Proceedings of the 2019 Winter Simulation Conference
N. Mustafee, K.-H.G. Bae, S. Lazarova-Molnar, M. Rabe, C. Szabo, P. Haas, and Y.-J. Son, eds.

REDESIGNING A TESTBED OF SIMULATION-OPTIMIZATION PROBLEMS AND SOLVERS FOR EXPERIMENTAL COMPARISONS

David J. Eckman

Department of Ind Eng and Mgmt Sciences
Northwestern University
2145 Sheridan Road
Evanston, IL 60208, USA

Shane G. Henderson

School of Operations Research and Info Eng
Cornell University
206 Rhodes Hall
Ithaca, NY 14853, USA

Raghu Pasupathy

Department of Statistics
Purdue University
250 N. University St
West Lafayette, IN 47907, USA

ABSTRACT

We describe major improvements to the testing capabilities of SimOpt, a library of simulation-optimization problems and solvers. Foremost among these improvements is a transition to GitHub that makes SimOpt easier to use and maintain. We also design two new wrapper functions that facilitate empirical comparisons of solvers. The wrapper functions make extensive use of common random numbers (CRN) both within and across solvers for various purposes; e.g., identifying random initial solutions and running simulation replications. We examine some of the intricacies of using CRN to compare simulation-optimization solvers.

1 INTRODUCTION

SimOpt (SimOpt Library 2019) is a public repository of simulation-optimization problems and solvers, created to address the lack of a testbed for comparing the empirical performance of solvers (Pasupathy and Henderson 2006; Pasupathy and Henderson 2011). Over the years, the number of problems and solvers in the library has grown considerably—it presently consists of about 50 problems and 10 solvers. The types of problems vary in terms of the nature of their decision variables (continuous, discrete, or categorical) and constraints (deterministic, stochastic, or no constraints). We aim to grow the SimOpt library to be representative of the wide variety of simulation-optimization problems and solvers, so that for any type of solver, there are several problems to test it on and solvers to test it against.

The SimOpt problem library is composed of optimization problems for which some form of stochastic simulation is used to evaluate the objective function at a given solution. A representative example is the problem of positioning ambulance bases within a city with the objective of minimizing the expected call response time. For a given arrangement of ambulance bases, a day's worth of operations are simulated by generating emergency calls according to a Poisson process and dispatching ambulances accordingly. Problems of this kind resemble those on which simulation-optimization solvers are used in practice. For many problems in SimOpt, including this one, there is little (if any) information about the structure of the

objective function, e.g., Lipschitz continuity, smoothness, or convexity. Moreover, the optimal solutions are in most cases unknown.

Unlike libraries for deterministic-optimization problems, e.g., CUTEst (Gould et al. 2015; RAL Computational Mathematics 2019), SimOpt includes few problems for which the objective function has an explicit functional form. Challenging deterministic-optimization problems, such as Rosenbrock's function (Rosenbrock 1960), can be converted into stochastic-optimization problems by adding a mean-zero random error when evaluating the objective function. However, we consider such synthetic problems to be ill-suited for testing simulation-optimization solvers. Our main concern is how simulation-optimization solvers behave on such problems when common random numbers (CRN) are used to evaluate the objective function at different solutions. If the random error is additive and has a common variance for all solutions, using CRN in a sample-average approximation scheme has the effect of shifting the objective function up or down while leaving the optimizer unchanged. The optimization problem thereby degenerates into a deterministic one.

In soliciting new problems for the SimOpt library, a few properties are especially desirable. First, problems should be operations research-related and incorporate simple simulation models. Second, problems with known structural properties or known optimal solutions would be appreciated as such problems are currently underrepresented. Third, high-dimensional problems would be a valuable addition to the library, since most of the current problems have fewer than ten dimensions (decision variables). In addition to being more representative of practical problems, high-dimensional problems are crucial for testing how the performance of a solver scales.

The SimOpt solver library includes direct-search methods (e.g., Nelder-Mead by Barton and Ivey Jr. 1996 and the procedure of Anderson and Ferris 2001), gradient-based methods (e.g., stochastic approximation Kiefer and Wolfowitz 1952), and metamodel methods (e.g., STRONG by Chang et al. 2013). Most of the solvers in SimOpt are for simulation optimization over continuous decision variables; we welcome more solvers designed for discrete simulation optimization. One class of methods not present in the library is ranking-and-selection (R&S) procedures, which are designed for problems for which the feasible region is finite and all solutions can be simulated to some degree within the available budget (Hong et al. 2015). We are also looking to add heuristic methods, such as simulated annealing (Branke et al. 2008).

