Proceedings of the 2024 Winter Simulation Conference
H. Lam, E. Azar, D. Batur, S. Gao, W. Xie, S. R. Hunter, and M. D. Rossetti, eds.

REPEATEDLY SOLVING SIMILAR SIMULATION-OPTIMIZATION PROBLEMS: INSIGHTS FROM DATA FARMING

Nicole Felice[1], Sara Shashaani[1], David J. Eckman[2], and Susan M. Sanchez[3]

[1]Dept. of Industrial & Systems Eng., North Carolina State University, Raleigh, NC, USA
[2]Dept. of Industrial & Systems Eng., Texas A&M University, College Station, TX, USA
[3]Dept. of Operations Research, Naval Postgraduate School, Monterey, CA, USA

ABSTRACT

We study a setting in which a decision maker solves a series of similar simulation-optimization problems, where parameters of the problem vary over time to reflect the most up-to-date conditions. For example, in a generalized newsvendor problem, the newsvendor must choose how much of each raw material to order each day in response to the latest prices for procuring and salvaging raw materials. At the beginning of each day, a simulation-optimization solver is used to recommend a solution for that day's problem. This setting prompts several questions of potential interest: How sensitive are the solver's recommended solution and performance to changes in the problem's parameters? Does the use of a fixed set of solver hyperparameters lead to consistently good performance, or would judiciously tuning these hyperparameters lead to a practically significant improvement? We leverage data farming to provide some preliminary answers and describe additional insights uncovered during the process.

1 INTRODUCTION

Simulation-optimization (SO) is a framework for making operational decisions under uncertainty with the aid of stochastic simulation. SO solvers, i.e., algorithms, use mathematical programming and statistical analysis to optimize an objective function based on the output distribution of a simulation model. An SO solver usually possesses some hyperparameters whose settings can significantly impact its finite-time performance, namely, the solver's ability to consistently find good solutions. Recommended default values for the hyperparameters are often based on large-scale experimentation or informed by theory. However, attaining optimal performance of an SO solver on a given problem instance requires tuning its hyperparameters. This hyperparameter tuning adjusts the solver's operational logic based on properties of the problem and comes at added computational cost. There is a large body of research on hyperparameter tuning, the majority of which relies on rudimentary-level design of experiments, global neighborhood search methods, or local gradient-based optimization algorithms; see Huang et al. (2019) for a recent survey. We take a different approach to hyperparameter tuning by harnessing the power of data farming, as suggested by Shashaani et al. (2024). Data farming is an inference framework that involves building metamodels using a dataset "grown" from an experimental design (Sanchez 2020). This application of data farming for studying SO solvers enables users to learn more about the interaction effects between problem and solver factors, which can aid in making decisions about whether and how to tune a solver's hyperparameters.

In this paper, we demonstrate the potential analysis and insights offered by using data farming to study solver performance in a setting in which a decision maker will repeatedly apply a chosen SO solver on a problem whose parameters vary over time. More precisely, we are concerned with *factors* of the simulation model whose outputs determine the objective function of the optimization problem. Factors of simulation models are typically regarded as fixed and may represent known or unknown quantities. In practice though, many of these factors are subject to change over time. For example, in a newsvendor problem, costs of materials that will be purchased, along with their recourse or salvage costs, can change from day to day depending on market conditions. The values of these factors are known at the start of the day, before the

2024 Winter Simulation Conference (WSC) | 979-8-3315-3420-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/WSC63780.2024.10838951

solver is run, and can affect the properties of that day's problem, including the geometry of its objective function, and consequently its optimal solution. Thus, when using a solver on a series of similar problems, performing hyperparameter tuning on a repeated basis can become a computational hurdle. We instead ask whether a systematic approach of conducting a designed experiment offline and analyzing the resulting data can offer the decision maker guidance about how best to customize and run their SO solver.

Shashaani et al. (2024) describe three types of users of SO solvers: generalists who are interested in general solver performance, specialists who need to solve a particular class of problems on a regular basis, and focused users who use a solver only occasionally. A specialist, such as someone solving variants of the generalized newsvendor problem each day, has two overarching tasks when adopting the data farming approach we advocate here. The first can be considered offline, and involves conducting a data farming experiment to gain insight about the performance of solvers they are considering. This would typically involve the following steps:

1. Specifying experiment designs for the solver factors and problem factors of interest.
2. Running multiple macroreplications for each solver-problem pair.
3. Conducting additional confirmation runs at a solver's recommended solutions from each macroreplication to obtain better estimates of the solver's performance on the corresponding problem instance.
4. Normalizing the results from the individual problem instances to assess the relative performance of the solver variants.

The time required for such a large-scale experiment can be much longer than the time to run a single macroreplication of a given solver. However, this experiment is to be conducted once, offline, and the macroreplications of the different solver-problem pairs can be run in parallel.

The specialist's second task happens in real time. Once the insights have been gained from the larger experiment, the problem specialist will need to select a setting of the solver's hyperparameters with which to run the solver on the current problem. They will subsequently run that solver variant and implement the final recommended solution, without performing any confirmation runs.

This paper demonstrates how a specialist might design and analyze a large-scale experiment to improve the speed and efficacy of an SO solver when used to solve similar problems over time. We achieve this through an extensive study of a single SO problem and a single SO solver using the SimOpt library (Eckman et al. 2023; Eckman et al. 2020), an open-source SO testbed. Although the results presented in this paper are specific to the solver-problem pair, the experiment setup and the analysis and plotting methods are intended to serve as a blueprint for how to extract meaningful insights about solver performance when leveraging data farming in this setting.

The remainder of this paper is organized as follows: In Section 2, we briefly outline how we measure the finite-time performance of SO solvers in our experiments. In Sections 3 and 4, we describe a generalized newsvendor problem and adaptive-sampling trust-region solver, respectively, that form the basis for our experiments. We present the results of a data farming analysis in Section 5 and conclude with remaining challenges in Section 6.

