles*. PhD thesis, Stanford University, Stanford, CA, 1998.

Shing-Hoi Lee and Peter W. Glynn. Computing the distribution function of a conditional expectation via Monte Carlo: Discrete conditioning spaces. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 13(3):238–258, July 2003.

Jialin Li and Ilya O. Ryzhov. Convergence rates of epsilon-greedy global optimization under radial basis function interpolation. Working paper, 2020.

Wai Kei Mak, David P. Morton, and R. Kevin Wood. Monte Carlo bounding techniques for determining solution quality in stochastic programs. *Operations Research Letters*, 24(1): 47–56, 1999.

James E Matheson and Robert L Winkler. Scoring rules for continuous probability distributions. *Management Science*, 22(10):1087–1096, 1976.

José Luis Morales. A numerical study of limited memory BFGS methods. *Applied Mathematics Letters*, 15(4):481–487, 2002.

Jorge J Moré and Stefan M Wild. Benchmarking derivative-free optimization algorithms. *SIAM Journal on Optimization*, 20(1):172–191, 2009.

Netlib. Netlib. `http://netlib.org`, 2021. [Online; Accessed March 3, 2021].

Raghu Pasupathy and Shane G. Henderson. A testbed of simulation-optimization problems. In L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, editors, *Proceedings of the 2006 Winter Simulation Conference*, pages 255–263, Piscataway, New Jersey, 2006. Institute of Electrical and Electronics Engineers, Inc.

Raghu Pasupathy and Shane G. Henderson. SimOpt: A library of simulation optimization problems. In S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and M. Fu, editors, *Proceedings of the 2011 Winter Simulation Conference*, pages 4075–4085, Piscataway, New Jersey, 2011. Institute of Electrical and Electronics Engineers, Inc.

H. H. Rosenbrock. An automatic method for finding the greatest or least value of a function. *The Computer Journal*, 3(3):175–184, 1960.

Sara Shashaani, Fatemeh S Hashemi, and Raghu Pasupathy. ASTRO-DF: A class of adaptive sampling trust-region algorithms for derivative-free stochastic optimization. *SIAM Journal on Optimization*, 28(4):3145–3176, 2018.

Hao-Jun Michael Shi, Melody Qiming Xuan, Figen Oztoprak, and Jorge Nocedal. On the numerical performance of derivative-free optimization methods based on finite-difference approximations. *ArXiv e-prints*, 2021. Preprint arXiv:2102.09762, `http://arxiv.org/abs/arXiv:2102.09762v1`.

Yunpeng Sun, Daniel W. Apley, and Jeremy Staum. Efficient nested simulation for estimating the variance of a conditional expectation. *Operations Research*, 59(4):998–1007, 2011.

Wikipedia. Test functions for optimization. `https://en.wikipedia.org/wiki/Test_functions_for_optimization`, 2021. [Online; Accessed March 3, 2021].

Stefan Wild. Personal communication with authors, October, 2019.

Appendices

A Proof of Theorem 1

Recall the definition of $\mu(t; L, M, N)$ as

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

Let $R(L, M, N)$ be the numerator and $S(L)$ be the denominator of (1) and let $r = \mathbb{E}R(L, M, N)$ and $s = \mathbb{E}S(L)$ be their expected values. Then $\mu(t) = r/s = g(r, s)$, where the function $g(x, y) := x/y$. To establish Part 1, i.e., consistency, it suffices to show that the numerator and denominator of (1) each converge in probability. Each term in the denominator is simply a sample average, so the law of large numbers implies the needed convergence. As for the numerator, Chebyshev's inequality establishes that for any $\epsilon > 0$

$$\Pr\{|R(L, M, N) - r| > \epsilon\} \leq \operatorname{var} R(L, M, N)/\epsilon^2.$$

