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.

SIMULATION OPTIMIZATION: AN INTRODUCTORY TUTORIAL ON METHODOLOGY

Sara Shashaani[1]

[1]Fitts Dept. of Industrial and Systems Engineering, North Carolina State University, Raleigh, NC, USA

ABSTRACT

With an upward trend for use in real-world problems of high uncertainty, the field of simulation optimization (SO) is evolving to aid in finding near-optimal solutions more rapidly and reliably. A comprehensive overview of the vast and diverse literature in continuous and discrete SO over large spaces is difficult. This short tutorial intends instead to introduce the methodological landscape, stirring a middle ground between statistical analysis of Monte Carlo sampling and mathematical analysis of numerical optimization. Particular attention to sampling and its impact on SO Algorithms highlights open and promising research directions.

1 INTRODUCTION

Simulation Optimization (SO) broadly refers to all problems where either the objective function or constraints or both need to be evaluated with a Monte Carlo simulation. A typical formulation for SO is

$$\min_{x\in\mathcal{X}} \{f(x) := \mathbb{E}_\xi[F(x,\xi)]\}\,,\ \text{s.t.}\ \{c_i(x) := \mathbb{E}_\xi[C_i(x,\xi)]\} \leq 0\ \text{ for } i = 1,2,\cdots,n_c. \tag{P}$$

In problem (P), $f : \mathcal{X} \to \mathbb{R}$ and $c_i : \mathcal{X} \to \mathbb{R}$ are real-valued functions and ξ is the random object defined on a probability space $(\Xi, \mathcal{F}, P)$. $\mathcal{F}$ is a σ-algebra, that can loosely be viewed as the collection of all events, and P is a function assigning a probability to every event. In this paper, we consider bound constraints on x to be observed in the d-dimensional feasible space $\mathcal{X}$. The real-valued random functions $F : \mathcal{X} \times \Xi \to \mathbb{R}$ and $C_i : \mathcal{X} \times \Xi \to \mathbb{R}$ (capital letters denoting randomness) can be computed from the observed outputs of a stochastic simulation. The noise space Ξ contains all possible values that the random object can take with a nonzero probability. We denote the optimal solution and function value by x^* and f^*, respectively.

Depending on the problem, the nature of ξ may range from a random variate to a complicated random process. For example, when F is a simulated utilization of a queue, ξ is the stochastic process of arrivals and service times. In simulating a newsvendor's profit at the end of a fixed time horizon, ξ may be a random vector of demands or an autoregressive time series. When F is the binary value indicating bankruptcy of an insurance company, ξ is a composite Poisson process that generates the number of claims and the amount of each claim. Or in simulated inventory costs, ξ may be a Markov process generated via transition probabilities of demand or lead time for each product at each time unit. Fixing ξ, the function $F(\cdot, \xi)$ is also called the *sample path* (or *random field*).

Formulation (P) contains most variants of an SO problem. The objective function f (constraint c_i) can also be a quantile (value-at-risk), probability, conditional expectation, or other distributional property of the stochastic objective function F (stochastic constraint C_i). We review methods that solve (P) with continuous or discrete $\mathcal{X}$ that is too large for an exhaustive search (ranking and selection—R&S) (Kim and Nelson 2006). Our chief focus is on numerical and iterative solution methods, a.k.a, *solvers*, that generate, at each iteration k, an incumbent solution X_k besides other algorithm-generated ingredients that evolve throughout the search, such as the step size Δ_k or the sample size N_k. Besides simulation-based problems, SO can also be applied to machine learning or calibration (Peng et al. 2023; Shashaani and Vahdat 2023).

SO problems are difficult as they require circumventing mathematical programming challenges *and* controlling the noise in evaluated quantities. Both asymptotic and finite-time effectiveness of SO solvers involves synergies between *estimation* and the typical exploration and exploitation trade-offs of numerical methods (Andradóttir and Prudius 2009). Managing these synergies matter because the commonly used sample averaging to replace a stochastic problem with a deterministic counterpart may derail the search towards suboptimal solutions; we will discuss this further in Section 2. In SO research, more novel sampling strategies not only can eliminate the possibility of divergence, they can also speed up the algorithm. To understand this better, we review the main metrics that can evaluate an SO algorithm.

1.1 Performance Metrics of Simulation Optimization Solvers and Clarification on Terminology

Convergence: A solver's desirable asymptotic property is convergence; namely, a probabilistic guarantee that the solver will eventually find an optimal solution. A place of stringency is the type of probabilistic guarantees, i.e., showing that the generated sequence of random iterates (a.k.a., solutions) $\{X_k\}_{k=1,2,\cdots}$ converges with probability 1 (wp1), with high probability (whp), in expectation (L_1), or in mean-squared error (L_2). The wp1 convergence is considered *strong*, implying that any possible trajectory produced by the solver will converge. For a nonconvex problem, often *first-order* convergence is desirable, where $\mathbb{P}\{\|\nabla f(X_k)\| \xrightarrow[k\to\infty]{} 0\} = 1$. *Global* convergence in the same context implies convergence from any arbitrary x_0—not to be mistaken with global optimization algorithms that prove convergence to x^* (typically of interest in convex or discrete SO). For example, ASTRO (Adaptive Sampling Trust Region Optimization) (Ha et al. 2024) is a noncovex SO algorithm that globally converges to a stationary point wp1. Conversely, *local* convergence requires that x_0 is close enough to a minimizer and is used to derive convergence rates.

Complexity: While convergence guarantees arm the user with confidence, they lack knowledge about solver's efficiency and non-asymptotic behavior. In practice, a solution algorithm will terminate when a computational resource is exhausted or a pre-specified criterion indicates a *near-optimal* solution has been reached. A non-asymptotic guarantee for a solver often involves its complexity at termination or convergence rate, reflecting its speed and efficiency. *Iteration complexity* analyzes the number of iterations (steps) required to obtain *close enough* solutions, parameterized by tolerance ϵ. The metric that measures closeness depends on the problem class and context (e.g., $(f(X_k) - f^*)$ in convex settings and $\|\nabla f(X_k)\|$ in noncovex ones). In lieu of $f(X_k)$ or $\|\nabla f(X_k)\|$, given their unavailability, the algorithm can use their estimates. In discrete settings or for cases where f^* is unknown, the algorithm may use other metrics for optimality gap, such as distance between consecutive iterates. Iteration complexity concerns characterizing, as a function of ϵ, the random variable T_ϵ, which is the first iteration at which the solver finds an ϵ-optimal solution in some probabilistic sense. However, in SO, iteration complexity may not reveal the *effort* expended to "ϵ-solve" a problem. Effort refers to the number of simulation runs, since one run (of large simulation models) typically takes more time than most mathematical operations during each iteration of the algorithm. Algorithms that may take the same number of steps to reach ϵ-optimality may vastly differ in number of simulation runs. This has led to analyzing stochastic algorithms in terms of *sample complexity* (a.k.a. *simulation cost*). Letting random variable W_ϵ be the total number of simulation runs up to iteration T_ϵ, sample complexity involves characterizing its moments or tails as a function of ϵ.

For intuition, two algorithms ASTRO and SA (Stochastic Approximation) (Nemirovski et al. 2009) that we will discuss later both have proven L_1 iteration complexity of $\mathcal{O}(\epsilon^{-2})$ for stochastic nonconvex problems; however, ASTRO attains $\mathcal{O}(\epsilon^{-2}\log\epsilon^{-1})$ L_1 sample complexity, while the best proven SA result with variance reduction is $\mathcal{O}(\epsilon^{-3})$. This means to reach $\epsilon = 0.1$-optimality, SA may require thousands of simulation runs while ASTRO may only need hundreds. A reciprocal comparison with a fixed simulation budget b suggests SA reaches $\mathcal{O}(b^{-1/3})$-distance on average while ASTRO nears $\mathcal{O}(b^{-1/2})$. The latter comparison is one known as the **finite-time analysis**, monitoring performance as portions of the budget are expended. Deterministic benchmarking (e.g., CUTest (Fowkes et al. 2022) and COCO (Hansen et al. 2021)) seeks speed testing plots for such comparisons. A specialized framework for SO, SimOpt (Eckman et al. 2023), is a growing open-source library with graphical interfaces that compares average performance *and* variability of varying classes of solvers (continuous and discrete) when tested on simulation problems.

1.2 Unique Challenges and Opportunities in Simulation Optimization

Compared to typical approaches in stochastic optimization literature, SO problems have unique difficulties. First, recent trends of stochastic per-iteration sample sizes complicates the generated stochastic process $\{X_k\}$ by randomizing time (number of runs) between two iterations. Second, dependence between iterates of a solver creates a well-known challenge in analysis of algorithms. Third, in addition to the recognized difficulties of unknown constants in deterministic analysis, the unknown variance structure of stochastic values, e.g., $\mathrm{V}_F(x) = \mathbb{E}_\xi[(F(x,\xi)-f(x))^2]$, and their complex heteroscedastic behavior is another obstacle.