Users may download the source code for all problems and solvers, the majority of which is written in MATLAB. While other fields have been shifting towards Python, R, and Julia for their code libraries, e.g., Dlib (King 2019), we have kept MATLAB as the primary language for SimOpt. Besides the legacy cost of translating the library to a different language, another reason we continue to use MATLAB is the detailed control over random number generation that it offers. Managing random number streams and substreams is essential to using CRN within and across solvers. Furthermore, MATLAB software is widely used in engineering departments and academic licenses are heavily subsidized.

In other optimization fields, testbeds have been instrumental in identifying the strongest solvers and directions for future research. Examples include Scikit-Optimize (Skopt 2019) for Bayesian optimization, Global Optimization Software (Neumaier 2019) for global optimization, and ALGLIB (ALGLIB Project 2019) for numerical optimization. Another recent effort to develop cyberinfrastructure for operations research methods is the cORe project (Deng et al. 2019). The simulation-optimization community lags behind the deterministic-optimization community when it comes to having an established testbed for comparing solvers (Pasupathy and Henderson 2011; Dong et al. 2017). In particular, it is crucial that the simulation-optimization community develops a consensus on which solvers work well for different types of problems.

While many simulation-optimization solvers provide guarantees on their asymptotic performance (e.g., proofs of convergence, with or without rates), there has been far less empirical study of their finite-time performance. Dong et al. (2017) made a preliminary effort in this direction by comparing a handful of SimOpt's solvers on a subset of problems. Experimental comparisons of this sort can also help with tuning

a solver's internal parameters so that it performs well across different problems. The purpose of SimOpt is to provide such a testbed to the simulation-optimization community.

The website version of SimOpt is unfortunately more of a library than a testbed: the resources are readily available, but it is incumbent on the user to setup and run experiments. This paper describes our ongoing efforts to redesign SimOpt to improve its usability and maintenance. A major effort in our redesign is a move to GitHub, as discussed in Section 2. In Section 3, we present a new testing environment—built around two easy-to-use wrapper functions—for running experiments on SimOpt. In Section 4, we describe the schema for how CRN are implemented in the wrapper functions. In Section 5, we conclude with future opportunities for expanding and enhancing SimOpt.

2 SIMOPT ON GITHUB

The website version of SimOpt has a number of limitations that make it difficult to use and maintain:

- **File Management:** Source code files must be downloaded one at a time and then manually saved in the user's local directory. It is also important that the user adopt a particular folder structure, as the solver and wrapper functions refer to the relative paths of the files.
- **Synchronization:** The user cannot easily track changes to the source code files and synchronize their local copies with the latest versions. The only way for the user to update their local copy is to delete it and download the latest version from the website.
- **Submission:** The process for submitting new problems and solvers is handled offline. This often entails emailing files back and forth between the contributer and the core development team before a final version of the code is approved and posted on the website.
- **Experimentation:** The existing wrapper function is extremely limited, allowing the user to run only one macroreplication of one solver on one problem. Furthermore, the wrapper function only prints output to a text file, leaving the user responsible for aggregating and plotting the data.

These shortcomings prevent SimOpt from fulfilling its intended purpose as a convenient platform for testing solvers. The first three are addressed by transitioning the SimOpt repository to GitHub—the fourth is addressed by designing new wrapper functions, described in Section 3.

Web-based platforms such as GitHub (GitHub 2019) and Bitbucket (Atlassian 2019) are commonly used for managing team coding projects in both industry and academia. These online services also offer free user accounts for public repositories and various licensing options. GitHub's web interface has several convenient features that make it an appealing alternative to the website version of SimOpt. Linked wiki pages can be used to guide the reader through a project or tutorial, while README files offer another way to display helpful information and documentation. In addition, many file types, including PDFs and source code, can be directly viewed in the web browser, allowing users to skim through project files without having to download and open them in an appropriate editor. GitHub even colors syntax for many programming languages, including MATLAB.