In the *Independence* case, using the identity $\operatorname{var}(A+B) \leq 2\operatorname{var} A + 2\operatorname{var} B$ and the conditional variance formula, conditioning on $X_{1:M}(t) := \{X_m(t) \colon m = 1, 2, \ldots, M\}$ gives

$$\begin{aligned}
\operatorname{var}\left(\frac{1}{M}\sum_{m=1}^{M} f_N(X_m(t))\right) &= \operatorname{var}\left(\frac{1}{M}\sum_{m=1}^{M}[f_N(X_m(t)) - \mathbb{E}f(X_1(t))]\right) &(2)\\
&= \mathbb{E}\operatorname{var}\left[\frac{1}{M}\sum_{m=1}^{M}[f_N(X_m(t)) - \mathbb{E}f(X_1(t))]\middle| X_{1:M}(t)\right]\\
&\quad + \operatorname{var}\mathbb{E}\left[\frac{1}{M}\sum_{m=1}^{M}[f_N(X_m(t)) - \mathbb{E}f(X_1(t))]\middle| X_{1:M}(t)\right]\\
&= \frac{1}{M^2}\mathbb{E}\sum_{m=1}^{M}\frac{\sigma^2(X_m(t))}{N} + \frac{1}{M^2}\operatorname{var}\sum_{m=1}^{M}[f(X_m(t)) - \mathbb{E}f(X_1(t))] &(3)\\
&= \frac{\mathbb{E}\sigma^2(X_1(t))}{MN} + \frac{\operatorname{var} f(X_1(t))}{M}. &(4)
\end{aligned}$$

In this calculation it was not strictly necessary to subtract the constant $\mathbb{E}f(X_1(t))$, but this step will prove useful when we re-use this argument for the CRN case. Moreover, $\operatorname{var} f_L(x^*) = \sigma^2(x^*)/L$. Since $N \geq 1$, the second term in (4) dominates in that expression, and we conclude that the variance of the numerator is $\mathcal{O}(M^{-1} + L^{-1})$ in the *Independence* case, yielding the desired result. In the *CRN* case, define $Y_{mn}(t)$ to be the estimated objective value from the nth postreplication on the mth macroreplication solution $X_m(t)$. As in the Independence case, we use the identity $\operatorname{var}(A+B) \leq 2\operatorname{var} A + 2\operatorname{var} B$ to bound the variance

of the numerator by twice the variance of (2) plus $2\sigma^2(x^*)/L$. Then (3) instead becomes

$$\begin{aligned}
&\frac{1}{M^2}\mathbb{E}\,\mathrm{var}\left[\frac{1}{N}\sum_{n=1}^{N}\sum_{m=1}^{M}Y_{mn}(t)\middle| X_{1:M}(t)\right]+\frac{1}{M^2}\,\mathrm{var}\sum_{m=1}^{M}[f(X_m(t))-\mathbb{E}f(X_1(t))]\\
&\quad=\frac{1}{M^2N}\,\mathrm{var}\left[\sum_{m=1}^{M}Y_{m1}(t)\middle| X_1(t)\right]+\frac{\mathrm{var}\,f(X_1(t))}{M} \qquad (5)\\
&\quad=O\left(\frac{1}{N}+\frac{1}{M}\right), \qquad (6)
\end{aligned}$$

where the last line follows since the variance of a sum of M identically distributed terms is $\mathcal{O}(M^2)$. Thus, under CRN, the variance of the numerator is $\mathcal{O}(L^{-1}+M^{-1}+N^{-1})$ and consistency follows.

Now consider Part 2 in the *Independence* case. For some point ξ lying on the line segment between (r,s) and $(R(L,M,N),S(L))$,

$$\begin{aligned}
&\mu(t;L,M,N)-\mu(t)=g(R(L,M,N),S(L))-g(r,s)\\
&\quad=\nabla g(r,s)'(R(L,M,N)-r,S(L)-s)+(\nabla g(\xi)-\nabla g(r,s))'(R(L,M,N)-r,S(L)-s), \qquad (7)
\end{aligned}$$