2 MEASURING SOLVER PERFORMANCE

We consider an optimization problem of the form

$$\min_{x \in \mathcal{X}} \{f^w(x) := \mathbb{E}_\xi[F^w(x, \xi)]\}$$

where $w \in \mathcal{W}$ represents a setting (or instance) of problem factors in a defined problem-factor space $\mathcal{W}$ and $F^w(\cdot, \xi)$ is a random function with stochastic primitives represented by ξ. Although not reflected in the notation, we allow the distribution of ξ to depend on w. When running on a problem instance w, an SO solver with hyperparameters $s \in \mathcal{S}$ generates, in each run (macroreplication), a sequence of recommended solutions at different times until the solver terminates. For a macroreplication i, $i = 1, 2, \ldots, m$, we treat

this sequence of recommended solutions as a continuous-time stochastic process

$$\{X_i^{w,s}(t)\colon t \in [0,T]\},$$

where T is the termination time for the solver, as specified by the allowed number of simulation replications for the solver to expend, referred to as the simulation budget.

Let f_0^w and f_*^w denote the true objective function values of the initial solution and optimal solution, respectively, for problem instance w. The relative optimality gap at time t on macroreplication i is defined as

$$\nu_i^{w,s}(t) = \frac{f(X_i^{w,s}(t)) - f_*^w}{f_0^w - f_*^w}. \tag{1}$$

Normalizing by the initial optimality gap usually leads to $\nu_i^{w,s}(t)$ taking values between 0 and 1, which facilitates comparisons across different problem instances. We refer to the stochastic process $\{\nu_i^{w,s}(t)\colon t \in [0,T]\}$ as the *normalized progress curve* for macroreplication i, $i = 1, 2, \ldots, m$, for solver instance s on problem instance w (Eckman et al. 2023). Because the true objective function values are unknown in SO problems, one would naturally estimate them by their post-processed sample averages from, say, N independent simulation replications, i.e., confirmation runs. (The confirmation runs do not count toward the budget T and make use of common random numbers (CRN) across solutions, but are seeded independently from the replications used when running the solver.) Likewise, the unknown optimal solution can be replaced with the estimated best solution found on any macroreplication.

Normalized progress curves form the basis of several measures of solver performance. For example, the estimated *terminal relative optimality gap* for macroreplication i, $i = 1, 2, \ldots, m$, is defined as $\hat{\nu}_i^{w,s}(T)$, where the $\hat{}$ denotes that the true objective function values in (1) are replaced with their estimates. A solver instance s's estimated average performance upon termination on problem instance w is given by

$$\hat{\mathbb{E}}[\nu^{w,s}(T)] = \frac{1}{m}\sum_{i=1}^{m} \hat{\nu}_i^{w,s}(T).$$

Besides the terminal relative optimality gap, one can use other solver performance metrics such as the area under the normalized progress curves or the α-solve time—the first time at which the normalized progress curve drops below $\alpha \in (0,1)$. These solver performance metrics are automatically computed for experiments performed in SimOpt.

3 MULTI-PRODUCT NEWSVENDOR PROBLEM

We study a multi-product newsvendor problem adapted from Kim et al. (2015). In the problem, a vendor seeks to choose initial order quantities to optimize expected profit over a single sales period, in this case, a single day. At the start of the day, the vendor must decide what quantities to order of q different raw materials which can be used to produce p different products. Let the matrix $A \in \mathbb{R}^{p\times q}$ describe the bill of materials required to produce each product from the various raw materials, i.e., row $A_j = (a_{j1}, a_{j2}, \ldots, a_{jq})$ contains the amounts of each raw material required to produce one unit of product j, $j = 1, 2, \ldots, p$. At midday, the vendor observes that day's demand for each product j, denoted by the random variable ξ_j, which is assumed to be Poisson distributed with mean λ_j. Demands for different products are assumed to be independent. After observing the demand, the vendor has another opportunity to order additional raw materials, at higher costs, to help fulfill the observed demand. For each raw material i, $i = 1, 2, \ldots, q$, the costs to order one unit at the start of the day and at midday are $c_{m,i}$ (m for material) and $c_{r,i}$ (r for recourse), respectively, where $c_{m,i} < c_{r,i}$. The raw materials ordered at midday are delivered to the vendor later in the day, after which the on-hand inventory can be processed into finished products according to the requirements specified by A. The cost to produce one unit of product j is denoted by $c_{d,j}$. Product j is sold at a per-unit sales price of $p_{s,j}$ and any excess raw materials at the end of the day are sold at per-unit

salvage prices of $p_{v,i}$ for $i = 1, 2, \ldots, q$. Because products are produced only after observing the daily demand, there are no left-over products at the end of the day. The vendor has a budget b for purchasing raw materials and processing them into finished products.

The optimization problem can be formulated as a two-stage stochastic program

$$\max_{x \in \mathbb{R}_+^q} \left\{ \mathbb{E}_\xi[\Pi(x, \xi)] - c_m^\top x \right\} \text{ subject to } c_m^\top x \leq b$$

where the first-stage decision variables $x = (x_1, x_2, \ldots, x_q)$ represent the quantities of raw material i that are ordered at the start of the day for $i = 1, 2, \ldots, q$, and

$$\begin{aligned} \Pi(x, \xi) := \max_{y \in \mathbb{R}_+^q,\, z \in \mathbb{R}_+^p} & \left\{ (p_s - c_d)^\top z - c_r^\top y + p_v^\top (x + y - A^\top z) \right\} \\ \text{subject to } & -y + A^\top z \leq x \\ & z \leq \xi \\ & c_r^\top y + c_d^\top z \leq b - c_m^\top x \end{aligned}$$

represents the profit associated with the second-stage decision, after observing the demands. In the second-stage problem, $y = (y_1, y_2, \ldots, y_q)$ represents the quantities of the raw materials that are ordered midday and $z = (z_1, z_2, \ldots, z_p)$ represents the quantities of each product that are produced. Although the demand in the problem is assumed to be integer-valued, the order quantities (x and y) and production quantities (z) are modeled as real values. When tackling this problem using stochastic simulation, one replication corresponds to simulating one day's demand for fixed initial order quantities x.

