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.

EVALUATING SOLVERS FOR LINEARLY CONSTRAINED SIMULATION OPTIMIZATION

Natthawut Boonsiriphatthanajaroen[1], Rongyi He[1], Litong Liu[1,2], Tinghan Ye[1,2],
and Shane G. Henderson[1]

[1]School of Operations Research and Information Engr., Cornell University, Ithaca, NY, USA
[2]H. Milton Stewart School of Industrial and Systems Engr., Georgia Tech, Atlanta, GA, USA

ABSTRACT

Linearly constrained simulation optimization problems are those that include deterministic linear constraints in addition to an objective function that can only be evaluated through simulation. We provide several solvers for linearly constrained simulation optimization that all rely on gradient estimates of the objective function. We compare these solvers on random instances of 4 test problems from SimOpt.

1 INTRODUCTION

During the COVID-19 pandemic, a simulation model was used to help decide how to allocate a limited testing capacity across subgroups of the Ithaca campus and surrounding population of Cornell University (Frazier et al. 2022). In that setting, a fixed daily budget of testing capacity was to be shared between alternative groups. This is an instance of a linearly constrained simulation optimization problem (LCSOP), where a simulation model is used to estimate an objective function (here, risk-weighted expected infections across groups) that we wish to optimize over a linearly constrained feasible region.

We define LCSOPs to be of the form

$$\begin{aligned} \min_{x\in[l,u]} \ & f(x) = \mathbb{E} f(x,\xi) \\ \text{s.t. } & A_e x = b_e \quad \forall e \in E \\ & A_i x \le b_i \quad \forall i \in I. \end{aligned} \tag{1}$$

Here the deterministic bound vectors l, u could be finite or infinite, A_e and A_i are deterministic row vectors for each $e \in E$ and $i \in I$, b_e and b_i are deterministic constants, and E and I are finite sets. Moreover, the distribution of the random element ξ does not depend on the decision vector x.

This formulation encompasses many problems, e.g., see Section 3. Still, it is not completely general; objective functions that are built on quantiles do not fall into this class, nor do objective functions that are nonlinear functions of means of random variables.

We compare a selection of solvers that are designed to solve such problems using gradient estimates of the function $f(\cdot)$. We chose these solvers because they are readily implemented and are available in SimOpt (Eckman et al. 2023b). Other solvers, such as a variant of L-BFGS designed for stochastic problems (Bollapragada et al. 2018) might also be implemented and compared in future work. For the comparison we use computational results on a variety of test problems. Our goals are as follows.

1. To further develop a collection of solvers for linearly constrained simulation optimization; the solvers presented here were substantially improved during the course of our experimentation.
2. To showcase the availability, capability and limitations of these solvers.
3. To promote the use of disciplined modeling; some of our test problems are convex, which makes it easier to assert near global optimality of solutions and allows simplification of solver logic.

4. To showcase the benefits of SimOpt (Eckman et al. 2023a, 2023b) in comparing solvers on classes of simulation optimization problems, with the goal of inspiring similar work.

A related study (Dong et al. 2017) used an early version of SimOpt to compare a variety of solvers for box-constrained or unconstrained simulation optimization problems. Compared to that study, in this paper we specialize to solvers for LCSOPs that exploit gradient estimates, the solvers in question have undergone more development, and we showcase some recent SimOpt features such as the ability to automatically generate random instances of test problems and diagnostic plots.

The remainder of this paper is organized as follows. Section 2 presents several solvers for linearly constrained simulation optimization when gradient estimates for the objective are available. Section 3 reviews the test problems we use to compare the solvers, including how each problem is randomized to generate additional problems. Section 4 explains our experimental design, provides the computational results and comparison plots, and discusses the results.

2 SOLVERS

We explore 12 different solvers. These are obtained through all combinations of 3 gradient-based solvers (projected gradient descent, an active-set method, and away-step Frank-Wolfe), and 4 different line search methods (backtracking, adaptive step search, interpolation, and zoom). Adaptive step search is not a line search method because it uses a single step size in each iteration, but it replaces the line search, so we consider it as such. All of these solvers maintain feasibility (both active-set and Frank-Wolfe solvers use a ratio test to prevent infeasible step sizes) and seek optimality, so a comparison can be made purely on objective function values. None of these solvers use stopping rules because, in SimOpt, solvers are run until a problem-specific budget of simulation replications is exhausted, to facilitate comparisons. Solvers determine how much of the budget to expend on each iteration, which influences the accuracy of the function and gradient estimates. Implementations of the solvers can be found in the SimOpt Library (2024).

All descriptions use the notation x_k as the current iterate at step k, $\hat{f}(x_k)$ and $\hat{g}(x_k)$ as the estimated objective function and estimated gradient at x_k, and $\mathscr{X}$ as the feasible region of the optimization problem.

2.1 Projected Gradient Descent

Algorithm 1, Projected Gradient Descent (PGD), adapted from Iusem (2003), moves in the direction of the negative gradient, and uses an L_2 projection $\mathscr{P}_{\mathscr{X}}$ back to the feasible region to retain feasibility. The projection is obtained by solving a quadratic program using CVXPY (Diamond and Boyd 2016, Agrawal et al. 2018). Stepsizes are constrained to an iteration-specific maximum stepsize that is chosen adaptively.

2.2 Active-set Method

Algorithm 2 (AS) is a tailored version of an active-set method originally designed for linearly constrained convex quadratics (Nocedal and Wright 2006, §16.5). AS maintains a subset of constraints ($\mathscr{W}_k$) that are *active*, i.e., those that hold as equalities. In each iteration, we solve the quadratic direction-finding subproblem

$$\min_{d_k \in \mathbb{R}^n} \frac{1}{2}||d_k||_2^2 + \hat{g}(x_k)^T d \quad \text{s.t. } A_i d_k = 0 \quad \forall i \in \mathscr{W}_k, \tag{2}$$

which ensures that the constraints in the active set are enforced as equalities.