where $'$ denotes the transpose operator. The second term in (7) is a remainder term that, by a Taylor expansion of $\nabla g(\xi)-\nabla g(r,s)$ can be seen to be of order $O_p(L^{-1}+M^{-1})$. We define $\widetilde{\mu}(t;L,M,N)$ to be the first term in (7), which has mean 0. To obtain the order of variance of this random variable, we write

$$\begin{aligned}
&\widetilde{\mu}(t;L,M,N)=s^{-1}(1,-r/s)'(R(L,M,N)-r,S(L)-s)\\
&\quad=\frac{1}{s}\left[\frac{1}{M}\sum_{m=1}^{M}[f_N(X_m(t))-\mathbb{E}f(X_1(t))]-[f_L(x^*)-f(x^*)]-(r/s)[f_L(x_0)-f(x_0)-(f_L(x^*)-f(x^*))]\right]\\
&\quad=\frac{1}{s}\left[\frac{1}{M}\sum_{m=1}^{M}[f_N(X_m(t))-\mathbb{E}f(X_1(t))]-(1-r/s)[f_L(x^*)-f(x^*)]-(r/s)[f_L(x_0)-f(x_0)]\right]. \qquad (8)
\end{aligned}$$

The variance of (8) separates into two terms, the first of which we have already computed in (4) to be of order $\mathcal{O}(M^{-1})$. The second of these terms is

$$\frac{1}{s^2L}\left[(1-r/s)^2\sigma^2(x^*)+(r/s)^2\sigma^2(x_0)+2(1-r/s)(r/s)\rho(x_0,x^*)\sigma(x^*)\sigma(x_0)\right],$$

which is $\mathcal{O}(L^{-1})$. Here $\rho(x_0,x^*)$ denotes the correlation between the estimated objective function values from a single postreplication at each of x_0 and x^*. Thus, the variance of $\widetilde{\mu}(t;L,M,N)$ can be written as

$$\varsigma^2(L,M,N)=\frac{\mathbb{E}\sigma^2(X_1(t))}{s^2MN}+\frac{\mathrm{var}\,f(X_1(t))}{s^2M}+\frac{(1-\frac{r}{s})^2\sigma^2(x^*)+(\frac{r}{s})^2\sigma^2(x_0)+2(1-\frac{r}{s})\frac{r}{s}\rho(x_0,x^*)\sigma(x_0)\sigma(x^*)}{s^2L}.$$

Now consider Part 3, i.e., the *CRN* case. We still have (7) where the second (error) term is now $O_p(L^{-1} + M^{-1} + N^{-1})$ due to (5). Define $\widehat{\mu}(t; L, M, N)$ to be the first term in (7), which differs from $\widetilde{\mu}(t; L, M, N)$ due to the difference between the dependence structure in the *Independence* and *CRN* cases. Then $\widehat{\mu}(t; L, M, N)$ still has mean 0, but from (5) its variance is now of order $\mathcal{O}(L^{-1} + M^{-1} + N^{-1})$. $\square$

B Bootstrapping for Error Estimation

We propose a general bootstrapping approach for error estimation and point out specific applications to the various metrics presented in this paper. Suppose we have performed M independent macroreplications of a single solver on a single problem and we want to compute the error associated with estimating some functional of the progress curve. Relevant examples include the value of the mean or quantile progress at a fixed time t, the expected area under a progress curve or a quantile of the α-solve time, as arises in solvability profiles. Here we fix the problem and solver, so we suppress dependence on these quantities.

The following process outlines how one runs multiple macroreplications of a solver on a given problem, runs postreplications and bootstraps the results.

1. **Run Macroreplications:** Run M independent macroreplications of the solver to obtain sequences of recommended solutions $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_M$.

2. **Run Postreplications:** For each macroreplication $m = 1, 2, \ldots, M$, run N postreplications at each distinct solution in $\boldsymbol{X}_m = \{X_m(t) \colon t \in [0, 1]\}$ using CRN across the solutions within each macroreplication. Let $\boldsymbol{Y}_{mn} = \{Y_{mn}(t) \colon t \in [0, 1]\}$ denote the sample path of noisy observations from the nth postreplication at the solutions in $\boldsymbol{X}_m$. In addition, run L postreplications at x_0 and x^* and let Y_{0l} and Y_{*l} denote the corresponding noisy observations from the lth postreplication, $l = 1, 2, \ldots, L$.