Many parameters of the problem could be varied in an experiment design to yield different problem instances. We choose to vary the costs of raw materials at the start of the day (c_m) and at midday (c_r) and the salvage prices of the finished products (p_v), because their values could conceivably change from day to day, depending on the industry domain, and the vendor may have little to no control over them. The other problem parameters are fixed at values shown in Table 1.

Table 1: Fixed parameters of the multi-product newsvendor problem.

Problem Parameter	Description	Value
q	Number of raw materials	4
p	Number of products	3
A	Bill of materials matrix	$\begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 2 & 2 & 0 \\ 0 & 0 & 3 & 3 \end{bmatrix}$
c_d	Development costs for products	[0.1, 0.1, 0.1]
p_s	Sales prices for products	[6, 12, 20]
λ	Mean demands for products	[20, 25, 15]
b	Newsvendor $$ budget	1200

For illustration purposes, we opted to construct a small design by varying only a few of the problem parameters; we call these *problem factors* to distinguish them from fixed problem parameters. Accordingly, we treat the start-of-day order cost of raw material 1, $c_{m,1}$, as a factor and introduce three other factors that represent ratios between the order costs and salvage prices of the different raw materials.

- *Start-of-day cost ratio, M.* The ratio between each of the start-of-day order costs of raw materials with consecutive indices, i.e., $M = c_{m,i+1}/c_{m,i}$ for $i = 1, 2, \ldots, m-1$. Hence, raw materials with higher indices are more expensive to order at the start of the day.

- *Recourse cost ratio,* R. A common ratio between each raw material's midday order cost and start-of-day order cost, i.e., $R = c_{r,i}/c_{m,i}$ for $i = 1, 2, \ldots, q$. R reflects the premium paid for ordering raw materials later in the day.
- *Salvage price ratio,* S. A common ratio between each raw material's end-of-day salvage price and start-of-day order cost, i.e., $S = p_{v,i}/c_{m,i}$ for all $i = 1, 2, \ldots, q$.

These four factors are varied subject to lower and upper bounds given in Table 2.

Table 2: Factors of the multi-product newsvendor problem and ranges of their values.

Problem Factor	Description	Low Value	High Value
$c_{m,1}$	Cost of material 1	0.8	1.2
M	Material cost ratio	1	1.2
R	Recourse cost ratio	1.5	2.5
S	Salvage price ratio	0.4	0.6

We use a nearly-orthogonal Latin hypercube (NOLH) design over the four problem factors with three stacks, for a total of 17 + 16 + 16 = 49 design points, i.e., 49 different multi-product newsvendor problems.

For the parameter values shown in Table 1 and factor values and ranges shown in Table 2, some general statements can be made about the various products. Product 1 is composed of a small number of raw materials that are the cheapest to procure, and has the smallest profit margin. Product 2 is composed of a greater number of the same raw materials and is more profitable when sold. Product 3, on the other hand, requires more raw materials and more *expensive* raw materials, yet has the largest profit margin. The product with the highest mean demand (hence the highest variance) is Product 2; Product 3, meanwhile, has the lowest mean demand. These relationships influence the vendor's strategy for ordering raw materials in the most profitable way.

4 ASTRO-DF SOLVER

We use an adaptive-sampling trust-region solver called ASTRO-DF (Shashaani et al. 2018) with its recent coordinate-based direct search feature (Ha and Shashaani 2024); a brief description of the solver follows.

At each iteration, ASTRO-DF gathers simulation output data from a small experiment and constructs an approximate local model of the objective function, in a derivative-free way. Often this means fitting a local model to estimated objective function values of solutions carefully selected in a neighborhood around the incumbent solution. In our implementation, ASTRO-DF uses CRN across solutions when carrying out this local experiment. Hyperparameters $\gamma_1 > 1$ and $\gamma_2 \in (0, 1)$ govern the increase in the trust-region radius Δ_k, which can be viewed as the search step size in each iteration. If ASTRO-DF succeeds in finding a sufficiently better solution in an iteration, the trust-region size is expanded as $\Delta_{k+1} = \gamma_1 \Delta_k$ to allow faster progress toward optimality in the next iteration. On the contrary, when ASTRO-DF fails to find a better solution in an iteration, the trust-region size is shrunk as $\Delta_{k+1} = \gamma_2 \Delta_k$ to allow the algorithm to refine its approximation of the objective function in the next iteration.

Other hyperparameters η_1 and η_2 satisfying $0 < \eta_1 < \eta_2 < 1$ control the success criteria in ASTRO-DF. Determining whether the algorithm found a better solution in an iteration entails comparing the improvement in the objective function estimates with the improvement in the approximating local model; the ratio of these improvements is deemed sufficient (hence, success of the algorithm) if it is larger than η_1 and insufficient otherwise. If the ratio is larger than η_2, then the iteration is considered very successful and the trust region is expanded, whereas if the ratio is between η_1 and η_2, the trust-region size is unchanged. The algorithm's direct search feature bypasses this ratio test if one of the points used to fit the model reduces the objective function value by more than the model-recommended candidate solution, in which case a different sufficient-reduction test on that direct-search solution is performed.

ASTRO-DF has other hyperparameters, but for illustration purposes we limit our exploration to these four in our data farming experiments, as shown in Table 3.

Table 3: Factors of the ASTRO-DF solver, default values, and ranges of their values for the experiment.