There are also several features in an SO problem such as (i) ability to leverage estimates of stochastic noise (standard errors or quantiles) and mechanisms to impact more than just average performance; (ii) variance reduction stemming from dependence between estimates and reuse of data, or free access to other simulation-generated variables; and (iii) possible benefits of sample-path structures in F rather than population-level structures in f. More specifically, SO methods embrace a close dialogue between the inferred noise from the stochastic oracle and the inferred function's landscape from the optimization engine.

1.3 Paper Organization

With a focus on the synergy between sampling in the *noise space* and moving in the *decision space*, we skip some SO fundamentals; see some of landmark surveys by Fu (2002), Hong and Nelson (2009), Jian and Henderson (2015). We start by discussing three sampling paradigms (Section 2), namely, sample average approximation (SAA), retrospective approximation (RA), and adaptive sampling (AS), linking to well-known budget allocation schemes in R&S literature and sequential estimation in statistics (Ghosh and Mukhopadhyay 1979). By emphasizing the sampling strategies beyond any solution method framework, we provide a viewpoint to estimation mechanisms whose potential benefits in different algorithm classes' performance may vary and are mostly unexplored. Then a review of continuous (Section 3) and discrete (Section 4) SO methods follows with examples. Due to space limit, we provide brief commentary on approaches towards constrained SO and other trends (Section 5).

2 SAMPLING (IN THE NOISE SPACE)

A classic approach to SO is SAA or so-called a *mini-batch* approach, fixing $\xi_1, \xi_2, \cdots, \xi_n$ as i.i.d. (independent and identically distributed) realizations of ξ, and instead of a solution to problem (P) seeking

$$\min_{x\in\mathcal{X}} \{\bar{F}_n(x) := n^{-1} \sum_{j=1}^{n} F(x,\xi_j)\}. \tag{SAA}$$

Problem (SAA) is now deterministic and viewed as a *sample-path problem* since we only search n sample paths for the optimal solution. A deterministic algorithm is typically run until either reaching ϵ-optimality or exhausting the simulation budget b. See Algorithm 1 for main steps.

Algorithm 1 Sample Average Approximation

```
1: Input: Initial point $x_0$. Solver of interest. Optimality gap proxy function $\phi : \mathcal{X} \to \mathbb{R}^+$. Tolerance value $\epsilon$.
   Sample size $n$. Sampling budget $b$. Iteration number $i = 0$. Total samples counter $W = 0$.
2: while $W \leq b$ or $\phi(X_i) > \epsilon$ do
3:     Find $X_{i+1}$ using estimated quantities needed by the solver, at or around $X_i$, via SAA with $n$ replications.
4:     Increment $W$ with the total simulation runs in iteration $i$ and set $i = i + 1$.
5: end while
```

Let the asymptotic solution to (SAA) be X_n^*, with the finite-time approximate $\hat{X}^*_{n,b,\epsilon}$ that depends on termination criteria ϵ and b. Can one claim that $\bar{F}_n(X_n^*) \to f^*$ and $X_n^* \to x^*$ as $n \to \infty$ in some probabilistic sense? See (Homem-de Mello and Bayraksan 2014a) for an excellent review for this question, listing *strong uniform consistency* of estiamtor $\bar{F}_n$ on $\mathcal{X}$, i.e., $\sup_{x\in\mathcal{X}} |\bar{F}_n(x) - f(x)| \xrightarrow[n\to\infty]{wp1} 0$, (which generally holds at the rate of $1/\sqrt{n}$) as a requirement for wp1 convergence, i.e., $\bar{F}_n(X_n^*) \xrightarrow[n\to\infty]{wp1} f^*$.

2.1 Sample Size

Given that the convergence guarantees are asymptotic, one naturally may ask how large should n be to close the gap between X_n^* and x^*? Shapiro et al. (2021) showed that if f is bounded and κ_f-Lipschitz continuous, i.e., $|f(x_1) - f(x_2)| \leq \kappa_f \|x_1 - x_2\|$ for all $x_1, x_2 \in \mathcal{X}$ and some $\kappa_f > 0$, and F has finite variance, i.e, $\mathrm{V}_F(x) \leq \sigma^2$ for all $x \in \mathcal{X}$, then in order for a δ-optimal solution of problem (SAA) to be among the ϵ-optimal solutions ($\delta < \epsilon$) of problem (P) with probability at least $1-\alpha$, we need $n \geq \frac{12\sigma^2}{(\epsilon-\delta)^2} \log \frac{1}{\alpha} \left(\frac{2\kappa_f D}{\epsilon-\delta}\right)^d$ samples with feasible space's diameter $D = \sup_{x,y\in\mathcal{X}} \|x-y\|_\infty$. Even problems with modest choices such as $\epsilon = 0.1, \delta = 0.05, D = 100, \kappa_f = 10, \sigma^2 = 1$, and $\alpha = 0.05$ lead to extremely large $n \approx 167K$ to obtain this guarantee. A similar lower bound exists in bounded discrete spaces, i.e., $n \geq \frac{3\sigma^2}{(\epsilon-\delta)^2} \log \frac{|\mathcal{X}|}{\alpha}$ with $|\mathcal{X}| = 2^d$ for binary decisions. Alarmingly, this sample size requirement is for every x evaluated during the optimization, irrespective of $\mathrm{V}_F(x)$ or closeness to optimality; this renders SAA far from efficiency. In principle, we must let $n \to \infty$ for convergence but how fast should it be to maintain efficiency?

RA (Chen and Schmeiser 2001) is a sampling strategy whose main idea is to solve multiple (SAA) problems sequentially, each with a fixed sample size until it reaches a pre-determined level of optimality (tolerance). Then the tolerance decreases and the sample size is increased following a prescribed operator. Simply put, RA has outer iterations that produce $\{X_k\}$ progressively closer to x^* via the tolerance sequence $\{\epsilon_k\}$, while between consecutive outer iterations, a sequence of inner iterations with a fixed sample size $n_k = \mathcal{O}(\epsilon_k^{-2})$–Monte Carlo error rate–solves a new sample-path problem until exhausting the outer loop's budget $\{b_k\}$. The term "retrospective" points to each new sample-path problem using the previous solution for a warm start. The process is simple as specified in Algorithm 2. Similar to SAA, RA is broad enough and can be applied with any solver of choice, or even a collection of solvers; yet it is largely unexplored.

Algorithm 2 Retrospective Approximation

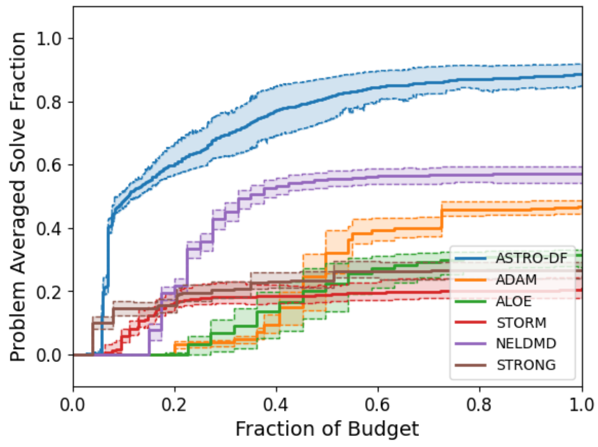
```
1: Input: Initial point $x_0$. A solver of interest. Optimality gap proxy function $\phi : \mathcal{X} \to \mathbb{R}^+$. A decreasing
   tolerance sequence $\{\epsilon_k\}$. An increasing sample size sequence $\{n_k\}$. A sequence of budgets $\{b_k\}$.
2: for $k = 1, 2, \cdots$ do
3:     Set $x_0 = X_k$ (warm start) and run Algorithm 1 with $\epsilon_k$, $n_k$ and $b_k$ to return $X_{k+1}$.
4: end for
```

Another strategy is *adaptive sampling* (AS), which follows the sequential nature of RA but relies heavily on the information received in each step of the algorithm. Rather than fixing a tolerance ahead of running the algorithm, AS uses a proxy function for optimality gap, $\phi(X_k)$, and keeps adding new samples until the estimated standard error of the estimated quantity $e_n(X_k)$ (e.g., if the quantity used by the solver is $\bar{F}_n(X_k)$, then $e_n(X_k) = \hat{\sigma}_n(X_k)/\sqrt{n} = \sqrt{\sum_{i=1}^n \left(F(X_k,\xi_i) - \bar{F}_n(X_k)\right)^2/n})$ is bounded by that optimality gap proxy (Step 4 in Algorithm 3). This *stopping time* sampling strategy provides clear intuition (saving budget for good solutions later in the search) and a dramatic advantage in implementation (Byrd et al. 2012; Shashaani et al. 2018; Bollapragada et al. 2018) (See Fig. 1).