For a nonzero direction d_k, we determine how far we can go while remaining in the feasible space through a ratio test. If the obtained step size leaves us tight on a blocking constraint then we include the blocking constraint in the active set for the next iteration. When we obtain a zero direction from the subproblem, AS leverages the Lagrange multipliers associated with the subproblem's constraints to decide whether to drop a redundant constraint from the active set; otherwise a solution is identified that satisfies the Karush–Kuhn–Tucker (KKT) conditions.

Algorithm 1 Projected Gradient Descent (PGD)

```
Initialize feasible solution $x_0 \in \mathscr{X}$; maximum step size $\gamma_0^{max} \in [1,\infty)$; step size discount factor $r \in (0,1)$
for $k = 0,1,2,...$ do
    $d_k \leftarrow \frac{-\hat{g}(x_k)}{||\hat{g}(x_k)||_2}$
    $y_{k+1} \leftarrow x_k + \gamma_k^{max} d_k$
    if $y_{k+1}$ is NOT feasible then
        $y_{k+1} \leftarrow \mathscr{P}_{\mathscr{X}}(y_{k+1})$
        $d_k \leftarrow \frac{(y_{k+1}-x_k)}{\gamma_k^{max}}$
    end if
    Let $\gamma_k = \arg\min_{\gamma\in[0,\gamma_k^{max}]} \hat{f}(x_k+\gamma d_k)$    (line search)
    if $\gamma_k = \gamma_k^{max}$ then
        $\gamma_k^{max} \leftarrow \frac{\gamma_k^{max}}{r}$
    else
        $\gamma_k^{max} \leftarrow r\gamma_k^{max}$
    end if
    $x_{k+1} \leftarrow x_k + \gamma_k d_k$
end for
```

2.3 Frank-Wolfe

Algorithm 3 is developed from the Frank-Wolfe algorithm. The original algorithm (Frank and Wolfe 1956) determines the direction for each step by solving a linear program to find the vertex of the feasible region that improves a linear approximation of the objective function the best. The current solution is represented in terms of a convex combination (vertices' representation) of a subset of the vertices $\mathscr{V}$ of the feasible region, i.e., $x_k = \sum_{u\in\mathscr{V}} \alpha_u^{(k)} u$. Define the *active vertex set* at iteration k, $\mathscr{S}^{(k)}$, to be all vertices such that the coefficients are positive, i.e. $\mathscr{S}^{(k)} = \{v \in \mathscr{V} | \alpha_v^{(k)} > 0\}$. In each iteration, we limit the step size to a value γ_k^{max} that ensures feasibility. It is possible that the algorithm may alternate between several vertices, resulting in zigzagging behavior, which is inefficient. This motivates "away steps" (Julien and Jaggi 2015), where we replace the search direction with a step in a direction that moves away from a "bad" vertex.

2.4 Line Search Methods

Our four line search methods aim for a *sufficient decrease* in the function values (assuming minimization problems). Another condition that is sometimes used is the *curvature condition* where we check if the directional derivative in the search direction at the current step is sufficiently small.

2.4.1 Backtracking

Backtracking in Algorithm 4 starts with the maximum stepsize, and progressively scales it by $r \in (0,1)$ until we obtain sufficient decrease with parameter $\theta \in (0,1)$ as in Nocedal and Wright (2006). During the line search, if we run out of simulation budget or hit a budget on the number of backtracking steps, we will return the final stepsize, i.e., $r^t\gamma_k^{max}$, where t is the number of backtracks.

2.4.2 Adaptive Step Search

Adaptive step search in Algorithm 5 uses a single step size in each iteration, increasing the step size when sufficient descent occurs, and decreasing it otherwise (Berahas et al. 2021; Jin et al. 2021).

Algorithm 2 Active-set Method (AS)

```
Initialize feasible solution $x_0 \in \mathscr{X}$; maximum step size $\gamma^{max} \in [1,\infty)$; initial active set $\mathscr{W}_0 = E$
for $k = 0,1,2,\ldots$ do
    Generate direction $d_k$ and Lagrange multipliers $\pi^k$ from Equation (2)
    if $d_k = 0$ (no feasible direction) then
        if $\pi_i^k \geq 0$ for all $i \in \mathscr{W}_k \cap I$ then
            Break (KKT conditions satisfied)
        else
            Let $q = \arg\min\{\pi_i^k : i \in \mathscr{W}_k \cap I\}$
            $\mathscr{W}_{k+1} \leftarrow \mathscr{W}_k \setminus \{q\}$ and $x_{k+1} \leftarrow x_k$ (drop one of the constraints from the active set)
        end if
    else
        if $A_i d_k \leq 0$ for all $i \notin \mathscr{W}_k$ (there are no blocking constraints) then
            Let $r_k = \arg\min_{\gamma \in [0,\gamma^{max}]} \hat{f}(x_k + \gamma d_k)$ (line search)
            $\mathscr{W}_{k+1} \leftarrow \mathscr{W}_k$ and $x_{k+1} \leftarrow x_k + r_k d_k$
        else
            $\alpha_k \leftarrow \min_{i \notin \mathscr{W}_k : A_i d_k > 0} \{\frac{b_i - A_i x_k}{A_i d_k}\} := \{\frac{b_q - A_q x_k}{A_q d_k}\}$ (ratio test)
            if $\alpha_k > \gamma^{max}$ then
                Let $r_k = \arg\min_{\gamma \in [0,\gamma^{max}]} \hat{f}(x_k + \gamma d_k)$ (line search)
                $\mathscr{W}_{k+1} \leftarrow \mathscr{W}_k$ and $x_{k+1} \leftarrow x_k + r_k d_k$
            else
                Let $r_k = \arg\min_{\alpha \in [0,\alpha_k]} \hat{f}(x_k + \alpha d_k)$ (line search)
                if $r_k = \alpha_k$ (the optimal step is tight on the blocking constraint) then
                    $\mathscr{W}_{k+1} \leftarrow \mathscr{W}_k \cup \{q\}$ (add one of the blocking constraints to the active set)
                end if
                $x_{k+1} \leftarrow x_k + r_k d_k$
            end if
        end if
    end if
end for
```