Solver Factor	Description	Default	Low Value	High Value
γ_1	Trust-region shrinkage multiplier	1.5	1.0	3.0
γ_2	Trust-region expansion multiplier	0.5	0.1	0.9
η_1	Threshold indicating successful iteration	0.1	0.1	0.5
η_2	Threshold indicating very successful iteration	0.8	0.5	0.9

Our initial solver design has 50 design points. It arises from three stacks of an NOLH design over the four solver factors with three stacks, for a total of 49 design points, augmented with an additional design point representing the default ASTRO-DF variant.

Two other ASTRO-DF hyperparameters are also worth mentioning. One is $\lambda_{\min}$, which denotes the coefficient for the growing sequence of sample size lower bounds at each iteration. (This is not to be confused with the problem parameters λ_i representing the mean demands for products.) The other hyperparameter of interest is T, the solver's total budget of simulation replications. For a fixed T, ASTRO-DF will tend to complete fewer iterations if $\lambda_{\min}$ is large because it will sample more at each evaluated solution.

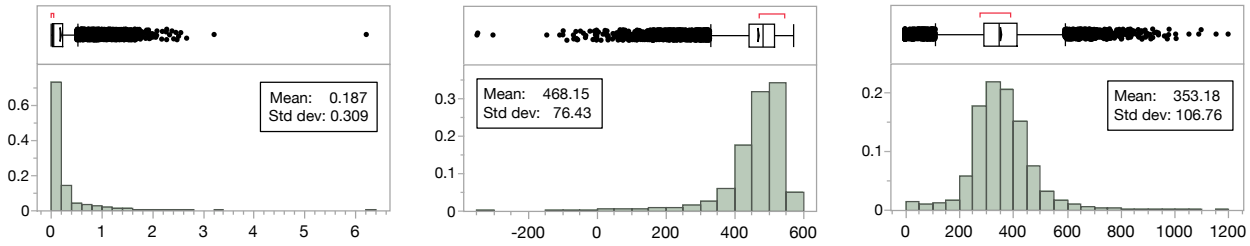
Why might the analyst decide to change solver hyperparameters from their default settings? Depending on the problem instance, tuning can lead to the solver recommending better solutions or solving the problem faster. For example, if the value of γ_1 was changed from the default of 1.5 to 3, the expansion of the trust-region would be twice as large. This could help ASTRO-DF reach a near-optimal solution in less time. But it could also lead the solver, when it struggles to build good approximating models due to local structure and noise behavior, to take a large number of iterations—and hence simulation replications—to adequately reduce the approximating region and improve the quality of the local model. Note, the lower bound of 1 for γ_1 corresponds to the trust region not expanding at all after a very successful iteration. In the case of η_1, increasing from the default value of 0.1 to 0.5 would mean that solutions must show a 50% relative improvement over the approximation model to be considered successful rather than the 10% relative improvement used in the default version of the solver. This means that the solver is much more strict about how much of an improvement ratio constitutes a successful iteration. Depending on the noise level and local structure (such as the local Lipschitz constant), which could both depend on the problem factors, it could be beneficial to use a solver that prioritizes small or vast improvements in the solutions. It is also possible that when changing the solver hyperparameters, interactions may come into play. For example, when using a large γ_1 it may be beneficial to use a small η_1 and vice versa.

5 EXPERIMENTS AND ANALYSIS

Our initial experiment involves crossing two designs: the 49 design-point NOLH for the problem factors, and the 50 design-point design for the solver factors, for a total of 2450 design points, each involving 10 macroreplications.

The primary questions specialists seek to answer are: (1) What solver instance(s) should I use for solving the daily procurement problems? (2) Should the choice of solver instance depend on the problem factors observed at the beginning of the day? and (3) What is my expected profit? However, as part of the knowledge discovery from sifting through and visualizing the farmed data, other types of insights may arise about the problem, the solver, or interactions between the two. For instance, we gain a great deal of information about the small-sample performance of ASTRO-DF on this problem that previously was not available. Also, this is a new problem in the SimOpt suite and it gives us an opportunity to subject it to a massive sensitivity analysis for verification and validation purposes.

First, we consider the spread of some key performance metrics based on the confirmation runs performed across the entire experiment. Figure 1 summarizes the spread of values for the terminal relative optimality gap $\hat{\nu}_1(T)$, the objective function value (i.e., estimated expected daily profit), and the cost of raw materials purchased in the morning across all macroreplications and all design points. All three histograms are skewed, the most pronounced being that of the $\hat{\nu}_1(T)$, where values greater than 1.0 indicate that the final solution is worse than the original solution on that macroreplication. Unsurprisingly, these same macroreplications tend to yield recommended solutions with lower objective function values, and together indicate a potential problem we delve into in more detail below. Figure 1b shows that on very few macroreplications, ASTRO-DF ultimately recommends solutions that are unprofitable. The distribution in Figure 1c shows that the total cost of purchasing materials at the beginning of the day is always less than the newsvendor's budget of $b = \$1200$. This is good because it means the initial budget was not violated on any macroreplications—as we intended when we chose a large value for b because SimOpt's current implementation of ASTRO-DF cannot handle linear constraints.



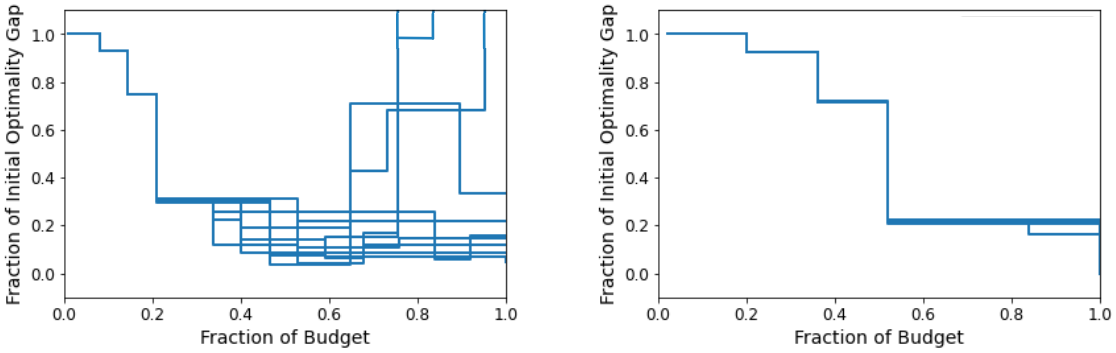
(a) Terminal relative optimality gap. (b) Recommended objective value. (c) Initial material order cost.