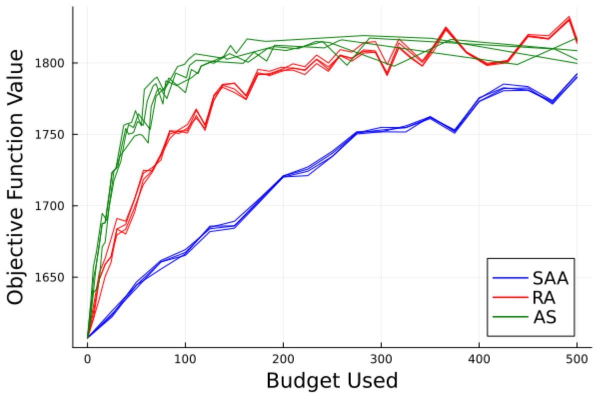
Of the three sampling techniques, AS can be considered the most dynamic as every simulation run can be used to learn the sample size. The adaptivity in AS is both because the tolerance is stipulated from the algorithm's learned distance to optimality and because the local variance is learned from simulation runs, gaining flexibility for problems whose noise level vastly changes at different solutions. This is reminiscent of the sampling procedures used for R&S with fixed precision, which are widely applied to selecting the best among a relatively small number of solutions with probabilistic guarantees (see Fu and Henderson (2017)'s R&S history and references therein). The main requirement in using AS with a known deterministic solver is to find the right proxy function $\phi(X_k)$. The preference is on using quantities such as the step size Δ_k that are known at the beginning of iteration k and stay unchanged as new simulation runs at X_k are collected (as opposed to using $\bar{F}_{N_k}(X_k)$ or estimated gradient, e.g.). In trust-region methods (described in Section 3.4), depending on (i) sample-path structure (e.g., smoothness), (ii) access to unbiased gradient estimates, and (iii) stochastic noise behavior (e.g., subexponential tails), $\phi(X_k)$ ranges from $\mathcal{O}(\Delta_k^2/\sqrt{k})$ to $\mathcal{O}(1/\log k)$, the latter leading to a much smaller rate of sampling (as $\Delta_k \xrightarrow[k\to\infty]{wp1} 0$). The other difficulty is the stopping time nature of AS since $\hat{\sigma}_n$ changes with every new sample added. A remedy is a two-stage approach to estimate $\hat{\sigma}_n$ in the first stage and compute N_k in the second stage (Ha et al. 2024).

Algorithm 3 Adaptive Sampling (Approximation)

```
1: Input: Initial point $x_0$. A solver of interest. An increasing sample size lower bound sequence $\{n_k\}$. A proxy
   function $\phi(\cdot)$ for optimality gap. Iteration number $k = 0$. Total samples counter $W = 0$.
2: while $W \le b$ or $\phi(X_k) > \epsilon$ do
3:     Estimate quantities needed by the solver with $N_k = \min\{n \ge n_k :\ e_n(X_k) \le \phi(X_k)\}$ samples.
4:     Use the solver to find $X_{k+1}$ and evaluate it with $N_{k+1} = \min\{n \ge n_k : e_n(X_{k+1}) \le \phi(X_{k+1})\}$ samples.
4:     Increment $W$ with the total simulation runs in iteration $k$ and set $k = k + 1$.
5: end while
```



(a) The solvability profile (% of 60 problems from SimOpt library, solved to 0.1-optimality) of ASTRO-DF compared to SAA-based solvers signifies advantage of AS across problems.



(b) Solving a newsvendor problem via fixed-step stochastic gradient method reveals increasingly fewer but better solutions with RA/AS and faster progress with limited simulation budget.

Figure 1: The sampling strategy affects the finite-time performance of SO algorithms. Note the rapid progress with AS-based solvers at the beginning of the search and slow down towards the end.

Besides SAA, RA, and AS, pre-selected total simulation cost in every algorithm milestone can be allocated to different points being evaluated. Budget allocation strategies such as OCBA (Optimal Computing Budget Allocation) (Lee et al. 2010) and others fall in this category. While useful for algorithms that must evaluate a collection of points in each iteration (direct search methods or evolutionary algorithms), budget allocation strategies may not be flexible for other optimization algorithms since the number of iterations before termination T_ϵ is a stopping time random variable itself.

Evaluation: When evaluating and comparing solvers of any of these algorithms, it is crucial to independently assess the value of solutions using a separate stream of random numbers not used during optimization, i.e., evaluating the terminal solution $\hat{X}_b^*$, with $\bar{F}_{n'}(\hat{X}_b^*)$ using n' (large) independent samples. This avoids *optimization bias* (Mak et al. 1999), which occurs because the estimated values used by the solver are conditioned on the information that led it to select those solutions, hence biased for performance evaluation.

2.2 Variance Reduction

Besides the choice of sample size, one can be more deliberate in the selection of samples to improve the efficiency of the algorithm. An established idea for variance reduction is the use of *common random numbers* (CRN). Using CRN when calling the oracle means "holding the random number fixed" (at say ξ) as the oracle is called at the points $x_1, x_2, \dots, x_{2d+1}$, that is, the inputs to the oracle are $(x_1,\xi),(x_2,\xi),\dots,(x_{2d+1},\xi)$. The oracle will then return $F(x_1,\xi), F(x_2,\xi),\dots,F(x_{2d+1},\xi)$. This is in contrast to independent sampling with $(x_1,\xi_1),(x_2,\xi_2),\dots,(x_{2d+1},\xi_{2d+1})$ inputs to the oracle, with i.i.d. $\xi_1,\xi_2,\dots,\xi_{2d+1}$. In other literature, CRN is also referred to as the "simultaneous queries" and considered a powerful tool (Arjevani et al. 2023). CRN enables leveraging sample-path structure, e.g., Lipschitz continuity

$$|F(x_1,\xi)-F(x_2,\xi)| \le \kappa_F(\xi)\|x_1-x_2\|,\ \forall x_1,x_2 \in \mathcal{X} \tag{1}$$

for all $\xi \in \Xi$ except a set of measure 0. Such properties can tremendously alleviate the cost of sampling and improve the per-iteration sampling complexity (Ha et al. 2024; Rinaldi et al. 2024).

While CRN enforces staying on the same sample-path with the same ξ, other variance reduction techniques such as control variates, stratified sampling, importance sampling, and Latin hypercube sampling either correct (or modify weight of) an output $F(\cdot,\xi)$ or control the choice of ξ. Control variate technique (Nelson 1990) leverages other simulation generated data in one oracle call and their correlation with the objective function to correct for the error, via $\bar{F}_N^{\text{cv}}(x) = \bar{F}_N(x) - \frac{\widehat{\text{Cov}}(F(x,\xi),C(x,\xi))}{\mathbb{V}_C(x)}\frac{1}{N}\sum_{i=1}^N (C(x,\xi_i)-\mathbb{E}[C(x,\xi)])$. In this new estimator, another simulation output $C(x,\xi)$ (which could be one used for stochastic constraints) whose mean and variance may be known is used. Additionally, *database Monte Carlo* ideas allow the use of previously simulated data (perhaps at a different solution) for control variates (Rosenbaum and Staum 2015). While the reduction in variance in $\bar{F}_N^{\text{cv}}(\cdot)$ has been well-studied, whether or not control variates can directly reduce the simulation cost of an SO algorithm is yet to be shown.

Stratified sampling often divides the noise space Ξ into partitions assuming that the distribution of F for a fixed x varies drastically from one partition to another. This is a harder idea to justify unless with evidence of heterogeneity. But harder than that is implementation during optimization unless there is a strong expert judgement for stratification used for all solutions. Dynamic stratification with binary trees (in the noise space) or control-variate like ideas have proven effective in practice (Jain et al. 2024). In terms of their help with analysis, there are promising directions for alleviating the sampling requirement, as shown in a special case of Latin hypercube sampling, where under smooth sample paths the error rate is $\mathcal{O}(n^{-3/2})$ in lieu of $\mathcal{O}(n^{-1/2})$ Monte Carlo rate (Asmussen and Glynn 2007). Benefits of this and other variance reduction techniques on overall algorithm complexity and risk are open questions that may vary for continuous and discrete SO, which we will discuss next.

3 CONTINUOUS SIMULATION OPTIMIZATION

Example 1: The Newsvendor Problem Consider the problem of finding the optimal amount of a commodity $x \in \mathbb{R}$ to purchase at the beginning of the day that maximizes the expected profit by the end of the day when demand is uncertain. Given c as the purchase cost, p as the selling cost, w as the salvage price, and ξ as the random daily demand, one simulation output (profit given x on a random day) is $F(x,\xi) = p\min\{x,\xi\} - cx + w\max\{x-\xi,0\}$. When a random demand is realized, then we can immediately calculate the newsvendor's profit under the choice of x. The point is that profit has a likelihood based on the probability measure on ξ.

This is an example of a continuous optimization problem, since $\mathcal{X} = \mathbb{R}$ in continuous. For this class of functions, certain properties such as continuity, differentiability, smoothness, and convexity is important for optimization. But one of the first things that a continuous optimization solver will want is the derivative of the function. Derivatives can be exploited to point the solver to a promising (descent) directions. With a good direction, the solver should also decide the step size in that direction to make an improvement in the next *incumbent* solution. The algorithm then iteratively continues this process until termination.