2.4.3 Interpolation

In Algorithm 6, we use a quadratic function $q(s)$ to interpolate $\hat{f}(x_k + sd_k)$ in iteration k. If the optimal step size γ_k for $q(s)$ is too small, we reduce the maximum step size and interpolate again (Nocedal and Wright 2006).

2.4.4 Zoom

The Zoom line search method (Algorithm 7) returns a step size that satisfies both the *sufficient decrease* and the *curvature condition* (Nocedal and Wright 2006). The method starts from a small step size and increases until it hits the maximum step size.
The **Zoom**(γ_ℓ, γ_h) function in the algorithm accepts two past step sizes in to form an interval that contains the optimal step size. The inputs for **Zoom** should satisfy the following properties.

1. The interval (γ_ℓ, γ_h) contains a step size that satisfy both sufficient decrease and curvature condition.
2. γ_ℓ is the step size that gives the smallest (if minimization) function evaluation so far.
3. γ_h is picked so that $\phi'(\gamma_\ell)(\gamma_h - \gamma_\ell) < 0$, where $\phi(s) = \hat{f}(x_k + sd_k)$ and ϕ' denote the derivative over the step size s.

Algorithm 3 Away-Step Frank-Wolfe (FW)

```
Initialize $x_0$ to be a vertex $v$ and set $\alpha_v^{(0)} = 1$ and all other coefficients to 0
for $k = 0, 1, 2, \ldots$ do
    $s_k \leftarrow \arg\min_{s \in \mathscr{X}} \hat{g}(x_k)^T s$
    $v_k \leftarrow \arg\max_{u \in \mathscr{X}} \hat{g}(x_k)^T u$ (the "bad" vertex)
    $d_k^{FW} \leftarrow s_k - x_k$   (FW direction)
    $d_k^{A} \leftarrow x_k - v_k$   (away direction)
    if $-\hat{g}(x_k)^T d_k^{FW} \geq -\hat{g}(x_k)^T d_k^{A}$ then
        $d_k \leftarrow d_k^{FW}$ and $\gamma_k^{max} \leftarrow 1$   (normal FW case)
    else
        $d_k \leftarrow d_k^{A}$ and $\gamma_k^{max} \leftarrow \alpha_{v_k}^{(k)}/(1 - \alpha_{v_k}^{(k)})$  (away-step FW)
    end if
    $\gamma_k \leftarrow \arg\min_{\gamma \in [0, \gamma_k^{max}]} \hat{f}(x_k + \gamma d_k)$   (line search)
    $x_{k+1} \leftarrow x_k + \gamma_k d_k$
    Update the weights $\alpha^{(k)}$ giving the vertex representation of $x_{k+1}$ to $\alpha^{(k+1)}$
end for
```

Algorithm 4 Backtracking Line Search (iteration k)

```
Given max step size $\gamma_k^{max}$, a starting point $x_k$, and a direction $d_k$
Given parameters $\theta$, $r \in (0,1)$
Initialize $\gamma_k \leftarrow \gamma_k^{max}$
while $\hat{f}(x_k + \gamma_k d_k) \geq \hat{f}(x_k) + \theta \gamma_k \hat{g}(x_k)^T d_k$ do
    $\gamma_k \leftarrow r \gamma_k$
end while
```

Algorithm 5 Adaptive Step Search (iteration k)

```
Given max step size $\gamma^{max}$, a step size $\gamma_k$, a starting point $x_k$, and a direction $d_k$
Given parameters $\theta$, $r \in (0,1)$, $\kappa > 0$
if $\hat{f}(x_k + \gamma_k d_k) \leq \hat{f}(x_k) - \theta \gamma_k ||d_k||^2 + 2\kappa$   (modified Armijo condition) then
    $\gamma_{k+1} \leftarrow \min\{\frac{\gamma_k}{r}, \gamma^{max}\}$
else
    $\gamma_{k+1} \leftarrow r \gamma_k$
end if
```

Algorithm 6 Interpolation Line Search (iteration k)

```
Given max step size $\gamma_k^{max}$, a starting point $x_k$, and a direction $d_k$
Given parameters $\theta$, $r \in (0,1)$
Initialize $\gamma_k \leftarrow \gamma_k^{max}$
while $\hat{f}(x_k + \gamma_k d_k) > \hat{f}(x_k) + \theta \gamma_k \hat{g}(x_k)^T d_k$ do
    Use $\hat{f}(x_k), \hat{f}(x_k + \gamma_k^{max} d_k)$, and the directional derivative $\hat{g}(x_k)^T d_k$ to get the quadratic $q$
    $\gamma_k \leftarrow \arg\min_{\gamma \in [0, \gamma_k^{max}]} q(\gamma)$
    $\gamma_k^{max} \leftarrow r \gamma_k^{max}$
end while
```

3 PROBLEMS

Here we sketch the test problems. Detailed descriptions can be found in the SimOpt Library (2024).