Figure 1: Relative frequency histograms and boxplots of key performance metrics for the generalized newsvendor problem.

It is also instructive to look at the variability of some of these performance metrics over individual design points. While often only end-of-run data are available for this, SimOpt reports the entire normalized progress curves for each macroreplication associated with each solver-problem pair. Figure 2 shows two such plots: Figure 2a reveals problematic behavior for a variant of ASTRO-DF, as the curves for several of the macroreplications initially drop down but then increase. Figure 2b shows the results from rerunning this design point after increasing $\lambda_{\min}$ to 10; clearly, there is much less variability among the macroreplications, and all the progress curves are monotonically nonincreasing.

Trellis plots can also be informative. Figure 3 shows two views of the general progress made by variants of ASTRO-DF with $\lambda_{\min} = 10$. In Figure 3a, each subplot shows the average objective function value (i.e., average expected profit) for each solver over time, where the average occurs over macroreplications and problem instances. Note that all subplots start at the same initial value, which is the average profit associated with the initial solution of $x_0 = (20, 20, 20, 20)$. A few different patterns emerge. Most often, the objective value increases steadily and rapidly by iteration, but in some instances the overall gain is more modest. Another recurring pattern shows the average decreases for the last several iterations: this might be partly due to different numbers of iterations within ASTRO-DF for different macroreplications, as well as curves such as those in Figure 2a where the objective value gets worse over time. In Figure 3b, curves of the average objective for each macroreplication are superimposed for each problem and averaged over all solvers. We see that the macroreplications have similar behavior. If, for example, one set of random number seeds was particularly challenging for all problems, we would see the curve for that macroreplication begin to deviate below the others as the iteration increased. The starting values differ greatly, unlike in Figure 3a. This makes sense because the costs and prices, and hence the objective functions, differ by problem.

Of all the runs, 14,376 (59%) of them were nonmonotonic when $\lambda_{\min} = 4$, whereas 2301 (9%) were nonmonotonic when $\lambda_{\min} = 10$. Monotonicity is not guaranteed: while ASTRO-DF and other solvers



(a) Curves for solver 11(a) with $\lambda_{\min} = 4$; several show very erratic behavior near the end of the budget.

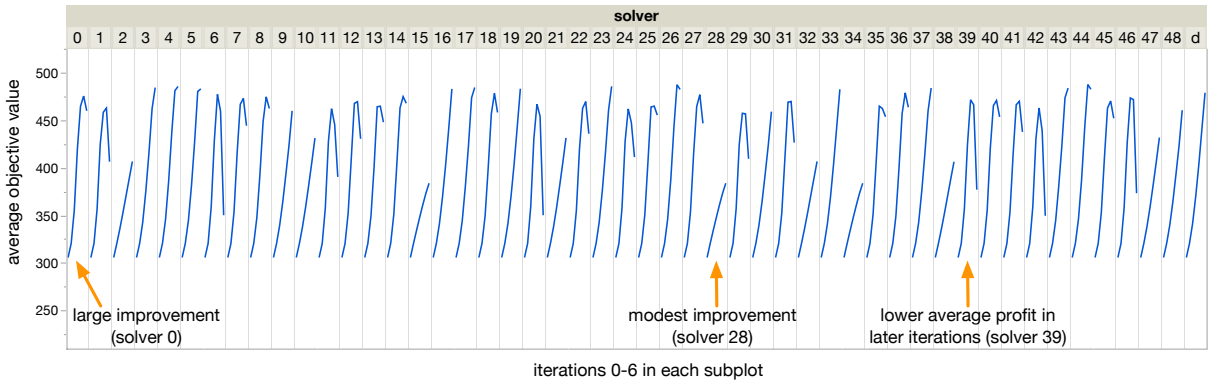
(b) Curves for solver 11(b) with $\lambda_{\min} = 10$ are more consistent and progress is monotonic.

Figure 2: Progress curves for ten macroreplications on problem 3 for solver 11 with two values of $\lambda_{\min}$: (a) $\lambda_{\min} = 4$ as originally specified and (b) $\lambda_{\min} = 10$.