3.1 Derivative Estimation

Derivatives provide information about necessary and sufficient conditions for local optimality. When we are at a locally optimal solution, moving to any other direction with small steps will worsen the objective function value. With a Taylor expansion at x^*, and noting all solutions nearby must be worse, note

$$f(x^*+s) = f(x^*) + \nabla f(x^*)^\intercal s + \frac{1}{2}s^\intercal \nabla^2 f(x^*)s + \mathcal{O}(\|s\|^3) \geq f(x^*) \text{ for all small } s \in \mathbb{R}^d,$$

which means (i) $\nabla f(x^*)^\intercal s \leq 0$ for all small $s \in \mathbb{R}^d$, hence $\nabla f(x^*) = 0$—necessary optimality condition, and (ii) $s^\intercal \nabla^2 f(x^*)s \leq 0$ for all $s \in \mathbb{R}^d$, implying a positive curvature at x^*, i.e., $\nabla^2 f(x^*) \succ 0$ (positive definite Hessian)—sufficient condition. Proving the positive definiteness of the Hessian yields a *second-order* convergence; however, as mentioned in Section 1.1, first-order stationarity is often adequate in stochastic settings. For the newsvendor problem, one can derive the derivative of F, denoted by ∇F as $\nabla F(x,\xi) = (p-c)\mathbb{I}(x<\xi) + (w-c)\mathbb{I}(x>\xi)$. Thus with a realized demand ξ, we can directly compute the derivative of the profit at x. This is a *first-order* stochastic oracle, whereby a direct observation of both the function value and derivative is attainable from each simulation run. Accordingly the gradient estimate can be computed via SAA similar to the function estimate. However, there is an implicit assumption:

$$\nabla f(x) = \nabla_x \mathbb{E}[F(x,\xi)] = \mathbb{E}[\nabla_x F(x,\xi)] \tag{2}$$

which may not hold. The necessary condition for (2) is interchangability of expectation (integral) and derivative operator (limit). One condition for interchangeability (via Dominated Convergence Theorem) is Lipschitz sample paths as in (1) where $\mathbb{E}_\xi[\kappa_F(\xi)] < \infty$. Another is through uniform integrability of F.

The derivative explicitly derived in Example 1 is an application of the *infinitesimal perturbation analysis* (IPA) (Chong and Ramadge 1993). Under differntiability of F, this derivation of ∇F (the sample-path gradient) can be done with full access to its mathematical expression; when no access or too complicated, automatic differentiation tools (Ford et al. 2022) may obtain IPA estimators. Back-propagation in neural networks is an automatic differentiation, providing the IPA derivative. If (2) holds, then IPA yields unbiased gradients; though even without unbiasedness guarantees, IPA estimators may still be useful.

Another widely known technique for derivative estimation is the likelihood ratio (LR) technique (Glynn 1987). LR re-structures the problem by putting all the influence of x on the sampling distribution rather than on the function value, i.e.,

$$f(x) = \int F(\xi)p_x(\xi)d\xi = \int F(\xi)\frac{p_x(\xi)}{q(\xi)}q(\xi)d\xi = \int F(\xi)\ell_x(\xi)q(\xi)d\xi,$$

where q is another probability measure independent of x and $\ell_x(\xi) = \frac{p_x(\xi)}{q(\xi)}$ is the likelihood ratio. Then, under conditions that allow interchangability of integral and limit in this set-up, we get $\nabla f(x) = \tilde{\mathbb{E}}[F(\xi)\nabla_x \ell_x(\xi)]$, with $\tilde{\mathbb{E}}$ representing expectation under the new probability measure q. A good choice of q is $p_{x=X_k}$ yielding $\nabla_x \ell_x(\xi) = \nabla_x \ln p_{x=X_k}(\xi)$, also known as the *score function*. LR is easier to implement as long as the reformulation of the problem is not difficult. However, its main disadvantage is large variance that grows linearly with the dimension of Ξ. While the above summarizes some popular derivative estimation methods, we recommend (Fu 2015) for a comprehensive review.

3.1.1 Derivative-free Methods

Unlike Example 1, in many SO problems $\nabla_x F(x,\xi)$ is not directly observable and the function value is the only information available, i.e., the *zeroth-order* stochastic oracle. The effort to augment a zeroth-order oracle with an IPA-like derivative observations is not convenient in many cases. There are also problems whose sample paths are not differentiable; for example, when the objective function is a probability, the step function sample path $F(x,\xi) = \mathbb{I}(A(x,\xi) \in a)$ for event A and region a has 0 derivative everywhere except at points of discontinuity where it is not defined. In other cases, sample paths may not be smooth despite being differentiable. In these and many other cases, it is important to have solvers that do not explicitly need derivative information. These classes of algorithms are called the derivative-free (DF) solvers (Conn et al. 2009; Rios and Sahinidis 2013; Larson et al. 2019). DF solvers either approximate the gradient from function values of a collection of points in each iteration, or move in $\mathcal{X}$ directly.

A predominant technique to *approximate* derivatives when derivations above are not possible is the finite-difference (FD) approximation. For example, forward FD approximates the derivative with

$$G_k := \left[\frac{F(X_k + h_k e_1, \xi_k^{1+}) - F(X_k, \xi_k)}{h_k}, \cdots, \frac{F(X_k + h_k e_d, \xi_k^{d+}) - F(X_k, \xi_k)}{h_k} \right], \tag{3}$$

where e_i is the i-th unit vector, and the *perturbation* sequence $\{h_k\}$ may be chosen in each iteration appropriately. One G_k evaluation therefore requires running the simulation at $d+1$ solutions. Central FD is similar to (3) but perturbing forward and backward and dividing by $2h_k$, hence evaluating $2d$ many points. The gradient estimator has a bias that disappears as $h_k \to 0$. However, smaller h_k blows up the variance, rendering an SAA warranted to control the variance with sample size n_k, with $\bar{G}_{N_k}(X_k) = N_k^{-1} \sum_{j=1}^{N_k} G_{k,j}$, where each j-th copy $G_{k,j}$ uses samples $(\xi_{k,j}^{i+}, \xi_{k,j}\ i = 1, 2, \cdots, d)$. Balancing the squared bias $\mathcal{O}(h_k^2)$ and variance in i-th direction $\mathbb{V}([\bar{G}_{N_k}(X_k)]_i) = N_k^{-1} h_k^{-2} \mathbb{V}(F(X_k + h_k e_i, \xi_k^{i+}) - F(X_k, \xi_k))$ requires $n_k = \mathcal{O}(h_k^{-4})$, which is very expensive. CRN can help by reducing the required sample size to $n = \mathcal{O}(h_k^{-3})$ since

$$\mathbb{V}\left(F(X_k + h_k e_i, \xi_k^{i+}) - F(X_k, \xi_k)\right) = \begin{cases} \mathcal{O}(1) & \xi_k^{i+} \perp \xi_k \text{ (independent random variables)}, \\ \mathcal{O}(h_k) & \xi_k^{i+} = \xi_k, \\ \mathcal{O}(h_k^2) & \xi_k^{i+} = \xi_k \ \&\ F(\cdot, \xi) \text{ satisfies (1) } \forall \xi. \end{cases}$$

Directional direct search (DS) methods are a different class of algorithms that sample points in the decision space with fixed or randomized directions (Kolda et al. 2003). The first algorithms in this class used coordinates e_i resembling FD. In DS methods, a set of polling solutions $\mathcal{X}_k = \{X_k + \Delta_k p :\ p \in \mathcal{P}_k\}$ are evaluated using a direction set $\mathcal{P}_k = \{p \in \mathcal{X} \subseteq \mathbb{R}^d :\ \min_{v \in \mathcal{X}} \frac{d^\intercal v}{\|d\| \|v\|} > 0\}$ that spans the feasible space. A solution that yields the best improvement to the incumbent X_k by at least a sufficient reduction criteria, i.e., $\inf_{p \in \mathcal{P}_k} \bar{F}_{N_k}(X_k) - \bar{F}_{N_k}(X_k + \Delta_k p) \geq c\Delta_k^2$ with a constant $c > 0$ is chosen as the next incumbent and the step size increases for a wider search in next iteration. Otherwise, if no such point in found among the polling solutions, the current incumbent does not change and the step size is reduced to look in closer neighborhoods for improvement. The randomized and opportunistic polling reduce the dependence of the complexity on the problem dimension. Mesh or grid-based methods such as pattern search are also a group of DS solvers that only require simple decrease; mesh adaptive direct search (MADS) is the basis of a well-known software called NOMAD (Le Digabel 2009). Another example is DS is the Nelder-Mead algorithm (Nelder and Mead 1965) that in each iteration performs five main operations on the simplex to create a new one: reflection, expansion, outside contraction, inside contraction, and shrinking. The new point generated with each operation follows $X = X_k^c + \alpha(X_k^c - X_k^w)$, where X_k^c is the centroid of all points except the worst one X_k^w and α is a scalar. In this recursion the direction of the line connecting the centroid and the worst point may be seen as a possible descent direction. Two optimality gap proxies often used here are the size of the simplex and the standard deviation of the function values in the simplex. Barton and Ivey Jr (1996) modified this algorithm for SO, re-evaluations are warranted before shrinking the simplex in response to lack of progress (which signals that we may be close to an optimal region). particularly a promising framework for nonsmooth problems (Garmanjani and Vicente 2013).