Algorithm 7 Zoom Line Search (iteration k)

```
Given max step size $\gamma_k^{max}$, a starting point $x$, and a direction $d_k$
Given parameters $\theta, r, \rho \in (0,1)$, $\delta > 1$.
Initialize $\gamma_{k,0} \leftarrow 0$ and $\gamma_{k,1} \in (0, \gamma_k^{max})$.
Let $\phi(\gamma_k) = \hat{f}(x_k + \gamma_k d_k)$
while True do
    if $\phi(\gamma_{k,i}) \geq \phi(0) + \theta \gamma_{k,i} \phi'(0)$ then
        return Zoom($\gamma_{k,i-1}, \gamma_{k,i}$)
    end if
    if $|\phi'(\gamma_{k,i})| \leq -\rho \phi'(0)$ then
        return $\gamma_{k,i}$
    end if
    if $\phi'(\gamma_{k,i}) \geq 0$ then
        return Zoom($\gamma_{k,i}, \gamma_{k,i-1}$)
    end if
    $\gamma_{k,i+1} \leftarrow \delta \gamma_{k,i}$
    if $\gamma_{k,i+1} \geq \gamma_k^{max}$ then
        return $\gamma_k^{max}$
    end if
end while
```

Algorithm 8 Zoom(γ_ℓ, γ_h)

```
while True do
    Interpolate with a quadratic function as in Algorithm 6 and get $\gamma_{k,i+1}$ as an optimizer between $\gamma_\ell, \gamma_h$
    if $\phi(\gamma_{k,i+1}) \geq \phi(0) + \theta \gamma_i \phi'(0)$ then
        $\gamma_h \leftarrow \gamma_{k,i+1}$
    else
        if $|\phi'(\gamma_{k,i+1})| \leq -\rho \phi'(0)$ then
            return $\gamma_{k,i+1}$
        end if
        if $\phi'(\gamma_{k,i+1})(\gamma_h - \gamma_\ell) \geq 0$ then
            $\gamma_h \leftarrow \gamma_\ell$
        end if
        $\gamma_\ell \leftarrow \gamma_{k,i+1}$
    end if
    $i \leftarrow i + 1$
end while
```

3.1 Constrained Stochastic Activity Network (SAN)

Consider a stochastic activity network (SAN) with m nodes and n arcs. Each arc A_i is associated with a task with random duration X_i that is exponentially distributed with mean θ_i. Task durations are independent.

Let $T(\theta)$ be the duration of the longest path from a distinguished source node to a distinguished sink node. Suppose we can choose the mean task duration $\theta_i > 0$ for each arc i. The objective is to minimize $\mathbb{E}[T(\theta)]$, subject to $\sum_i \theta_i \geq L$, where L is a lower bound for the sum of the arc means. This is a convex optimization problem. To get gradient estimates we use infinitesimal perturbation analysis (IPA).

To generate random instances, we randomly sample a fixed number of arcs included in the network, $\{A_i\}_{i=1}^n$, the lower bound L, and the simulation-replication budget B.

3.2 Stochastic Max Flow (SMF)

Consider a maximum flow problem with a source node s and a sink node t, where each arc i has an assigned initial capacity $x_i > 0$. Each arc i also has a normally distributed random noise n_i (independent across arcs), and the actual capacity for each arc equals $\max\{x_i - n_i, 0\}$. The (non-convex) problem is to select $\mathbf{x}$ to maximize the expected total flow out of the source node s, while satisfying $\sum_i x_i \leq C$, an upper bound for the total initial arc capacities. To get gradient estimates we use IPA, exploiting duality.

To generate a random instance we sample the arcs $\{A_i\}_{i=1}^n$ in the network. We also sample the budget B to align with the problem dimension. The noise terms remain as independent normal random variables.

3.3 Convex Stochastic Max Flow (SMFCVX)

Consider the same stochastic maximum flow problem, except that the realized capacity of arc i is now $n_i x_i$, where the (independent across arcs) noise factors n_i are now Erlang distributed. This is a convex optimization problem. Gradient estimates and random instances are obtained as before.

3.4 Cascade Network (Cascade)

Let $G = (V, E)$ be a directed acyclic graph. Each edge $e \in E$ has a weight that corresponds to the activation probabilities $p_e, \forall e \in E$; each node v has an activation cost c_v and an initial node activation probability $u_v \in [0, 1], \forall v \in V$. If edge $e = (i, j)$ is activated then if node i is activated, so is node j.

Progressive cascades of the network are simulated through the independent cascade model (Kempe et al. 2003). Specifically, we first generate the activated edges, each with probability p_e, to form a subgraph of G. After that, we identify the nodes that are activated at the start of the cascade, each with probability u_v. Then, we perform the cascade, progressively activating inactive nodes that are connected by an activated edge to an activated node. Lastly, we count the total number of nodes in the connected components of the initially activated nodes.

The objective is to choose the node activation probabilities u_v that maximize the expected number of post-cascade activated nodes, subject to $\sum_v c_v u_v \leq C$, where C is a budget on the total node activation cost. We do not believe this problem is convex. To get gradient estimates, we use finite differences.

To generate a random instance, we generate a random acyclic directed graph G, randomly sample the activation costs c_v, the cost budget C, and the simulation-replication budget B.

Two excluded problems: In our initial experiments, we included two additional problems — an open Jackson network problem and a network queuing system design problem. We later discarded these problems because the objective curves were too flat in our particular random instances to generate interesting problems for solver comparisons. Furthermore, it was challenging to obtain accurate gradient estimates in the network queuing problem due to the large variance of the likelihood ratio gradient estimator used therein.

4 EXPERIMENTS

We first present our experimental design, then give selected results and discuss what we have learned.

4.1 Experimental Design

We generate 5 random instances for each of the 4 simulation problems mentioned in Section 3, resulting in 20 problems in total. Generating random instances of the same problem allows us to investigate how solver performance varies when handling problems with different parameters and hence different complexities. Each problem is solved by the 12 solvers introduced in Section 2. All problems are run using common random numbers across all solvers, to ensure a fair comparison.