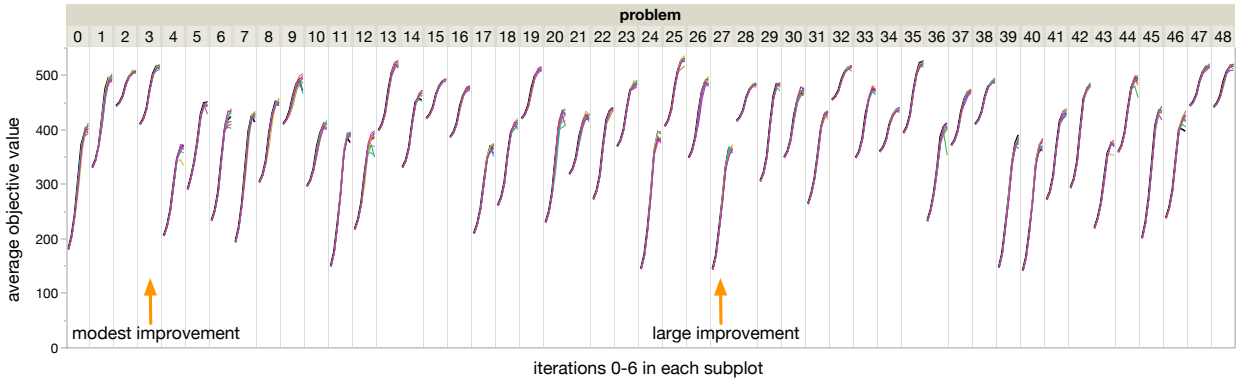
implemented in SimOpt report a new recommended solution only if its objective function value estimates show an improvement, the confirmation runs may change the estimates enough so that the last recommended solution does not necessarily have the best estimated objective function value. However, we anticipate this to happen rarely unless two or more recommended solutions have similar true expected profit, or if the solver was using a very low sample size to decide whether an improvement had been obtained. However, severe nonmonotonicity (i.e., $\hat{\nu}_1(T) > 1$) is problematic. Further inspection of the solutions for the troublesome curves in Figure 2a revealed that all had at least one raw material purchase of the minimum possible quantity (0.01), even if this was far from the solutions recommended at earlier iterations. 10% of the macroreps with $\lambda_{\min} = 4$ had this problem, but these were spread across the design points. 55% of the design points had all 10 good macroreplications, while 92% had at least 7 and all had at least 2. The results were better when $\lambda_{\min} = 10$; here, 95% of the design points had all 10 good macroreplications, 98% had at least 9, and all had at least 5. This led us to delve into the ASTRO-DF source code, where we found that a remnant function call from an obsolete version of ASTRO-DF was being called sporadically. This highlights a useful side benefit of a large-scale data farming experiment: we were able to identify and fix a coding error quickly (model verification), even though the bug might have gone unobserved or undetected with a smaller experiment, different solver, or different problem; in fact, up to this point, it had!

Metamodels of various types (e.g., stepwise regression, partition trees) for predicting the $\hat{\nu}_1(T)$ were of limited use for customizing the solver based on the problem specific costs available to the newsvendor in the morning. The solver factors γ_1, γ_2, and $\lambda_{\min}$ were most important when predicting $\hat{\nu}_1(T)$, but adding in the problem-specific factors did not improve the predictions. Consequently, for this newsvendor example, the specialist can pick a single solver for daily use based on their attitudes toward risk and their daily solve budget. There are interactions between solver factors that affect performance. For example, with low values of γ_1 ($\gamma_1 < 1.375$), the mean $\hat{\nu}_1(T)$ is better when $\lambda_{\min} = 4$, while for larger γ_1, the specialist should use $\lambda_{\min} = 10$ if they seek to improve (i.e., decrease) the mean and variability of $\hat{\nu}_1(T)$ (Figure 4a). Other criteria are possible: a specialist interested in a solver that meets solvability thresholds quickly would prefer low values of γ_1 if $\lambda_{\min} = 4$, but should avoid low values of γ_1 if $\lambda_{\min} = 10$ (Figure 4b).

To illustrate this in more detail, we focus on a few solver-problem pairs of potential interest. The solver instances are 0 and 28, and they correspond to a large and modest overall improvement in the solution value, respectively, in Figure 3a. We also consider two variants: 0a and 28a use $\lambda_{\min} = 4$, while 0b and 28b use $\lambda_{\min} = 10$. We picked three different problem instances: 5, 9, and 25. In an attempt to see if changing the underlying cost structure of the problem might reveal a situation where customizing the solver based on costs would be beneficial, we consider three variants of each problem. Problems 5, 9, and 25



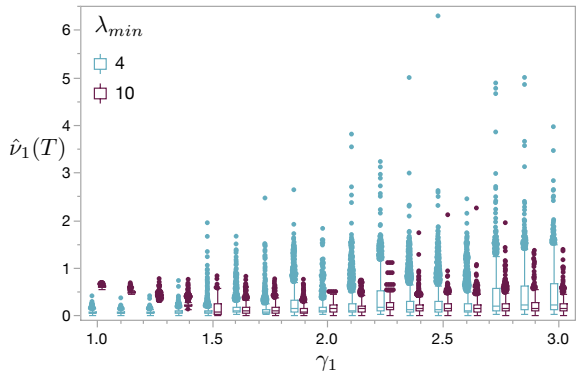
(a) Averages for a particular solver, where the averages are taken over all macroreplications and all problems.



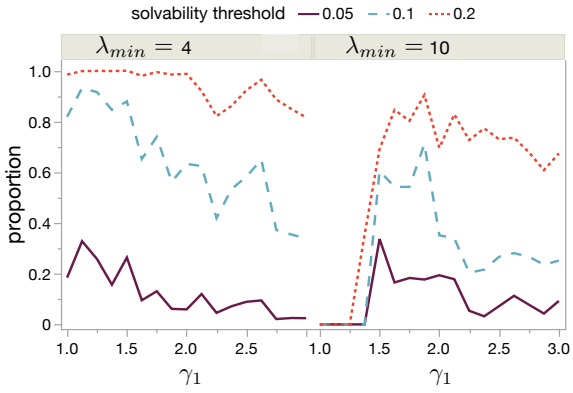
(b) Averages for each macroreplication of a particular problem, where the averages are taken over all solvers.

Figure 3: Objective function value improvement, by iteration, for ASTRO-DF solvers with $\lambda_{\min} = 10$.