Evolutionary methods such as the Genetic Algorithms (GA) (Boesel et al. 2003) or Particle Swarm Optimization (PSO) (Zhang et al. 2016) are among other DF methods include that are population based and rely on the intelligence a pool of genes/agents that evolves as more knowledge is obtained. GA learns from the population by mutating genes (bits) or crossovers and discarding bad points from one generation (iteration) to another. PSO sends a number of particles in difference directions (velocity) but corrects their trajectory using their best and population's best, which resembles the averaging technique we shall see in the stochastic approximation algorithms. Given their intuitive mechanisms and easy implementations, these methods have been widely used and often have available resources online. However, some lacking theoretical foundations has led to little developments to reduce their dependence on the problem dimension.

3.2 Stochastic Gradient / Stochastic Approximation

The classic stochastic approximation (SA) method by Robbins and Monro (1951) takes iterative steps in the opposite direction of the gradient, generating a sequence of iterates with a simple recursion

$$X_{k+1} = X_k - \Delta_k \nabla_x F(X_k, \xi_k). \tag{4}$$

Instead of using the recursion directly, its projection onto a compact set in $\mathcal{X}$ can help with generating feasible solutions and also alleviate SA's sensitivity in large search spaces (Andradóttir 1995).

It is well-known that the deterministic version of (4), the gradient descent method, converges to a first-order stationary point as long as $\Delta_k = \Delta < 2/\kappa_g$ at the rate $\mathcal{O}(k^{-1})$; where κ_g is the Lipschitz constant of function gradients, i.e., $\|\nabla f(x) - \nabla f(y)\| \leq \kappa_g \|x - y\|$ for all $x, y \in \mathcal{X}$ (f is κ_g-smooth). However, the stochastic counterpart does not converge with a fixed Δ. The reason is, given a κ_g-smooth (and for simplicity, μ-strongly convex) function f, $\Delta < 2/\kappa_g$, $\mathbb{E}[\nabla_x F(X_k, \xi) \mid \mathcal{F}_k] = \nabla f(X_k)$, and $\mathbb{V}(\nabla_x F(X_k, \xi) \mid \mathcal{F}_k) \leq \sigma^2$ for $X_k \in \mathcal{F}_k$, the *fundamental inequality for smooth functions* implies

$$\mathbb{E}[f(X_{k+1}) - f^* \mid \mathcal{F}_k] \leq (f(X_k) - f^*)(1 - 2\mu\Delta) + \frac{\kappa_g \Delta^2}{2}\sigma^2. \tag{5}$$

In other words, the optimality gap in each iteration improves by a factor (identical to the gradient descent analysis and providing geometrically fast improvements) plus another term that depends on σ^2 and does not go away with k. This means the algorithm nears optimality up to a neighborhood (whose size depends on the noise level in the stochastic gradient) and then exhibits a random walk in that neighborhood without ever converging. The main direction to drive this added term to 0 was the use of diminishing step sizes Δ_k as $k \to \infty$. Robbins Siegmund Theorem shows $\sum_k \Delta_k = \infty$ yet $\sum_k \Delta_k^2 < \infty$, e.g., $\Delta_k = \mathcal{O}(k^a)$ with $a \in (1/2, 1]$ (preventing step size to decay too quickly) guarantees wp1 convergence.

The choice of Δ_k has proven crucial. Conservatively small Δ_k slows the progress. The preference is always to keep the step size large for rapid progress, but too large Δ_k leads to oscillations and zigzags. Finding the iteration where reducing Δ_k is most beneficial is nontrivial. Some adaptive step size selection methods reduce Δ_k only with significant changes in the direction of consequent iterates to get unstuck, but keep up pace otherwise (Kesten 1958). Another direction is the iterate averaging and defining iterates as $\tilde{X}_k = k^{-1} \sum_{i=1}^k X_k$, with numerous variations in weight systems or sliding windows, that de-emphasize the effect of step size (Polyak and Juditsky 1992). For an excellent review of other advances in SA, including techniques that combine first- and zeroth-order derivative estimates, see (Chau and Fu 2014). Notably, these attempts challenge the third term's removal in (5) by means of the step size. Instead of using one gradient instance, another strategy is using a gradient estimate by averaging several i.i.d. observed stochastic gradients, i.e., $X_{k+1} = X_k - \Delta_k \bar{G}_{N_k}(X_k)$ with $\bar{G}_{N_k}(X_k) = \frac{1}{N_k} \sum_{j=1}^{N_k} \nabla F(X_k, \xi_{k,j})$. This imposes an N_k^{-1} coefficient to the third term (standard error) that vanishes as long as $N_k \xrightarrow[k\to\infty]{} \infty$. Recent adaptive N_k strategies by Bollapragada et al. (2018) ensure the estimated gradient is a descent direction whp; complexity analysis in this class of algorithms remains open. Variance-reduced versions of SA, such as SPIDER and SVRG, use simultaneous queries to reduce the sample complexity from $\mathcal{O}(\epsilon^{-4})$ to $\mathcal{O}(\epsilon^{-3})$.

Kiefer-Wolfowitz: Volatility of SA is exacerbated in the derivative-free settings; the SA algorithm that uses FD gradients is called the Kiefer Wolfovitz (KW) algorithm which is the same as (3.2) but with $\nabla F(X_k, \xi_{i,k})$ replaced by (3) (Kiefer and Wolfowitz 1952). The perturbation size for KW needs to satisfy

(i) $h_k \to 0$ as $k \to \infty$, (ii) $\sum_k \Delta_k h_k < \infty$ (perturbation size grows at least as fast as the step size), and (iii) $\sum_k (\Delta_k/h_k)^2 < \infty$. The trade-off between the two is that small h_k leads to noisy gradients knowing that FD gradients. On the other hand, large Δ_k leads to large oscillations. Typical choices here are $\Delta_k = \mathcal{O}(k^{-1})$ and $h_k = \mathcal{O}(k^{-1/4})$. As evident, for one iteration of the KW algorithm, $2d$ points (in the case of central finite difference) and $d+1$ points (in the case of forward finite difference) need to be evaluated. Whereas in the first-order oracle, we only need the evaluation of one point (irrespective of d). This is why DF methods are very expensive. A popular remedy for this curse of dimensionality is the Simultaneous Perturbation Stochastic Approximation (SPSA) method (Spall 1992; Spall 2005) that only requires one additional point evaluation in a random dimension, i.e., using $X_{k+1} = X_k - \Delta_k \bar{G}_{N_k}(X_k)$ where

$$[\bar{G}_{N_k}(X_k)]_i = \frac{1}{N_k}\sum_{j=1}^{N_k} \frac{F(X_k + h_k U_k, \xi_{k,j}^{+}) - F(X_k - h_k U_k, \xi_{k,j}^{-})}{2h_k [U_k]_i},$$

using $U_k \in \mathbb{R}^d$ as a random vector with elements sampled i.i.d. from a uniform distribution. This is on par with the notion of randomized directional DS methods (Gratton et al. 2015) that also remove dependence of $|\mathcal{P}_k|$ (see Section 3.1.1) on d. KW's success concerns its sensitivity to x_0, Δ_k, and the dilemma of balancing the noise in the derivative estimates (that now contain bias) and the progress of the algorithm.

3.3 Quasi-Newton Recursions

Recall that the typical SA recursion relies solely on the gradient values. A different remedy to the issues above has been the direct or indirect inclusion of the second order derivatives, as well as the reduction criterion in the function values. Broadly, inclusion of these elements is aligned with the Newton-Raphson numerical methods. For a quick review of Newton methods, recall that for κ_g-smooth functions, one can build an upper bound approximation $m_k(s) = f(X_k) + \nabla f(X_k)^\intercal s + \frac{1}{2\Delta_k}\|s\|^2$, for all $s \in \mathcal{X}$ that are close to X_k (using $\Delta_k < 2/\kappa_g$). The idea behind the Newton methods is to find the minimizer of this quadratic approximation; setting $\nabla_s m_k(s) = 0$ yields the solution $s^* = -\Delta_k \nabla f(X_k)$, which is the well-known gradient descent recursion. This approximation will be weak if the function's curvature is wobbly (i.e., large condition number of $\nabla^2 f(X_k)$ which is the ratio of its largest to smallest eigenvalue). This causes moving along the gradient to lead to many zigzags until the algorithm finds a good direction. A better approximation that can save those zigzaggy steps and find a good direction early and enable taking large steps in that direction must take the curvature information into account, i.e.,

$$m_k(s) = f(X_k) + \nabla f(X_k)^\intercal s + \frac{1}{2} s^\intercal \nabla^2 f(X_k) s, \tag{6}$$

where the Hessian replaces the identity matrix in the first approximation, where $\|s\|^2 = s^\intercal I_d s$ (I_d being an identity matrix in $\mathbb{R}^{d\times d}$). Then minimizing this approximation yields $s^* = -(\nabla^2 f(X_k))^{-1}\nabla f(X_k)$, a.k.a., the Newton-Raphson recursion. When $\nabla^2 f(X_k)$ is not accessible or expensive to compute which is a prevalent issue in SO, one would instead use a symmetric positive definite matrix, denoted by B_k, to get a *quasi*-Newton recursion. The objective with the quasi Newton methods is to have a good representative of the function curvature with easy and inexpensive updating (without requiring additional simulation runs) in each iteration; a common method is BFGS with limited memory (Liu and Nocedal 1989).