Based on SimOpt's random number generator, we ensure different random problem instances to be generated using different substreams, and each distinct problem factor and model factor has its own individual subsubstream. With this allocation of streams there is dependence in the generated problems across different problem classes, but that dependence should not have any bearing on our results.

To evaluate the solvers we use 30 macroreplications of a fixed replication budget and 50 postreplications for each problem-solver pair. We selected these numbers from initial experiments which indicated that the values were small enough to permit tolerable computation times while also giving informative results.

All problems and solvers come from the SimOpt architecture. For each problem, we evaluate the performance of different solvers by examining their mean objective progress curves and terminal objective violin plots. In addition, we generate summary plots that aggregate the solver performances on all problems. This includes α-solvability profile curves and area-under-progress-curve scatter plots (Eckman et al. 2023b).

All experiments were conducted on a Linux server equipped with dual Intel Xeon Gold 6226 CPUs at 2.7 GHz and 384 GB of RAM. The implementation of all problems and solvers can be found in the SimOpt Library (2024), along with information on how to replicate our results.

4.2 Results

In these results we name solvers "x-y," where "x" is one of PGD (projected gradient descent), AS (active set) and FW (Frank-Wolfe), and "y" is one of B (backtracking), SS (adaptive step search), I (interpolation) and Z (zoom). Problems are named SAN (stochastic activity network), SMF (nonconvex stochastic max flow), SMFCVX (convex stochastic max flow) and Cascade (cascade network). Figure 1 provides an overview of how all the solvers performed across all problems, with 3 of the plots zooming in on the top-performing solvers. Figures 2 (SAN and SMF) and 3 (SMFCVX, Cascade) provide a more detailed view using terminal objective function plots and unnormalized progress curves for each of the solvers on selected random instances of the problems.

The CDF solvability plot for all 12 solvers in Figure 1 indicates that no solver manages to solve more than about 75% of the problems. Moreover, there is quite a bit of variability in how rapidly solvers can get to approximately optimal solutions. Still, the plot is too cluttered to draw fine distinctions. The CDF solvability plots for the top performing solvers are easier to interpret. There we see the strength of FW-I and FW-SS for small budgets, and while the confidence intervals at the larger budget levels overlap, it does appear that these solvers remain competitive at larger budgets. AS-B also performs strongly, though not uniformly so, as the scatter plot indicates. So do the two PGD solvers that we selected, though not as well for smaller budgets. The PGD solvers do better relatively in the plot of 0.2 solve times, suggesting good performance for obtaining rough solutions.

The terminal progress plot and progress curve for the selected random problem instance of SAN in Figure 2 suggest that the FW solvers struggle compared with the other solvers. FW-Z performs the best out of the FW solvers, though with much slower progress compared with the non-FW solvers. The best performers are AS-I and AS-SS. Similar observations apply for the results for other random SAN problems (not shown).

The plots for the selected random problem instance of (non-convex) SMF in Figure 2 indicate that while the non-Zoom FW solvers find the best solutions, their performance is highly variable compared with the non-FW solvers. The other solvers mostly do well, though it appears that they identify a local minimum on this non-convex problem. The PGD solvers are the most consistent solvers in terms of the quality of the terminal objective, with little variability in their performance.

The plots for the selected random problem instance of SMFCVX in Figure 3 are puzzling, since we'd expect most solvers to attain near-optimal solutions, yet there is a wide range of terminal objective function values. The AS solvers mostly do well, though they have a nontrivial number of poor runs, while PGD struggles rather consistently.

The plots for the selected random problem instance of Cascade indicate the strength of the non-Zoom FW solvers, which dominate, though the non-Zoom AS solvers do very well. The stepwise progress of the Zoom solvers is apparent relative to the non-Zoom solvers, which make more steady progress. FW-Z does