3. **Bootstrap:** Generate B bootstrap instances of the relevant estimator by repeating Steps 3(a)–(c) below B times. The $\gamma/2$ and $1-\gamma/2$ sample quantiles of these bootstrap instances yield an approximate $100(1 - \gamma)\%$ confidence interval. Alternatively, the asymmetric bias-corrected bootstrap $100(1-\gamma)\%$ confidence interval can be computed [Efron, 1981].

   (a) **Outer-Level Bootstrap over Macroreplications:** Obtain a bootstrap sample of the sequences of recommended solutions, $\boldsymbol{X}_1^\star, \boldsymbol{X}_2^\star, \ldots, \boldsymbol{X}_M^\star$, i.e., sample M times with replacement from $\{\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_M\}$.

   (b) **Inner-Level Bootstrap over Postreplications:** For each bootstrapped sequence of recommended solutions, $\boldsymbol{X}_m^\star$, obtain a bootstrap sample of sample paths, $\boldsymbol{Y}_{m1}^\star, \boldsymbol{Y}_{m2}^\star, \ldots, \boldsymbol{Y}_{mN}^\star$, i.e., for each $m = 1, 2, \ldots, M$, sample N times with replacement from $\{\boldsymbol{Y}_{m1}, \boldsymbol{Y}_{m2}, \ldots, \boldsymbol{Y}_{mN}\}$. Obtain paired bootstrap samples of noisy observations from solutions x_0 and x^*, $Y_{01}^\star, Y_{02}^\star, \ldots, Y_{0L}^\star$ and $Y_{*1}^\star, Y_{*2}^\star, \ldots, Y_{*L}^\star$, i.e., sample L times with replacement from $\{\{Y_{01}, Y_{*1}\}, \{Y_{02}, Y_{*2}\}, \ldots, \{Y_{0L}, Y_{*L}\}\}$.

(c) **Compute the Bootstrap Instance of the Estimator:** Construct the M bootstrapped estimated progress curves

$$\boldsymbol{\nu}_m^\star(L,N) = \{\nu_m^\star(t;L,N)\colon t \in [0,1]\} = \left\{ \frac{f_N^\star(X_m^\star(t)) - f_L^\star(x^*)}{f_L^\star(x_0) - f_L^\star(x^*)} \colon t \in [0,1] \right\}$$

for $m = 1, 2, \ldots, M$, where

$$f_N^\star(X_m^\star(t)) = \frac{1}{N}\sum_{n=1}^{N} Y_{mn}^\star(t), \quad f_L^\star(x_0) = \frac{1}{L}\sum_{l=1}^{L} Y_{0l}^\star \quad \text{and} \quad f_L^\star(x^*) = \frac{1}{L}\sum_{l=1}^{L} Y_{*l}^\star.$$

Compute the bootstrap instance of the estimator as a functional of the estimated progress curves, $\boldsymbol{\nu}_\cdot^\star(L,N) = \{\boldsymbol{\nu}_1^\star(L,N), \boldsymbol{\nu}_2^\star(L,N), \ldots, \boldsymbol{\nu}_M^\star(L,N)\}$. For example, the estimator of the mean progress at time t for a given bootstrap instance is $M^{-1}\sum_{m=1}^{M} \nu_m^\star(t;L,N)$.

The bootstrapping algorithm above can be modified in a natural way for the case of multiple problems and/or solvers. Essentially, one simply replicates all steps over the additional dimensions of problems and solvers, though some care is necessary if CRN have been used. For example, for the solvability profile of a given solver, Steps 1–3 are carried out for each problem in $\mathcal{P}$. Each of these steps can be performed either independently across problems or using CRN, though the latter is unlikely to yield any variance reduction. For difference profiles, the same setup is performed for each solver, but here the use of CRN across solvers may be more beneficial.