3.4 Line Search and Trust Region Methods

Despite is far-reaching reputation, SA is still subject to its Achille's hill—the step size—especially in functions with negative curvatures. Line search (LS) methods have attempted to solve the step problem in SA by fixing a "good" direction and choosing how far to move in that direction by solving a one-dimensional problem $\min_{\Delta>0} f(X_k + \Delta P_k)$, where $P_k \in \mathcal{X}$ is a descent direction (e.g., gradient descent, Newton, quasi Newton, or conjugate gradients). Typically a large Δ initiates LS, and is reduced in a backtracking look if minimum reduction $f(X_k + \Delta P_k) > f(X_k) + c'\nabla f(X_k)^\intercal P_k$ is not achieved (Armijo condition).

$c' < 0.001$ makes the linear approximation an almost flat line. For a speed up, if the slope at the trial point signals potential faster reduction in that direction, i.e., $|\nabla f(X_k + \Delta P_k)^\intercal P_k| \leq c''|\nabla f(X_k)^\intercal P_k|$ for $c'' \in [0.1, 0.9]$ (curvature condition), then the step size can increase. Unlike SA, LS uses both the function value and the gradient at the trial point to accept it; this conservativeness tremendously helps in the stochastic settings. SASS (Jin et al. 2024) is a recent stochastic LS method that accepts $X_k - \Delta_k B_k^{-1}\bar{G}_{N_k}(X_k)$ if the reduction in the function estimates is more than $\Delta_k(\bar{G}_{N_k}(X_k))^\intercal B_k^{-1}\bar{G}_{N_k}(X_k)$.

Trust regions (TR) circumvent this by just minimizing the quadratic approximation of the function in a Δ_k-ball around the incumbent solution X_k. Therefore, TR evolves in the search space by looking in compact regions $\mathcal{X}_k := \{s \in \mathcal{X} : \|s\| \leq \Delta_k\}$ whose size increases when sufficient reduction is achieved and decreases otherwise. Concretely, the trial step $\tilde{X}_{k+1} := X_k + S_k$ in TR, with $S_k = \arg\min_{s \in \mathcal{X}_k} M_k(s)$ as defined in (6) but with estimated quantities, is accepted if $A_k := \bar{F}_{N_k}(X_k) - \bar{F}_{N_k}(\tilde{X}_{k+1})$ is similar to the predicted reduction $B_k := M_k(0) - M_k(S_k)$. An important ingredient for both TR and LS is to ensure the stochasticity in function and gradients does not derail the search. SASS stands on a common assumption for the stochastic gradient $\bar{G}_{N_k}(X_k)$ (Bottou et al. 2018), namely, $\mathbb{E}[\|\bar{G}_{N_k}(X_k) - \nabla f(X_k)\|^2 \mid \mathcal{F}_k] \leq c_1 + c_2\|\nabla f(X_k)\|$ for some $c_1, c_2 > 0$. Two stochastic TR algorithms, STORM (Blanchet et al. 2019) and ASTRO, both enjoy

$$|f(X_k) - \bar{F}_{N_k}(X_k)| = \mathcal{O}(\Delta_k^2), \quad \text{and } \|\nabla f(X_k) - \bar{G}_{N_k}(X_k)\| = \mathcal{O}(\Delta_k) \quad \text{whp or wp1;}$$

proven to find an ϵ-optimal solution with simulation cost $\mathcal{O}(\epsilon^{-6}\log\epsilon^{-1})$. However, ASTRO allows for CRN which, under smooth F, dramatically reduces this cost to $\mathcal{O}(\epsilon^{-2}\log\epsilon^{-1})$. Cao et al. (2023) have considered (without sample complexity guarantees) replacing the typical sufficient reduction criterion with $A_k + r > \theta B_k$, where r is a *relaxation* incorporating uncertainty in the function estimate. Lastly, inspired by regularizing (6) in deterministic settings, recent endeavours have embarked in SO, e.g., adding a cubic regularizer $\frac{1}{3\Delta_k}\|s\|^3$ (Scheinberg and Xie 2023) that has challenges in cubic minimization with no sample complexity guarantees. Regularized quadratic approximation (Curtis and Wang 2023) is unexplored in SO.

4 DISCRETE SIMULATION OPTIMIZATION

Example 2: The Ambulance Placement Problem In a city, when a call request comes in, the nearest idle ambulance is dispatched to the scene, driving at a random speed, and on arrival, spending a stochastic amount of time for treating the patient (Nickel et al. 2016). Where should ambulance bases be placed to minimize the expected patient call response time? This is an example of a discrete optimization problem, since the space in which x is defined is within a finite set $\mathcal{X}$ by, e.g., coordinates of intersections in a grid.

When the feasible set is countable and small, exhaustive search with R&S dominates the literature. For larger feasible sets, faster solvers that explore the decision space are needed. Heuristics and evolutionary algorithms are popular in practice for this setting and can be economized with smart sampling schemes (in the noise space) (Shashaani and Vahdat 2023). However, discrete SO solvers are advantageous due to their convergence to global solutions (showing that if the algorithm is run long enough, the entire feasible space will be covered) and complexity guarantees. A typical way of dealing with a high-dimensional discrete problem $f : \mathcal{X} \subseteq \mathbb{Z}^d \mapsto \mathbb{R}$ is (i) to solve its linear interpolation $\tilde{f} : \mathbb{R}^d \mapsto \mathbb{R}$ (that can maintain the original problem's properties such as convexity) to find $\tilde{x}^* \in \mathbb{R}^d$, and (ii) find the best neighboring solution to $\tilde{x}^*$ in $\mathcal{X} \subseteq \mathbb{Z}^d$. In SO, this is quite expensive as solving the continuous envelope of the original problem is itself time consuming with noise, and enumerating the neighborhood is also expensive in high dimensions.

4.1 Methods Based on Partitioning

This group of algorithms searches for good neighborhoods before searching for good solutions. Branch and bound methods have been used for a long time with the philosophy that maybe we can save lots of computation by screening a whole subregion out, or rather mark a subregion as a *promising* one to focus our efforts on. The SO counterparts of this approach are three algorithms: i) COMPASS (Xu et al. 2010), ii) nested partitioning (Shi and Ólafsson 2009), and iii) stochastic branch and bound methods (Xu and

Nelson 2013). In some sense, these group of algorithms pass on the best neighborhoods they find from one iteration to another. These partitioning schemes are different from randomized sub-spacing schemes in continuous optimization that have been studied for dimension reduction (Cartis and Roberts 2023).

4.2 Methods Based on Pseudo-Gradients

In line with the intuitive Newton methods, there are attempts in discrete SO that leverage finding good directions via gradient-like steps. Zhang et al. (2023) propose to use sub-gradients $\tilde{G}_k$ of $\tilde{f}$—a convex linear interpolation of f, which is non-smooth. They use truncation to reduce the simulation cost and project the trial step to the feasible region via $\Pi_{\mathcal{X}}(X_k - \Delta_k \tilde{G}_k)$, attaining $\mathcal{O}(\log d)$ dependence on problem dimension (a drastic reduction from $\mathcal{O}(d^3)$) under convexity and R&S-based sampling rules. A similar idea, Retrospective Search using Piecewise-Linear Interpolation and Neighborhood Enumeration (R-SPLINE), was proposed by Wang et al. (2013) that computes a *pseudo-gradient* on the nearest non-integer point $\tilde{X}_k$ to X_k via piece-wise interpolation to apply LS and backtracking to. Once a good solution $\tilde{X}_{k+1}$ was found, the algorithm chooses the best nearest in the feasible space and in doing so uses RA for sampling.

4.3 Methods Based on Sampling (in the Decision Space)

These methods define sampling distributions on the decision space and update those distributions based on the incoming data from simulated points. One can also include the partitioning methods in this category when point selection in each partition is done following history-informed distributions. In some studies, this group of solvers are called *model-based* methods (Hu 2014) were the word model refers to the distribution that governs the randomized selection of new points in the decision space. In fact, one can view these algorithms as passing along probability distributions (with evolving supports) from one iteration to another rather than passing iterates. The early algorithms in this category were the pure adaptive random search (Zabinsky 2013) that sampled new X_k uniformly in the most recent level set of the function and replaced the incumbent accordingly. The hesitant adaptive search (Zabinsky and Linz 2023) distribution that aligns with the idea behind hit-and-run methods (Kiatsupaibul et al. 2011), aiming for the sampling distribution to converge to a uniform distribution for global solutions and zero elsewhere. Adaptive annealing procedures (Romeijn and Smith 1994; Alrefaei and Andradóttir 1999) sample X_k from the entire space but increase the sampling probability in the promising region following the Boltzman distribution.