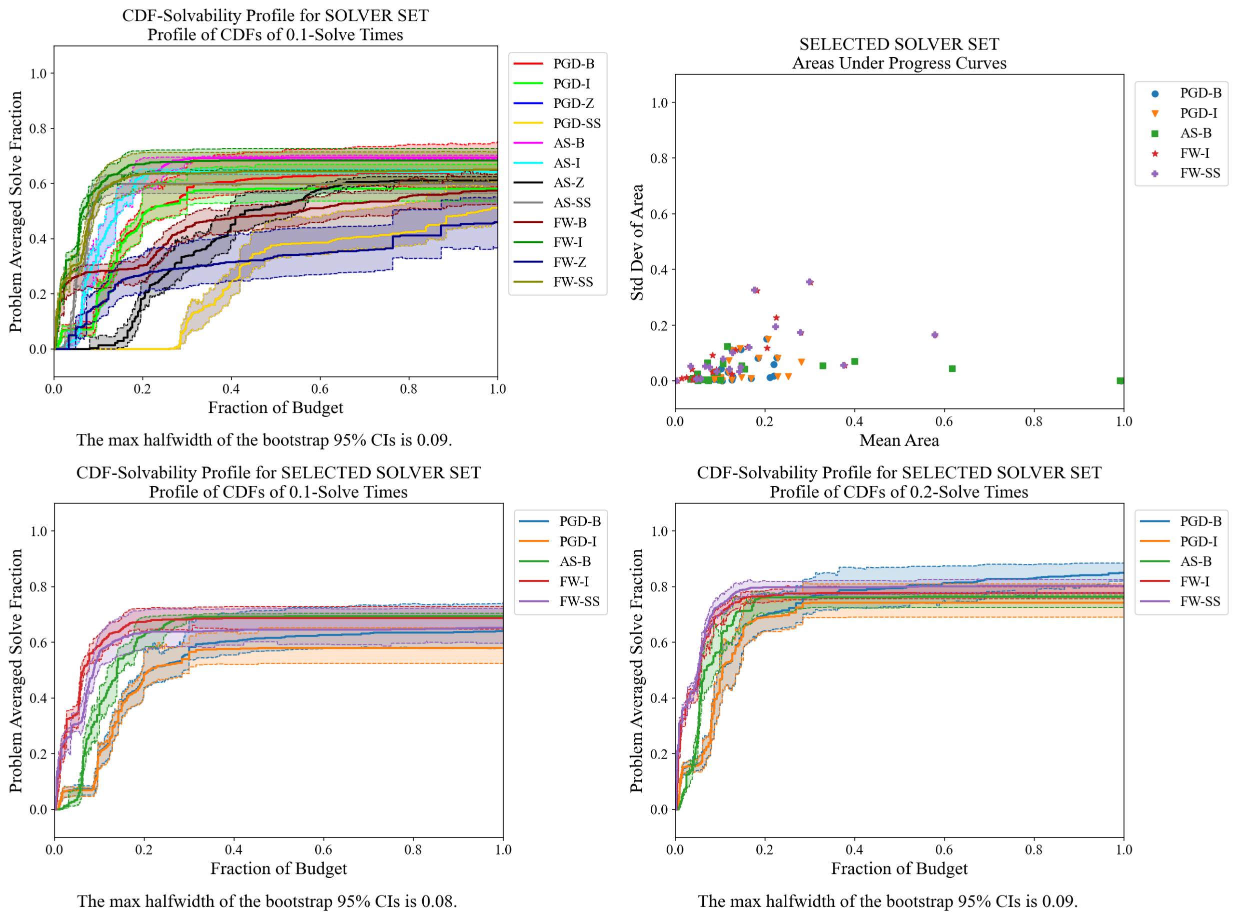

Figure 1: A CDF solvability profile at level 0.1 for all solvers (top left), an area scatter plot for solvers viewed to be amongst the top performers (top right) and CDF solvability profiles for those selected solvers at level 0.1 (bottom left) and 0.2 (bottom right). Solvability plots show the empirical cdf of the budgets required to get within 10% (respectively, 20%) of the best solution found by any solver on any run. The maximum height of the curves gives the fraction of problems solved to the stated accuracy. The shaded regions give 95% bootstrapped confidence intervals. The area scatter plot gives the mean and standard deviation of the area under normalized progress curves - better results lie in the bottom left of the plot.

very well for terminal progress on 2 instances, but otherwise isn't as effective as the other FW methods. On one of the Cascade instances that is not shown, the PGD solvers do the best out of all solvers.

4.3 Discussion

All the comments herein apply to *our solver implementations* only. We do not mean to imply that *all* possible implementations of these solvers would share the traits we have observed.

No class of solvers or class of line searches dominates across all problems. Nevertheless, several solvers gave strong all-round performance, including PGD-B, PGD-I, AS-B, FW-I and FW-SS.

On (non-convex) SMF, several solvers reach local optima and then fail to make further progress, suggesting that some method for detecting local convergence and restarting might be an important enhancement. (Regarding convergence to local optima, the progress plots in SimOpt would not provide much additional information while a solver was exploring solutions that are inferior to the best local minimum found so