The greatest benefit to transitioning the SimOpt repository to GitHub is the use of Git, a version control system that allows users to track and manage changes to files over time. Under the Git version control system, changes to the source code for problems and solvers can be managed and bugs can be easily fixed by reverting changes. Another useful feature of Git is the ability to create "branches," i.e., parallel versions of a repository. Branches allow the user to make changes to a repository without disrupting the master branch and later merge these changes. This feature is especially important for integration testing of new problems and solvers before incorporating them into SimOpt.

Researchers interested in running experiments on only the latest version of SimOpt can "clone" (copy) the repository to use for offline testing. Researchers wishing to keep up with changes to SimOpt—or contribute to it—can instead "fork" the repository, giving them a linked version on which to work. The forked repository can be easily synchronized with the main repository by "pulling" changes; see GitHub

Help (2019b) and GitHub Guides (2017) for helpful tutorials on forking repositories. In either case, the cloned or forked repositories share the same folder structure as the main repository, therefore any functions that use relative paths (e.g., a solver calling a problem function) will be unaffected. Researchers can also improve the reproducibility of their experiments by referencing a seven-digit hexidecimal string corresponding to the latest commit to the repository at the time their experiments were performed.

The move to GitHub also streamlines the process for submitting new problems and solvers:

1. The user creates files for new problems and/or solvers on their forked repository and then commits and pushes these changes to their remote copy.
2. Using the SimOpt GitHub web interface, the user submits a "pull request" for the proposed changes; see GitHub Help (2019a) for an introduction to pull requests. The pull request opens a dialog between the user and the core development team for discussing the proposed changes. Messages sent in the pull request dialog can also be set to appear in email notifications.
3. In the pull request dialog, the user and the core development team make line-item changes to the code to ensure that it works properly.
4. If the proposed change is approved by the core development team, it is merged with the master branch and becomes accessible to all users.

3 RUNNING EXPERIMENTS IN SIMOPT

Comparing the performance of simulation-optimization solvers poses several challenges not present in comparisons of deterministic-optimization solvers (Dong et al. 2017). Since the optimal solution to a simulation-optimization problem is typically unknown, a solver cannot be terminated when it identifies a solution that is within some neighborhood of the optimal solution or when the optimality gap drops below a specified threshold. Furthermore, simulation-optimization problems have no certificate of optimality to use as a stopping condition. One remedy is to fix a budget of simulation replications and track the performance of a solver until it exhausts the budget. For the performance criterion, we choose to measure the objective function value of the (random) estimated best solution found by a fixed time (Dong et al. 2017). This metric allows the progress of a solver to be plotted over time and compared against that of other solvers.

As an added challenge, the solutions visited by a simulation-optimization solver vary from run to run, even when starting from the same initial solution. This means that when observed over time, a solver's performance—in terms of the objective function value of the estimated best solution—is a stochastic process. On a given problem, statistics of the distribution of this stochastic process (e.g., the mean or quantiles over time) can provide meaningful information about how a solver performs on average or from run to run.

3.1 Experimental Design

We outline the process used to evaluate the performance of a solver on a given problem; for a more in-depth discussion, see Dong et al. (2017).

1. On a given problem, multiple macroreplications (i.e., runs) of a solver are performed. Each macroreplication entails running a solver until a fixed, problem-specific budget of simulation replications (i.e., objective function evaluations) is exhausted.
2. On a given macroreplication, the solver visits solutions, evaluates the objective function, and identifies new solutions until the budget is exhausted. At fixed points within the budget, the solver reports the solution that looks best—based on objective function evaluations—at that time, along with the sample mean and variance of its objective function evaluations.
3. After all macroreplications have been run, there is a post-processing step. At every macroreplication's recorded solutions, a fixed number of fresh simulation replications are run and the sample means of these objective function evaluations are calculated. The purpose of this step is to obtain more

precise estimates of the objective function values at the best-looking solutions while avoiding the optimization bias present in the data output by the solver.
4. At each budget point, summary statistics of the objective function estimates are calculated, e.g., the sample mean, median, and quantiles.
5. Plots are made depicting the objective function value of the best-looking solution over time.

In Step 2, instead of recording the best-looking solution visited thus far, a solver could return whichever solution it would recommend to the user if it had to stop then. For example, a stochastic approximation solver using Polyak-Ruppert averaging (Ruppert 1988; Polyak 1990) might return the average of the visited solutions, even if that solution has not been evaluated.