use the original costs and ratios described earlier. Problems 5a, 9a, and 25a cut the sales price of Product 3 in half to a value of 10 as part of a price war, changing the most profitable item to the least profitable. Finally, Problems 5b, 9b, and 25b include the reduced sales price for Product 3 but also drop the salvage price of Material 4 to \$0.00; this might correspond to a situation where the usual salvage customer for this raw material calls in sick or is otherwise known to be unavailable. In Figures 5 and 6 we consider the variation of the objective function values, as well as that of the recommended optimal solutions, for our original SimOpt budget of $T = 500$ for each solver-problem combination, and a budget b 10 times as large. For the smaller SimOpt budget, Figure 5a shows that the average objective function value of the recommended solution rises much more rapidly for Solver 0 than for Solver 28. The specialist might prefer using solver 0 because it is likely to be more profitable, on average. However, this higher average expected profit may not be easy to achieve in practice. The first-stage solution values vary dramatically across macroreplications. With this small SimOpt budget, the individual runs are much less likely to converge to the same local solution—and there is no guarantee that averaging locally good solutions is a reasonable strategy. Consequently, the specialist might prefer using Solver 28 if they want to be more certain about the results. Solver 0 has higher reward but higher risk than Solver 28. Figure 6 shows how the relative risk and reward change with the SimOpt budget. In Figure 6a we see that both solvers eventually achieve the same average objective function values. Figure 6b shows that the solvers are converging to much more similar locations in the solution space as well; note that the scale on the y-axis is much more compact than for Figure 5b, and the curves for the two solvers are much more similar for all materials and all problem



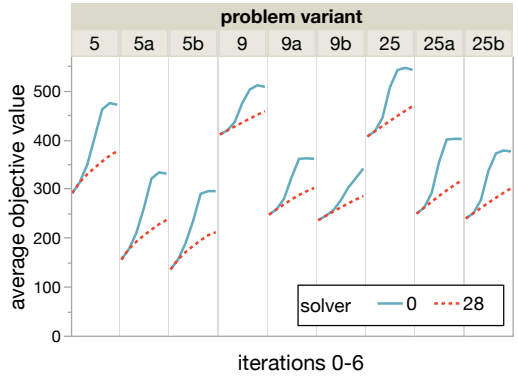
(a) Low values of the terminal relative optimality gap $\hat{\nu}_1(T)$ are desirable. Boxplots show that when γ_1 is low, it is better to set $\lambda_{\min} = 4$ than to set $\lambda_{\min} = 10$ in terms of both the mean and variability.



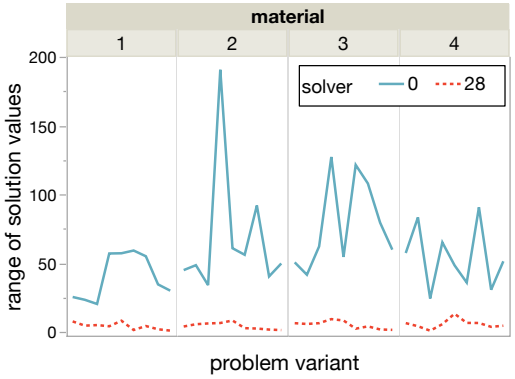
(b) High average solvability values are desired for all thresholds. Plots show that low values of γ_1 perform better when $\lambda_{\min} = 4$ but are unsuitable when $\lambda_{\min} = 10$.

Figure 4: Tradeoffs involving the choice of solver across all problem instances.

variants. This also brings an important insight to the generalist on which solver factor settings may be more robust to a larger range of simulation budget (and hence computation time) choices.



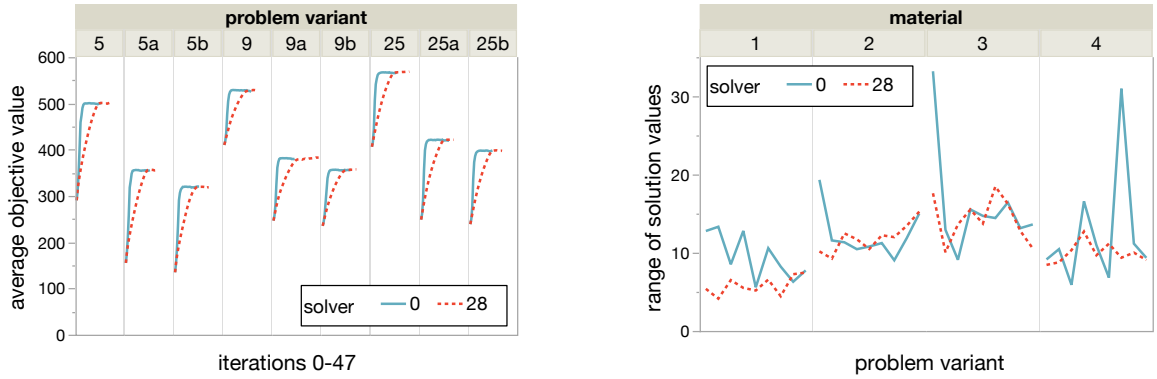
(a) The average objective function value of the recommended solutions rise more slowly for Solver 28 than for Solver 0.



(b) The morning purchase solutions for all four raw materials are much more variable for Solver 0 than for Solver 28.

Figure 5: Variation in average objective function values and ranges of recommended optimal solutions for two solvers and nine problem variants with a SimOpt budget of 500.

We return to the larger experiments to quantify the benefits of repeatedly tuning solver hyperparameters, rather than picking a particular solver and using it exclusively. Suppose that the 49 different problems represent the only possible cost configurations in future days. That is, the specialist will face one of these alternatives each morning. If the specialist were to choose to ignore the cost configuration variations, they might choose to use Solver 9a (the center point solver with $\lambda_{\min} = 4$) or Solver 9b (the center point solver with $\lambda_{\min} = 10$). If so, the expected objective function value (i.e., profit) for that day will differ from that computed using nominal costs. We used CRN to conduct 200 confirmation runs of the solutions reported for 49 newsvendor problems using Solvers 9a and 9b, and compared them to the best outcomes from the larger experiments. The results are summarized in Figure 7. Customizing

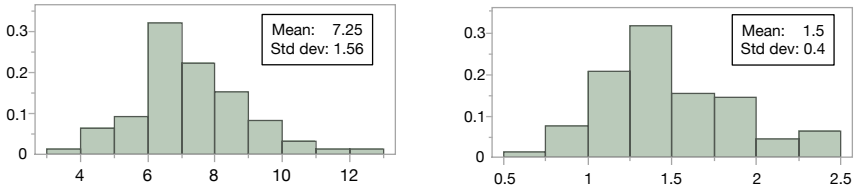


(a) Solver 28 and Solver 0 eventually achieve similar objective function values.