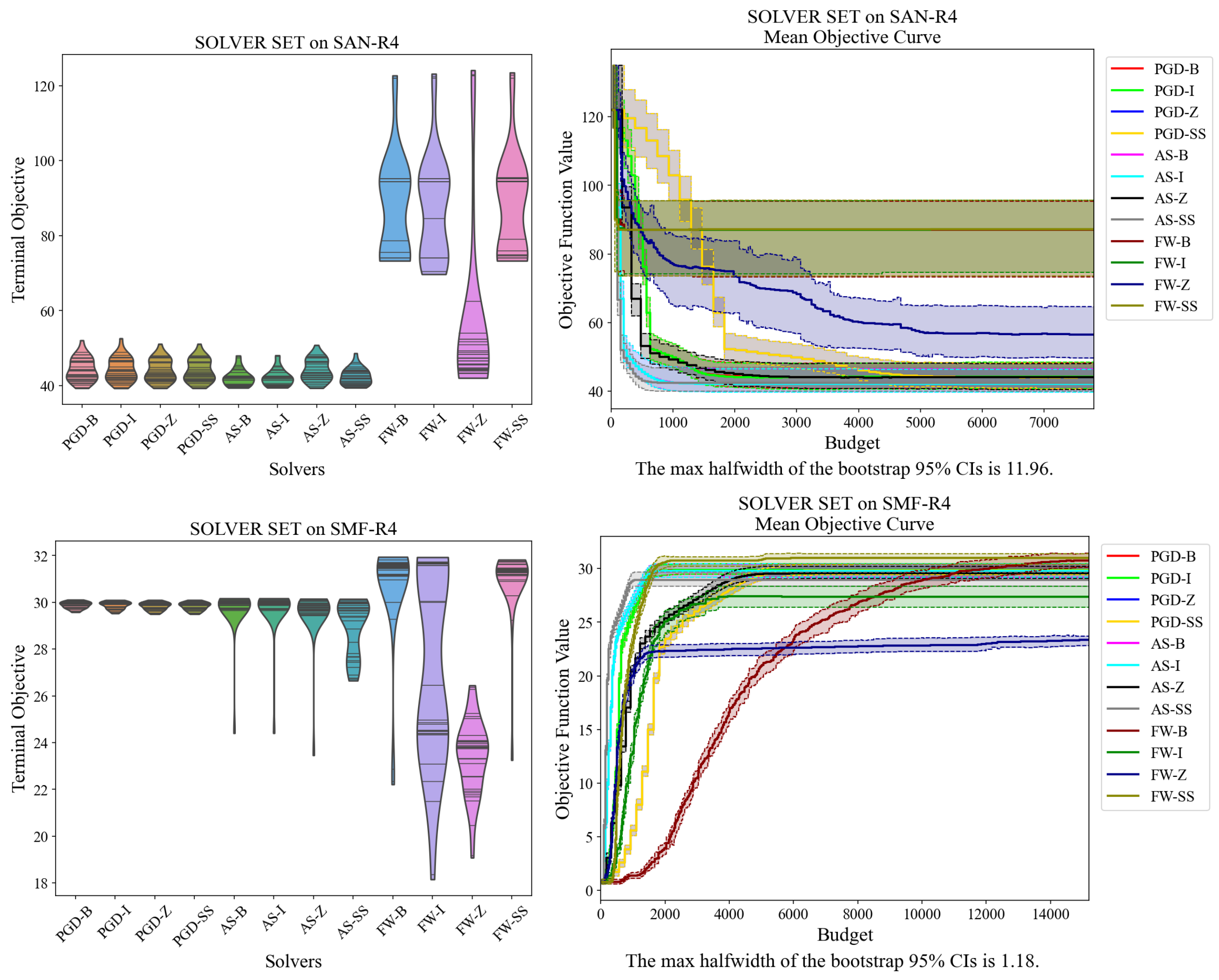

Figure 2: Terminal violin plots (left) and mean progress curves (right) for selected instances of SAN and SMF.

far.) On SMFCVX, which is convex, all solvers appear to struggle, suggesting the need for further solver development.

Zoom needs many simulation replications to make decisions, but often achieves substantial progress. Accordingly, the solvers that use a Zoom line search have progress curves that are more “stilted" than those solvers not using Zoom. The use of Zoom seems well-suited to problems requiring accurate solutions.

FW-style solvers work very well when the constraint set is reasonably “tight,” but these solvers did poorly when large bounds were added to a problem (SAN) that was more naturally modeled as having an unbounded feasible region. We believe this happens because FW solvers tend to select step sizes that are appreciable fractions of the size of the polytope. For search directions that head away from the origin in the SAN problems, this can lead to poor steps. On the other hand, it is also possible that FW solvers can be dramatically faster than other solvers if the solvers start at a vertex that is far from optimal, because FW solvers can move directly to a vertex that is close to optimal, while other solvers have to iteratively update their step size. Perhaps a FW-style solver could be developed that explicitly accounts for unbounded feasible regions. The paper Wang et al. (2022) is an interesting development in that direction, but only applies to a very specific class of unbounded feasible regions.

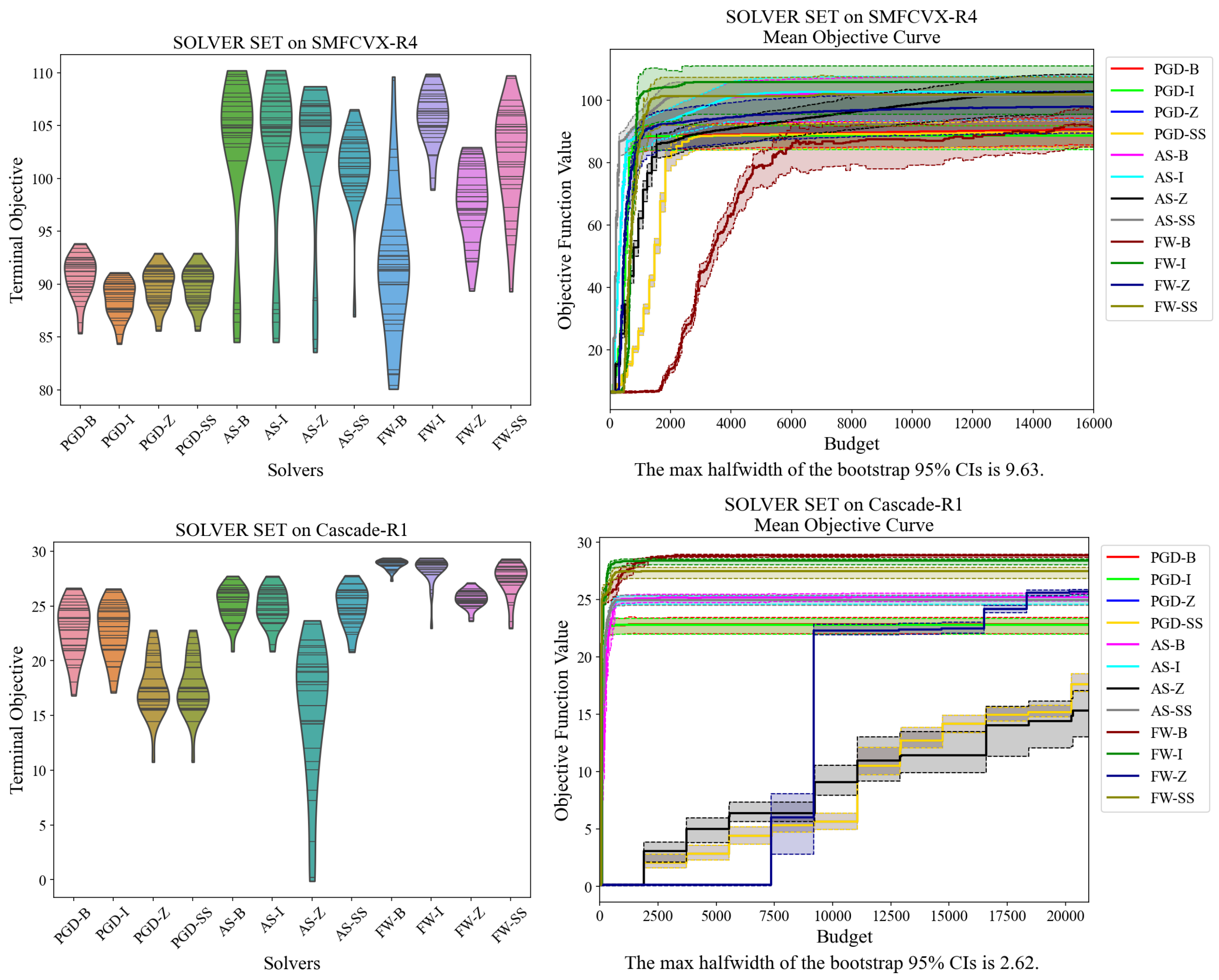

Figure 3: Terminal violin plots (left) and mean progress curves (right) for selected instances of SMFCVX and Cascade.