For most simulation-optimization solvers, its performance at an intermediate budget point can be interpreted as the solver's performance if it were given that smaller budget and run to completion. Yet for some solvers, this is not the case. For example, the SPSA solver (Spall 1992; Spall 1998) uses the total budget to set a parameter that determines its gain sequence, hence the solutions visited by the solver over time depend on the total budget. Rather than run such solvers for many different total budgets, we choose to acknowledge this difference in how a solver's plotted performance should be interpreted.

3.2 Wrapper Functions

Experiments on the SimOpt repository are performed using wrapper functions that call user-specified solver and problem functions. The wrapper function on the website version of SimOpt runs only one macroreplication of one solver on one problem and records some statistics in a text file. This setup makes it hard for the user to aggregate the output from multiple macroreplications and visually compare the performances of multiple solvers. To fix these shortcomings, we create two new wrapper functions (`RunWrapper.m` and `PlotWrapper.m`) that allow the user to run multiple macroreplications of multiple solvers on multiple problems and plot the results. In relation to the experimental design outlined in the previous section, `RunWrapper.m` performs Steps 1–2 while `PlotWrapper.m` performs Steps 3–5. Of the two wrappers, `RunWrapper.m` constitutes the bulk of the computational time of an experiment, because it involves running the solvers multiple times, whereas `PlotWrapper.m` runs a much smaller number of simulation replications.

1. **`RunWrapper.m`:** The user inputs an array of solver names, an array of problem names, and a common number of macroreplications for running each solver on each problem. Macroreplications of each solver-problem pair are run and the outputs (the best-looking solutions and the mean and variance of their objective function evaluations) are saved in MATLAB workspace files in a dedicated folder. (MATLAB workspace files allow the user to recreate the MATLAB environment as if the experiment had just been run; i.e., all of the variables are loaded into the workspace.)
2. **`PlotWrapper.m`:** The user inputs an array of solver names, an array of problem names, and a common number of replications to run at each recorded solution. For each solver-problem pair, the corresponding MATLAB workspace file output by `RunWrapper.m` is loaded into the MATLAB environment. If no such file is present, an error message is printed to the MATLAB terminal, directing the user to first call `RunWrapper.m`. At each of the solutions reported at the budget points across the macroreplications, independent simulation replications are run and the sample mean is recorded. For each problem, two plots are made: One plot shows the mean objective function value of the (random) best-looking solution visited by each solver at each budget point, along with 95% confidence intervals. The other plot shows the median objective function value of the best-looking solutions over time with 0.25 and 0.75 quantiles. The plots are displayed on the screen and saved as MATLAB figure (.fig) files in a dedicated folder.

`PlotWrapper.m` uses the identities of the best-looking solutions recorded by `RunWrapper.m`, but not the means and variances of their objective function evaluations. Instead these values are reported for the benefit of researchers interested in using them for other purposes, e.g., analyzing the optimization bias of a solver.

3.3 Demonstration

After cloning or forking the repository and setting the MATLAB path to the *Experiments* folder, both wrappers can be run from the command line in the MATLAB terminal. As an example, the command

```
RunWrapper({'SAN'},{'RANDSH','ANDFER'}, 30)
```

runs 30 macroreplications of the random-search and Anderson-Ferris solvers on a stochastic activity network (SAN) problem (Avramidis and Wilson 1996; Fu 2015). Likewise, the command

```
PlotWrapper({'SAN'},{'RANDSH','ANDFER'}, 50)
```

runs 50 fresh simulation replications at each of the best-looking solutions reported by the two solvers and produces the two plots shown in Figure 1. Both commands show another of our updates to SimOpt: problem and solver names have been changed to abbreviations of up to six letters, making them easier to reference.