5 CONCLUDING REMARKS AND OTHER ADVANCES

An aim and hope of this tutorial is to promote that besides great strides of the SO theory and algorithms exploited in statistical realms, exciting opportunities await. The cost of randomness in all classes of stochastic optimization algorithms provably dominates complexity. SO offers a new perspective: let's control that cost with sampling—a classical strength of simulation methods—in dialogue with mathematical programs!

We regretfully leave out other advances in SO due to space limit and refer the interested reader to excellent surveys in multiobjective SO (Hunter et al. 2019), parallelization (Hunter and Nelson 2017), Bayesian and metamodeling-based SO (Wu et al. 2018), (distributionally) robust methods and input uncertainty (Zhou and Xie 2015), and sequential SO (Luo et al. 2015). But we say a few words about constrained SO.

In (P), each constraint function c_i can be deterministic (e.g., a space capacity constraint for the newsvendor problem), or stochastic and only estimable using the outputs of a stochastic simulation (e.g., a constraint on the expected number of unsold items by the end of the day). When the constraints are deterministic, an SAA approach can adopt any of the existing constrained optimization solvers, such as with Lagrange multipliers and other penalty / dual methods, projection methods, or feasible direction methods including the conditional gradient (Frank-Wolfe) methods, manifold-based methods, or interior point methods to solve (P). In random search methods and those that sample points in the decision space, the feasibility can be directly enforced on the sampling distribution. Complexity analysis in these settings remains open. On the other hand, when the constraints are stochastic (including chance constraints), whether

X_k is feasible or not is uncertain. This is an active area of research; see review by Homem-de Mello and Bayraksan (2014b), feasibility metrics proposed by Eckman et al. (2023), and a *cooperative* SA method for a convex $\mathcal{X}$ that may move using estimated constraint gradients instead (Lan and Zhou 2020).

ACKNOWLEDGMENTS

This work was partially supported by grants CMMI-2226347 of the U.S. National Science Foundation and N000142412398 of the U.S. Office of Naval Research. I also thank Robert Smith for help with Fig. 1b.

REFERENCES

Alrefaei, M. H. and S. Andradóttir. 1999. “A Simulated Annealing Algorithm with Constant Temperature for Discrete Stochastic Optimization”. *Management Science* 45(5):748–764.

Andradóttir, S. 1995. “A Stochastic Approximation Algorithm with Varying Bounds”. *Operations Research* 43(6):1037–1048.

Andradóttir, S. and A. A. Prudius. 2009. “Balanced Explorative and Exploitative Search with Estimation for Simulation Optimization”. *INFORMS Journal on Computing* 21(2):193–208.

Arjevani, Y., Y. Carmon, J. C. Duchi, D. J. Foster, N. Srebro and B. Woodworth. 2023. “Lower Bounds for Non-convex Stochastic Optimization”. *Mathematical Programming* 199(1):165–214.

Asmussen, S. and P. W. Glynn. 2007. *Stochastic Simulation: Algorithms and Analysis*, Volume 57. Springer.

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

Blanchet, J., C. Cartis, M. Menickelly, and K. Scheinberg. 2019. “Convergence Rate Analysis of a Stochastic Trust-region Method via Supermartingales”. *INFORMS Journal on Optimization* 1(2):92–119.

Boesel, J., B. L. Nelson, and N. Ishii. 2003. “A Framework for Simulation-optimization Software”. *IIE Transactions* 35(3):221–229.

Bollapragada, R., R. Byrd, and J. Nocedal. 2018. “Adaptive Sampling Strategies for Stochastic Optimization”. *SIAM Journal on Optimization* 28(4):3312–3343.

Bottou, L., F. E. Curtis, and J. Nocedal. 2018. “Optimization Methods for Large-scale Machine Learning”. *SIAM Review* 60(2):223–311.

Byrd, R. H., G. M. Chin, J. Nocedal, and Y. Wu. 2012. “Sample Size Selection in Optimization Methods for Machine Learning”. *Mathematical Programming* 134(1):127–155.

Cao, L., A. S. Berahas, and K. Scheinberg. 2023. “First- and Second-order High Probability Complexity Bounds for Trust-region Methods with Noisy Oracles”. *Mathematical Programming* 199(1):1–52.

Cartis, C. and L. Roberts. 2023. “Scalable Subspace Methods for Derivative-free Nonlinear Least-squares Optimization”. *Mathematical Programming* 199(1):461–524.

Chau, M. and M. C. Fu. 2014. “An Overview of Stochastic Approximation”. *Handbook of Simulation Optimization*:149–178.

Chen, H. and B. W. Schmeiser. 2001. “Stochastic Root Finding via Retrospective Approximation”. *IIE Transactions* 33(3):259–275.

Chong, E. K. and P. J. Ramadge. 1993. “Optimization of Queues using an Infinitesimal Perturbation Analysis-based Stochastic Algorithm with General Update Times”. *SIAM Journal on Control and Optimization* 31(3):698–732.

Conn, A. R., K. Scheinberg, and L. N. Vicente. 2009. *Introduction to Derivative-free Optimization*. SIAM.

Curtis, F. E. and Q. Wang. 2023. “Worst-Case Complexity of TRACE with Inexact Subproblem Solutions for Nonconvex Smooth Optimization”. *SIAM Journal on Optimization* 33(3):2191–2221.

Eckman, D. J., S. G. Henderson, and S. Shashaani. 2023. “SimOpt: A Testbed for Simulation-Optimization Experiments”. *INFORMS Journal on Computing* 35(2):495–508.

Eckman, D. J., S. Shashaani, and S. G. Henderson. 2023. “Stochastic Constraints: How Feasible is Feasible?”. In *Proceedings of the 2023 Winter Simulation Conference*. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc. https://doi.org/10.1109/WSC60868.2023.10408734.

Ford, M. T., S. G. Henderson, and D. J. Eckman. 2022. “Automatic Differentiation for Gradient Estimators in Simulation”. In *Proceedings of the 2022 Winter Simulation Conference*, 3134–3145. Institute of Electrical and Electronics Engineers, Inc. https://doi.org/10.1109/WSC57314.2022.10015421.

Fowkes, J., L. Roberts, and Árpád Bűrmen. 2022. “PyCUTEst: An Open Source Python Package of Optimization Test Problems”. *Journal of Open Source Software* 7(78):4377.

Fu, M. C. 2002. “Optimization for Simulation: Theory vs. Practice”. *INFORMS Journal on Computing* 14(3):192–215.

Fu, M. C. 2015. “Stochastic Gradient Estimation”. *Handbook of Simulation Optimization*:105.

Fu, M. C. and S. G. Henderson. 2017. “History of Seeking Better Solutions, AKA Simulation Optimization”. In *Proceedings of the 2017 Winter Simulation Conference*, 131–157. Institute of Electrical and Electronics Engineers, Inc. https://doi.org/10.1109/WSC.2017.8247787.

Garmanjani, R. and L. N. Vicente. 2013. “Smoothing and Worst-case Complexity for Direct-search Methods in Nonsmooth Optimization”. *IMA Journal of Numerical Analysis* 33(3):1008–1028.

Ghosh, M. and N. Mukhopadhyay. 1979. “Sequential Point Estimation of the Mean when the Distribution is Unspecified”. *Communications in Statistics-Theory and Methods* 8(7):637–652.

Glynn, P. W. 1987. “Likelilood Ratio Gradient Estimation: An Overview”. In *Proceedings of the 1987 Winter Simulation Conference*, 366–375 https://doi.org/10.1145/318371.318612.

Gratton, S., C. W. Royer, L. N. Vicente, and Z. Zhang. 2015. “Direct Search based on Probabilistic Descent”. *SIAM Journal on Optimization* 25(3):1515–1541.

Ha, Y., S. Shashaani, and M. Menickelly. 2024. “Two-Stage Estimation and Variance Modeling for Latency-Constrained Variational Quantum Algorithms”. *arXiv preprint arXiv:2401.08912*. Just Accepted in INFORMS Journal on Computing.

Ha, Y., S. Shashaani, and R. Pasupathy. 2024. “Complexity of Zeroth-and First-order Stochastic Trust-Region Algorithms”. *arXiv preprint arXiv:2405.20116*. Just Accepted in SIAM Journal on Optimization.

Hansen, N., A. Auger, R. Ros, O. Mersmann, T. Tušar and D. Brockhoff. 2021. “COCO: A Platform for Comparing Continuous Optimizers in a Black-Box Setting”. *Optimization Methods and Software* 36:114–144.

Homem-de Mello, T. and G. Bayraksan. 2014a. “Monte Carlo Sampling-based Methods for Stochastic Optimization”. *Surveys in Operations Research and Management Science* 19(1):56–85.

Homem-de Mello, T. and G. Bayraksan. 2014b. “Stochastic Constraints and Variance Reduction Techniques”. In *Handbook of Simulation Optimization*, 245–276. Springer.

Hong, L. J. and B. L. Nelson. 2009. “A Brief Introduction to Optimization via Simulation”. In *Proceedings of the 2009 Winter Simulation Conference*, 75–85. Institute of Electrical and Electronics Engineers, Inc. https://doi.org/10.1109/WSC.2009.5429321.

Hu, J. 2014. “Model-based Stochastic Search Methods”. *Handbook of Simulation Optimization*:319–340.