(b) The variability in morning purchase solutions is greatly reduced from that in Figure 5.

Figure 6: Variability of the recommended optimal solutions for two solvers and three variants of three problems with a budget of 5000. Both solvers yield similar solutions at the end of this expanded budget.

the solver choice never resulted in a lower expected daily profit: the benefit estimates range from \$3.75 to \$12.17 with an average of \$7.25 (Figure 7a). This corresponds to a very small but statistically significant (p-value< 0.0001) benefit in terms of the percent improvement in profit, ranging from 0.7% to 2.5% (Figure 7b). Nonetheless, this comes at essentially no additional cost—with such a small number of potential cost configurations, a table lookup could be used if running the solver takes too long or is not an option on a particular day. We also remark that no single solver dominated: across the 49×2 situations explored, 36 different solvers were used at least once and 17 solvers were used three or more times.



(a) Average improvement (\$).

(b) Percent improvement.

Figure 7: Relative frequency histograms illustrate the benefit of selecting solver hyperparameters based on problem factors.

6 CONCLUSION

In this paper, we have motivated the tuning of SO solvers by specialists who are faced with repeatedly solving similar problems. We propose the specialist first make use of data farming experiments in an offline setting to explore tradeoffs among potential solvers. At the end of this stage, the specialist might decide to use a single solver on a regular basis, or might decide to customize the solver by using different hyperparameters based on the current problem factors.

We demonstrated this approach for a generalized newsvendor problem, where the specialist's daily activity involves ordering raw materials in the morning based on currently available price information, with the possibility of a secondary order once the day's demand for multiple products requiring these raw materials becomes known. Our results provide some initial guidance for the newsvendor problem.

Along the way, our data farming experiments uncovered a coding error that occasionally led to problematic output, but might have gone unnoticed in a smaller study. This is often an important side benefit of data farming experiments, because identifying bugs and interesting behavior early on accelerates the verification and validation process, and may suggest research questions worthy of further investigation. One issue we intend to study in the future is to identify situations where metamodels involving the problem characteristics can be used to set solver hyperparameters to achieve good finite-sample behavior while using a relatively efficient experiment design. Achieving this may depend on how easy or hard it is to solve a particular problem variant, or the choice of different stopping conditions.

ACKNOWLEDGMENTS

This work was partially supported by National Science Foundation grants CMMI-2226347 and CMMI-2206972. The authors also thank Yunsoo Ha for help debugging the code issue discovered in the paper.

REFERENCES

Eckman, D. J., S. G. Henderson, and S. Shashaani. 2023. "Diagnostic Tools for Evaluating and Comparing Simulation-Optimization Algorithms". *INFORMS Journal on Computing* 35(2):350–367.

Eckman, D. J., S. G. Henderson, S. Shashaani, and R. Pasupathy 2020. "Simulation Optimization Library". http://github.com/simopt-admin/simopt. Accessed 1st May 2020.

Ha, Y. and S. Shashaani. 2024. "Iteration Complexity and Finite-Time Efficiency of Adaptive Sampling Trust-Region Methods for Stochastic Derivative-Free Optimization". *IISE Transactions* (just-accepted):1–15.

Huang, C., Y. Li, and X. Yao. 2019. "A Survey of Automatic Parameter Tuning Methods for Metaheuristics". *IEEE Transactions on Evolutionary Computation* 24(2):201–216.

Kim, S., R. Pasupathy, and S. G. Henderson. 2015. "A Guide to Sample Average Approximation". In *Handbook of Simulation Optimization*, edited by M. C. Fu, Chapter 8, 207–243. New York: Springer.

Sanchez, S. M. 2020. "Data Farming: Methods for the Present, Opportunities for the Future". *ACM Transactions on Modeling and Computer Simulation* 30(4):Article 22. 1–30.

Shashaani, S., D. Eckman, and S. Sanchez. 2024. "Data Farming the Parameters of Simulation-Optimization Solvers". *ACM Transactions on Modeling and Computer Simulation* 34(4):1–29.

Shashaani, S., F. S. Hashemi, and R. Pasupathy. 2018. "ASTRO-DF: A Class of Adaptive Sampling Trust-region Algorithms for Derivative-free Stochastic Optimization". *SIAM Journal on Optimization* 28(4):3145–3176.

AUTHOR BIOGRAPHIES

NICOLE FELICE is an incoming Ph.D. student of the Operations Research Graduate Program at North Carolina State University. Her research interests are modeling and optimization of stochastic systems and combining those with advanced computational practices. Her email address is ndfelice@ncsu.edu.

SARA SHASHAANI is an Assistant Professor and Bowman Faculty Scholar in the Edward P. Fitts Department of Industrial and System Engineering at North Carolina State University. Her research interests are simulation optimization and probabilistic data-driven models. She is a co-creator of SimOpt library. Her email address is sshasha2@ncsu.edu and her homepage is https://shashaani.wordpress.ncsu.edu/.

DAVID J. ECKMAN is an Assistant Professor in the Wm Michael Barnes '64 Department of Industrial and Systems Engineering at Texas A&M University. His research interests deal with optimization and output analysis for stochastic simulation models. He is a co-creator of SimOpt, a testbed of simulation optimization problems and solvers. His e-mail address is eckman@tamu.edu.

SUSAN M. SANCHEZ is a Distinguished Professor Emerita in the Operations Research Department at the Naval Postgraduate School. Her research interests include the design and analysis of large-scale simulation experiments, data farming, robust design, and applied statistics. She has been an active member of the simulation community for many years, co-founded NPS's Simulation Experiments & Efficient Designs (SEED) Center for Data Farming, and has been recognized as a Titan of Simulation and an INFORMS Fellow. Her web page is http://faculty.nps.edu/smsanche/ and her email is ssanchez@nps.edu.