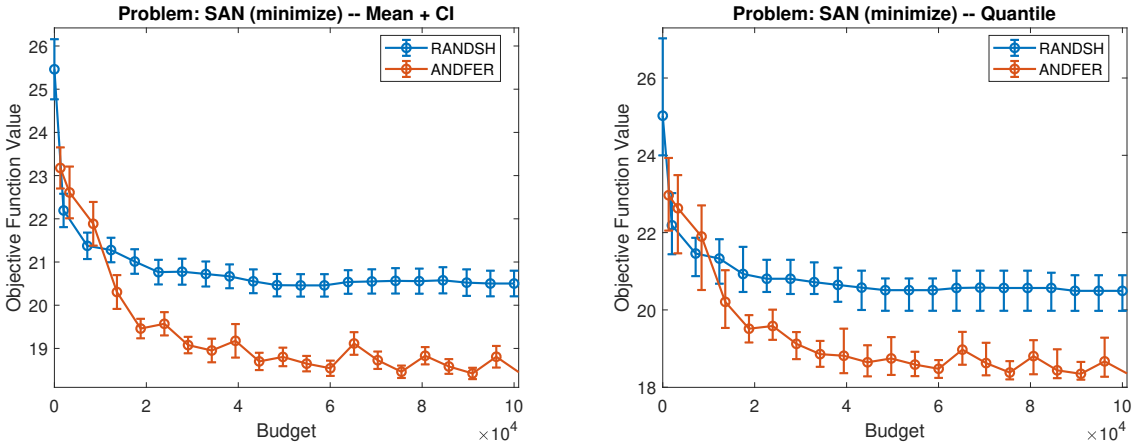

Figure 1: Performance of the random-search and Anderson-Ferris solvers on the SAN problem: (a) mean and 95% confidence intervals, (b) median and 0.25–0.75 quantiles.

Figure 1 shows how the Anderson-Ferris solver performs better than random search, making faster progress and finding better solutions after exhausting a budget of 10,000 simulation replications. Furthermore, a comparison of Figures 1(a) and 1(b) shows that the mean and median performance are very similar. In addition, it can be seen from the width of the quantile bars in Figure 1(b) that the performance of the Anderson-Ferris solver is more variable in the early stages.

4 COMMON RANDOM NUMBER MANAGEMENT

Random number management is a critical consideration when testing simulation-optimization solvers. MATLAB supports several random number generators (RNGs), including the `mrg32k3a` combined multiple recursive generator used in the SimOpt wrapper functions. Management of random number generation—creating random number streams and switching among them—is handled by the MATLAB `RandStream` constructor; see MathWorks (2019) for full details. In particular, MATLAB uses random number streams

and substreams—shorter, non-overlapping sequences of random numbers within a given stream. Random number seeds offer a finer level of control, but streams and substreams suffice for experiments on SimOpt.

The source code for the problems and solvers makes use of random numbers for three purposes:

- **Solver function:** Identifying solutions to evaluate (e.g., picking a random direction in SPSA).
- **Problem structure function:** Determining the random initial solution(s).
- **Problem function:** Running simulation replications at a given solution. This can consist of multiple sources, e.g., the random sequences of interarrival times and service times in a queueing network.

When comparing different solvers on the same problem, using CRN, i.e., the same random number streams and substreams, can help to reduce the variance of the difference in their estimated performances. However, care must be taken to ensure that CRN are implemented properly and that the resulting statistical analysis is valid. We now describe in detail the CRN schema implemented in the SimOpt wrapper functions. To elucidate the various applications of CRN, we refer to two arbitrary solvers: A and B.

4.1 CRN in RunWrapper.m

`RunWrapper.m` enables the use of CRN both within and across solvers, as depicted in Table 1.

- **Internal Randomness of a Solver:** On a given macroreplication, Solvers A and B use the same random number stream for any internal randomness. This use of CRN is for convenience; we do not expect different solvers to behave similarly if they share only this form of randomness. Some solvers (e.g., Nelder-Mead) have no internal randomness. Each macroreplication makes use of a dedicated stream that is reused across solvers.
- **Initial Solutions:** On a given macroreplication, Solvers A and B start at the same initial solution(s). A given solver may still start at different initial solutions on different macroreplications. Each macroreplication makes use of a dedicated stream that is reused across solvers.
- **Replications (Within Solvers):** Some solvers use CRN *across* solutions to reduce the variance of the difference in objective function estimates. For example, one version of the Nelder-Mead solver uses CRN across all solutions in the simplex (Barton and Ivey Jr. 1996). For a given macroreplication, a solver uses the same streams and substreams for replications taken at different solutions, but different substreams for replications taken at the same solution. Each macroreplication makes use of dedicated streams, but different substreams are used for the individual simulation replications. We make this the default setting for the solvers in the SimOpt library. If the user does not want a solver to use CRN across solutions, the solver's code can be easily modified by updating the substream counter after evaluating each solution.
- **Replications (Across Solvers):** On a given macroreplication, Solvers A and B use the same random number stream(s) to evaluate all solutions.

4.2 CRN in PlotWrapper.m

`PlotWrapper.m` also uses CRN when post-evaluating the solutions reported by `RunWrapper.m`, as depicted in Table 2.