Hunter, S. R., E. A. Applegate, V. Arora, B. Chong, K. Cooper, O. Rincón-Guevara *et al*. 2019. “An Introduction to Multiobjective Simulation Optimization”. *ACM Transactions on Modeling and Computer Simulation* 29(1):1–36.

Hunter, S. R. and B. L. Nelson. 2017. “Parallel Ranking and Selection”. In *Advances in Modeling and Simulation: Seminal Research from 50 Years of Winter Simulation Conferences*, 249–275. Springer.

Jain, P., S. Shashaani, and E. Byon. 2024. “Simulation Model Calibration with Dynamic Stratification and Adaptive Sampling”. *arXiv preprint arXiv:2401.14558*.

Jian, N. and S. G. Henderson. 2015. “An Introduction to Simulation Optimization”. In *Proceedings of the 2015 Winter Simulation Conference*, 1780–1794. Institute of Electrical and Electronics Engineers, Inc. https://doi.org/10.1109/WSC.2015.7408295.

Jin, B., K. Scheinberg, and M. Xie. 2024. “High Probability Complexity Bounds for Adaptive Step Search Based on Stochastic Oracles”. *SIAM Journal on Optimization* 34(3):2411–2439.

Kesten, H. 1958. “Accelerated Stochastic Approximation”. *The Annals of Mathematical Statistics*:41–59.

Kiatsupaibul, S., R. L. Smith, and Z. B. Zabinsky. 2011. “An Analysis of a Variation of Hit-and-run for Uniform Sampling from General Regions”. *ACM Transactions on Modeling and Computer Simulation* 21(3):1–11.

Kiefer, J. and J. Wolfowitz. 1952. “Stochastic Estimation of the Maximum of a Regression Function”. *The Annals of Mathematical Statistics*:462–466.

Kim, S.-H. and B. L. Nelson. 2006. “Selecting the Best System”. *Handbooks in Operations Research and Management Science* 13:501–534.

Kolda, T. G., R. M. Lewis, and V. Torczon. 2003. “Optimization by Direct Search: New Perspectives on Some Classical and Modern Methods”. *SIAM Review* 45(3):385–482.

Lan, G. and Z. Zhou. 2020. “Algorithms for Stochastic Optimization with Function or Expectation Constraints”. *Computational Optimization and Applications* 76(2):461–498.

Larson, J., M. Menickelly, and S. M. Wild. 2019. “Derivative-free Optimization Methods”. *Acta Numerica* 28:287–404.

Le Digabel, S. 2009. “NOMAD: Nonlinear Optimization with the MADS Algorithm”. *Rapport Technique G-2009-39, Les Cahiers du GERAD*:32.

Lee, L. H., C.-h. Chen, E. P. Chew, J. Li, N. A. Pujowidianto and S. Zhang. 2010. “A Review of Optimal Computing Budget Allocation Algorithms for Simulation Optimization Problem”. *International Journal of Operations Research* 7(2):19–31.

Liu, D. C. and J. Nocedal. 1989. “On the Limited Memory BFGS Method for Large Scale Optimization”. *Mathematical Programming* 45(1-3):503–528.

Luo, J., L. J. Hong, B. L. Nelson, and Y. Wu. 2015. “Fully Sequential Procedures for Large-scale Ranking-and-selection Problems in Parallel Computing Environments”. *Operations Research* 63(5):1177–1194.

Mak, W.-K., D. P. Morton, and R. K. Wood. 1999. “Monte Carlo Bounding Techniques for Determining Solution Quality in Stochastic Programs”. *Operations Research Letters* 24(1-2):47–56.

Nelder, J. A. and R. Mead. 1965. “A Simplex Method for Function Minimization”. *The Computer Journal* 7(4):308–313.

Nelson, B. L. 1990. “Control Variate Remedies”. *Operations Research* 38(6):974–992.

Nemirovski, A., A. Juditsky, G. Lan, and A. Shapiro. 2009. “Robust Stochastic Approximation Approach to Stochastic Programming”. *SIAM Journal on Optimization* 19(4):1574–1609.

Nickel, S., M. Reuter-Oppermann, and F. Saldanha-da Gama. 2016. “Ambulance Location under Stochastic Demand: A Sampling Approach”. *Operations Research for Health Care* 8:24–32.

Peng, Y., C.-H. Chen, and M. C. Fu. 2023. “Simulation Optimization in the New Era of AI”. In *Tutorials in Operations Research: Advancing the Frontiers of OR/MS: From Methodologies to Applications*, 82–108. INFORMS.

Polyak, B. T. and A. B. Juditsky. 1992. “Acceleration of Stochastic Approximation by Averaging”. *SIAM Journal on Control and Optimization* 30(4):838–855.

Rinaldi, F., L. Vicente, and D. Zeffiro. 2024. “Stochastic Trust-region and Direct-search Methods: A Weak Tail Bound Condition and Reduced Sample Sizing”. *SIAM Journal on Optimization* 34(2):2067–2092.

Rios, L. M. and N. V. Sahinidis. 2013. “Derivative-free Optimization: A Review of Algorithms and Comparison of Software Implementations”. *Journal of Global Optimization* 56(3):1247–1293.

Robbins, H. and S. Monro. 1951. “A Stochastic Approximation Method”. *The Annals of Mathematical Statistics*:400–407.

Romeijn, H. E. and R. L. Smith. 1994. “Simulated Annealing and Adaptive Search in Global Optimization”. *Probability in the Engineering and Informational Sciences* 8(4):571–590.

Rosenbaum, I. and J. Staum. 2015. “Database Monte Carlo for Simulation on Demand”. In *Proceedings of the 2015 Winter Simulation Conference*, 679–688. Institute of Electrical and Electronics Engineers, Inc. https://doi.org/10.1109/WSC.2015.7408206.

Scheinberg, K. and M. Xie. 2023. “Stochastic Adaptive Regularization Method with Cubics: A High Probability Complexity Bound”. In *Proceedings of the 2023 Winter Simulation Conference*, 3520–3531. Institute of Electrical and Electronics Engineers, Inc. https://doi.org/10.1109/WSC60868.2023.10408598.

Shapiro, A., D. Dentcheva, and A. Ruszczynski. 2021. *Lectures on Stochastic Programming: Modeling and Theory*. SIAM.

Shashaani, S., F. 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.

Shashaani, S. and K. Vahdat. 2023. “Improved Feature Selection with Simulation Optimization”. *Optimization and Engineering* 24(2):1183–1223.

Shi, L. and S. Ólafsson. 2009. *The Nested Partitions Method, Theory and Applications*. Springer.

Spall, J. 2005. *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. Wiley Series in Discrete Mathematics and Optimization. Wiley.

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

Wang, H., R. Pasupathy, and B. W. Schmeiser. 2013. “Integer-ordered Simulation Optimization using R-SPLINE: Retrospective Search with Piecewise-linear Interpolation and Neighborhood Enumeration”. *ACM Transactions on Modeling and Computer Simulation* 23(3):1–24.

Wu, D., H. Zhu, and E. Zhou. 2018. “A Bayesian Risk Approach to Data-driven Stochastic Optimization: Formulations and Asymptotics”. *SIAM Journal on Optimization* 28(2):1588–1612.

Xu, J., B. L. Nelson, and J. L. Hong. 2010. “Industrial Strength COMPASS: A Comprehensive Algorithm and Software for Optimization via Simulation”. *ACM Transactions on Modeling and Computer Simulation* 20(1):1–29.

Xu, W. L. and B. L. Nelson. 2013. “Empirical Stochastic Branch-and-bound for Optimization via Simulation”. *IIE Transactions* 45(7):685–698.

Zabinsky, Z. B. 2013. *Stochastic Adaptive Search for Global Optimization*, Volume 72. Springer Science & Business Media.

Zabinsky, Z. B. and D. D. Linz. 2023. “Hesitant Adaptive Search with Estimation and Quantile Adaptive Search for Global Optimization with Noise”. *Journal of Global Optimization* 87(1):31–55.

Zhang, H., Z. Zheng, and J. Lavaei. 2023. “Gradient-based Algorithms for Convex Discrete Optimization via Simulation”. *Operations Research* 71(5):1815–1834.

Zhang, S., J. Xu, L. H. Lee, E. P. Chew, W. P. Wong and C.-H. Chen. 2016. “Optimal Computing Budget Allocation for Particle Swarm Optimization in Stochastic Optimization”. *IEEE Transactions on Evolutionary Computation* 21(2):206–219.

Zhou, E. and W. Xie. 2015. “Simulation Optimization when Facing Input Uncertainty”. In *Proceedings of the 2015 Winter Simulation Conference*, 3714–3724. Institute of Electrical and Electronics Engineers, Inc. https://doi.org/10.1109/WSC.2015.7408529.

AUTHOR BIOGRAPHIES

SARA SHASHAANI is an Assistant Professor and the 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 robust data-driven modeling and analysis. She is a co-creator of the open-source SimOpt library at https://simopt.org. Her email address is sshasha2@ncsu.edu and her homepage is https://shashaani.wordpress.ncsu.edu/.