The interpolation line search method can struggle when function evaluations at two points are of different magnitudes, since then the interpolated function can be inaccurate. We observed this behavior with the SAN problems using the FW-I solver, for example.

ACKNOWLEDGMENTS

We thank David Eckman and Sara Shashaani for helpful discussions, and the many contributors who have developed test problems and solvers in SimOpt; they are too numerous to mention individually, but we are grateful! This research was partially supported by National Science Foundation grant CMMI-2035086.

REFERENCES

Agrawal, A., R. Verschueren, S. Diamond, and S. Boyd. 2018. "A rewriting system for convex optimization problems". *Journal of Control and Decision* 5(1):42–60.

Berahas, A. S., L. Cao, and K. Scheinberg. 2021. "Global convergence rate analysis of a generic line search algorithm with noise". *SIAM Journal on Optimization* 31(2):1489–1518.

Bollapragada, R., J. Nocedal, D. Mudigere, H.-J. Shi and P. T. P. Tang. 2018. "A progressive batching L-BFGS method for machine learning". In *International Conference on Machine Learning*, 620–629. PMLR.

Diamond, S. and S. Boyd. 2016. “CVXPY: A Python-embedded modeling language for convex optimization”. *Journal of Machine Learning Research* 17(83):1–5.

Dong, N. A., D. J. Eckman, M. Poloczek, X. Zhao and S. G. Henderson. 2017. “Comparing the finite-time performance of simulation-optimization algorithms”. In *Proceedings of the 2017 Winter Simulation Conference*, 2206–2217. Piscataway NJ: IEEE.

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

Eckman, D. J., S. G. Henderson, and S. Shashaani. 2023b. “SimOpt: A testbed for simulation-optimization experiments”. *INFORMS Journal on Computing* 35(2):495–508.

Frank, M. and P. Wolfe. 1956. “An algorithm for quadratic programming”. *Naval Research Logistics Quarterly* 3(1-2):95–110.

Frazier, P. I., J. M. Cashore, N. Duan, S. G. Henderson, A. Janmohamed, B. Liu, , *et al*. 2022. “Modeling for COVID-19 college reopening decisions: Cornell, a case study”. *Proceedings of the National Academy of Sciences* 119(2) https://doi.org/10.1073/pnas.2112532119.

Iusem, A. N. 2003. “On the convergence properties of the projected gradient method for convex optimization”. *Computational & Applied Mathematics 22*:37–52.

Jin, B., K. Scheinberg, and M. Xie. 2021. “High Probability Complexity Bounds for Adaptive Step Search Based on Stochastic Oracles”. *arXiv preprint arXiv:2106.06454*.

Julien, S. L. and M. Jaggi. 2015. “On the Global Linear Convergence of Frank-Wolfe Optimization Variants”. *Conference on Neural Information Processing Systems*.

Kempe, D., J. Kleinberg, and É. Tardos. 2003. “Maximizing the spread of influence through a social network”. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 137–146.

Nocedal, J. and S. J. Wright. 2006. *Numerical Optimization*. 2nd ed. Springer.

SimOpt Library 2024. “Simulation Optimization (SimOpt) Library”. https://github.com/simopt-admin/simopt/tree/WSC24.

Wang, H., H. Lu, and R. Mazumder. 2022. “Frank–Wolfe Methods with an Unbounded Feasible Region and Applications to Structured Learning”. *SIAM Journal on Optimization* 32(4):2938–2968 https://doi.org/10.1137/20M1387869.

AUTHOR BIOGRAPHIES

NATTHAWUT BOONSIRIPHATTHANAJAROEN is a Ph.D. student in Operations Research and Information Engineering at Cornell University. His research interests are stochastic optimization algorithms and applications of simulation. His email address is nb463@cornell.edu and his homepage is https://natthab.github.io/.

RONGYI HE is a Master of Engineering student in Operations Research and Information Engineering at Cornell University. Her email address is rh463@cornell.edu.

LITONG LIU is a Ph.D. student in the H. Milton Stewart School of Industrial and Systems Engineering at Georgia Tech. The majority of Litong’s work on SimOpt was conducted while she was a Master of Engineering student in Operations Research and Information Engineering at Cornell University. Her email address is ll936@cornell.edu and her homepage is https://litongliu.github.io/.

TINGHAN (JOE) YE is a Ph.D. student in the H. Milton Stewart School of Industrial and Systems Engineering at Georgia Tech. The majority of Joe’s work on SimOpt was conducted while he was an undergraduate student at Cornell University. His email address is joe.ye@gatech.edu and his homepage is https://tinghan-joe-ye.netlify.app.

SHANE G. HENDERSON holds the Charles W. Lake, Jr. Chair in Productivity in the School of Operations Research and Information Engineering at Cornell University. His research interests include simulation theory and a range of applications including emergency services. He is an INFORMS Fellow. He is a co-creator of SimOpt, a testbed of simulation optimization problems and solvers. His email address is sgh9@cornell.edu and his homepage is http://people.orie.cornell.edu/shane.