- **Replications (Across Budget Points):** Across different budget points, the same streams and substreams are used to run the simulation replications. This approach is statistically valid because separate confidence intervals are constructed at each budget point. If a solver reports the same best-looking solution at different budget points, the same objective function evaluations will be generated. This explains why in Figure 1, the mean, median, and quantiles of the performance of the random-search solver are constant for the last few budget points.

Table 1: Common random number schema for `RunWrapper.m` for two solvers (A and B), each of which takes simulation replications at two solutions over two macroreplications. Each simulation replication requires two sources of randomness where. Sx.SSy denotes Substream y of Stream x.

		Internal	Initial	Solution 1	Solution 2
Solver A	Macrorep 1	(S1.SS1)	(S2.SS1)	Rep 1 (S3.SS1, S4.SS1)	Rep 1 (S3.SS1, S4.SS1)
				Rep 2 (S3.SS2, S4.SS2)	Rep 2 (S3.SS2, S4.SS2)
				Rep 3 (S3.SS3, S4.SS3)	Rep 3 (S3.SS3, S4.SS3)
				...	...
	Macrorep 2	(S5.SS1)	(S6.SS1)	Rep 1 (S7.SS1, S8.SS1)	Rep 1 (S7.SS1, S8.SS1)
				Rep 2 (S7.SS2, S8.SS2)	Rep 2 (S7.SS2, S8.SS2)
				Rep 3 (S7.SS3, S8.SS3)	Rep 3 (S7.SS3, S8.SS3)
				...	...
Solver B	Macrorep 1	(S1.SS1)	(S2.SS1)	Rep 1 (S3.SS1, S4.SS1)	Rep 1 (S3.SS1, S4.SS1)
				Rep 2 (S3.SS2, S4.SS2)	Rep 2 (S3.SS2, S4.SS2)
				Rep 3 (S3.SS3, S4.SS3)	Rep 3 (S3.SS3, S4.SS3)
				...	...
	Macrorep 2	(S5.SS1)	(S6.SS1)	Rep 1 (S7.SS1, S8.SS1)	Rep 1 (S7.SS1, S8.SS1)
				Rep 2 (S7.SS2, S8.SS2)	Rep 2 (S7.SS2, S8.SS2)
				Rep 3 (S7.SS3, S8.SS3)	Rep 3 (S7.SS3, S8.SS3)
				...	...

Table 2: Common random number schema for `PlotWrapper.m` for two solvers (A and B), each of which takes simulation replications at the best-looking solutions recorded at two budget points over two macroreplications. Each simulation replication requires two sources of randomness. Sx.SSy denotes Substream y of Stream x.

		Budget Point 1	Budget Point 2
Solver A	Macrorep 1	Rep 1 (S1.SS1, S2.SS1)	Rep 1 (S1.SS1, S2.SS1)
		Rep 2 (S1.SS2, S2.SS2)	Rep 2 (S1.SS2, S2.SS2)
		Rep 3 (S1.SS3, S2.SS3)	Rep 3 (S1.SS3, S2.SS3)
		...	...
	Macrorep 2	Rep 1 (S3.SS1, S4.SS1)	Rep 1 (S3.SS1, S4.SS1)
		Rep 2 (S3.SS2, S4.SS2)	Rep 2 (S3.SS2, S4.SS2)
		Rep 3 (S3.SS3, S4.SS3)	Rep 3 (S3.SS3, S4.SS3)
		...	...
Solver B	Macrorep 1	Rep 1 (S1.SS1, S2.SS1)	Rep 1 (S1.SS1, S2.SS1)
		Rep 2 (S1.SS2, S2.SS2)	Rep 2 (S1.SS2, S2.SS2)
		Rep 3 (S1.SS3, S2.SS3)	Rep 3 (S1.SS3, S2.SS3)
		...	...
	Macrorep 2	Rep 1 (S3.SS1, S4.SS1)	Rep 1 (S3.SS1, S4.SS1)
		Rep 2 (S3.SS2, S4.SS2)	Rep 2 (S3.SS2, S4.SS2)
		Rep 3 (S3.SS3, S4.SS3)	Rep 3 (S3.SS3, S4.SS3)
		...	...

- **Replications (Across Solvers):** Across solvers, the same streams and substreams are used to run the simulation replications. This use of CRN results in sharper comparisons of the solvers when plotting their performance curves. For instance, if Solvers A and B report the same best-looking solution on the same macroreplication, the same objective function evaluations will be generated.

By averaging the means of the objective function evaluations over the macroreplications performed by `RunWrapper.m`, we form a point estimate and confidence interval for the performance of the best-looking solution at a given budget point. Using CRN across the macroreplications would invalidate the confidence interval, because the means of the objective function evaluations would be dependent. For a given solver at a given budget point, `PlotWrapper.m` uses different streams for running simulation replications at the solutions recorded from each macroreplication, as shown in Table 2.

For both `RunWrapper.m` and `PlotWrapper.m`, there is an outer layer of CRN *across* problems. This is for convenience of coding, as it requires fewer streams. We expect that this reuse of random numbers will have little impact on the performance of a solver on different problems. Moreover, comparisons across problems are likely not of interest to the user.

5 CONCLUSION

We summarize ongoing developments to the SimOpt library, centered around a transition to a GitHub repository and the benefits that it brings. The most important benefits are better version control and a streamlined process for contributing new problems and solvers. To address the limited testing capabilities of the library, we design two wrapper functions that facilitate empirical comparisons. We discuss in detail some of the complexities of implementing CRN in this framework for comparing simulation-optimization solvers.

Among the future opportunities for enhancing SimOpt, we seek to add more problems, preferably those with known structure, known optimal solutions, and more than ten dimensions. We also wish to add more solvers to the library, especially those designed for discrete simulation optimization. Another potential expansion of SimOpt is adding a R&S testbed, however the testing framework presented in this paper may not be suitable for comparing R&S solvers. In particular, R&S solvers are typically evaluated based on their efficiency at delivering guarantees on the probability of selecting an optimal or near-optimal solution or the expected optimality gap. Since most problems in SimOpt have unknown optimal solutions, evaluating these metrics is problematic. We will also explore ways to enable the wrapper functions to interface with problems and solvers written in other programming languages.

ACKNOWLEDGMENTS

We thank Suvrajeet Sen for sharing with us the progress of the cORe cyber-infrastructure project and David Newton for his help with revising the source code for some of the SimOpt problems. This work is supported by the National Science Foundation under grants DGE-1650441, CMMI-1537394, and TRIPODS+X DMS-1839346 and by the Army Research Office under grant W911NF-17-1-0094.

REFERENCES

ALGLIB Project 2019. "Optimization (Nonlinear and Quadratic)". http://www.alglib.net/optimization, accessed 24th April.

Anderson, E. J., and M. C. Ferris. 2001. "A Direct Search Algorithm for Optimization with Noisy Function Evaluations". *SIAM Journal on Optimization* 11(3):837–857.

Atlassian 2019. "Bitbucket". http://bitbucket.org/, accessed 17th April.

Avramidis, A. N., and J. R. Wilson. 1996. "Integrated Variance Reduction Strategies for Simulation". *Operations Research* 44(3):327–346.

Barton, R. R., and J. S. Ivey Jr.. 1996. "Nelder-Mead Simplex Modifications for Simulation Optimization". *Management Science* 42(7):954–973.

Branke, J., S. Meisel, and C. Schmidt. 2008. "Simulated Annealing in the Presence of Noise". *Journal of Heuristics* 14(6):627–654.

Chang, K.-H., L. J. Hong, and H. Wan. 2013. "Stochastic Trust-Region Response-Surface Method (STRONG)—A New Response-Surface Framework for Simulation Optimization". *INFORMS Journal on Computing* 25(2):230–243.

Deng, Y., J. Xu, C. Kesselman, and S. Sen. 2019. "Computational Operational Research Exchange (cORe): A Cyber-Infrastructure for Analytics". In *Proceedings of the 2019 Winter Simulation Conference*, edited by N. Mustafee, K.-H. G. Bae, S. Lazarova-Molnar, M. Rabe, C. Szabo, P. Haas, and Y.-J. Son. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Dong, N., D. J. Eckman, X. Zhao, M. Poloczek, and S. G. Henderson. 2017. "Empricially Comparing the Finite-Time Performance of Simulation-Optimization Algorithms". In *Proceedings of the 2017 Winter Simulation Conference*, edited by W. K. V. Chan, A. D'Ambrogio, G. Zacharewicz, N. Mustafee, G. Wainer, and E. Page, 2206–2217. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Fu, M. C. 2015. "Stochastic Gradient Estimation". In *Handbook of Simulation Optimization*, edited by M. C. Fu, Volume 216, 105–147. New York: Springer.

GitHub 2019. "GitHub". http://github.com, accessed 17th April.

GitHub Guides 2017. "Forking Projects". http://guides.github.com/activities/forking, accessed 17th April.

GitHub Help 2019a. "About Pull Requests". http://help.github.com/en/articles/about-pull-requests, accessed 17th April.

GitHub Help 2019b. "Fork a Repo". http://help.github.com/articles/fork-a-repo, accessed 17th April.

Gould, N. I., D. Orban, and P. L. Toint. 2015. "CUTEst: A Constrained and Unconstrained Testing Environment with Safe Threads for Mathematical Optimization". *Computational Optimization and Applications* 60(3):545–557.

Hong, L. J., B. L. Nelson, and J. Xu. 2015. "Discrete Optimization via Simulation". In *Handbook of Simulation Optimization*, edited by M. C. Fu, Volume 216, 9–44. New York: Springer.

Kiefer, J., and J. Wolfowitz. 1952. "Stochastic Estimation of the Maximum of a Regression Function". *Annals of Mathematical Statistics* 23(3):462–466.

Davis E. King 2019. "Dlib C++ Library". http://dlib.net, accessed 24th April.

MathWorks 2019. "Documentation: RandStream". https://www.mathworks.com/help/matlab/ref/randstream.html, accessed 18th April.

Arnold Neumaier 2019. "Global Optimization Software". https://www.mat.univie.ac.at/~neum/glopt/software_g.html, accessed 24th April.

Pasupathy, R., and S. G. Henderson. 2006. "A Testbed of Simulation-Optimization Problems". In *Proceedings of the 2006 Winter Simulation Conference*, edited by L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, 255–263. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Pasupathy, R., and S. G. Henderson. 2011. "SimOpt: A Library of Simulation Optimization Problems". In *Proceedings of the 2011 Winter Simulation Conference*, edited by S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and M. Fu, 4075–4085. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Polyak, B. T. 1990. "A New Method of Stochastic Approximation Type". *Automation and Remote Control* 7(2):98–107.

RAL Computational Mathematics 2019. "Constrained and Unconstrained Testing Environment with Safe Threads (CUTEst)". http://github.com/ralna/CUTEst, accessed 24th April.

Rosenbrock, H. H. 1960. "An Automatic Method for Finding the Greatest or Least Value of a Function". *The Computer Journal* 3(3):175–184.

Ruppert, D. 1988. "Efficient Estimators from a Slowly Convergent Robbins-Monro Procedure". Technical Report 781, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, New York.

SimOpt Library 2019. "Simulation Optimization (SimOpt) Library". http://www.simopt.org and http://github.com/simopt-admin/simopt/wiki, accessed 15th April.

Skopt 2019. "Scikit-Optimize Module". http://scikit-optimize.github.io, accessed 24th April.

Spall, J. C. 1992. "Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation". *IEEE Transactions on Automatic Control* 37(3):332–341.

Spall, J. C. 1998. "Implementation of Simultaneous Perturbation Algorithm for Stochastic Optimization". *IEEE Transactions on Aerospace and Electronic Systems* 34(3):817–823.

AUTHOR BIOGRAPHIES

DAVID J. ECKMAN is a postdoctoral research fellow in the Department of Industrial Engineering and Management Sciences at Northwestern University. He recently earned his Ph.D. in Operations Research from Cornell University. His research interests deal with simulation optimization with a focus on ranking and selection and exploratory multi-armed bandits. His email address is david.eckman@northwestern.edu.

SHANE G. HENDERSON is a professor in the School of Operations Research and Information Engineering at Cornell University. His research interests include discrete-event simulation and simulation optimization, and he has worked for some

time with emergency services and bike sharing applications. He co-edited the Proceedings of the 2007 Winter Simulation Conference. His email address is sgh9@cornell.edu and his web page is http://people.orie.cornell.edu/~shane.

RAGHU PASUPATHY is an associate professor in the Department of Statistics at Purdue University. His research interests lie broadly in Monte Carlo methods with a specific focus on simulation optimization. He is a member of INFORMS, IIE, and ASA, and serves as an associate editor for Operations Research and INFORMS Journal on Computing. His email address is pasupath@purdue.edu.