Stochastic Localization Methods for Convex Discrete Optimization via Simulation

Haixiang Zhang
Department of Mathematics, University of California, Berkeley, CA 94720, haixiang_zhang@berkeley.edu

Zeyu Zheng
Department of Industrial Engineering and Operations Research, University of California, Berkeley, CA 94720, zyzheng@berkeley.edu

Javad Lavaei
Department of Industrial Engineering and Operations Research, University of California, Berkeley, CA 94720, lavaei@berkeley.edu

We develop and analyze a set of new sequential simulation-optimization algorithms for large-scale multi-dimensional *discrete optimization via simulation problems* with a convexity structure. The "large-scale" notion refers to that the decision variable has a large number of values to choose from on each dimension. The proposed algorithms are targeted to identify a solution that is close to the optimal solution given any precision level with any given probability. To achieve this target, utilizing the convexity structure, our algorithm design does not need to scan all the choices of the decision variable, but instead sequentially draws a subset of choices of the decision variable and uses them to "localize" potentially near-optimal solutions to an adaptively shrinking region. To show the power of the localization operation, we first consider one-dimensional large-scale problems. We propose the shrinking uniform sampling algorithm, which is proved to achieve the target with an optimal expected simulation cost under an asymptotic criterion. For multi-dimensional problems, we combine the idea of localization with subgradient information and propose a framework to design stochastic cutting-plane methods, whose expected simulation costs have a low dependence on the scale and the dimension of the problems. In addition, utilizing the discrete nature of the problems, we propose a dimension reduction algorithm, which does not require prior information about the Lipschitz constant of the objective function and its simulation costs are upper bounded by a value that is independent of the Lipschitz constant. We implement the proposed algorithms on both synthetic and queueing simulation optimization problems, and demonstrate better performances compared to benchmark methods especially for large-scale examples.

Key words: Discrete optimization via simulation, convex optimization, shrinking uniform sampling algorithm, best achievable performance, stochastic cutting-plane methods, dimension reduction method

History: This paper was first submitted on January 19, 2022. The authors greatly appreciate the comments and suggestions from the anonymous reviewers and editors, which all have significantly benefited this work. Haixiang Zhang and Javad Lavaei were supported by grants from ARO, ONR, AFOSR and NSF. Haixiang Zhang was partially supported by the Two Sigma Ph.D. Fellowship. Zeyu Zheng was partially supported by the Hellman Fellows Fund and NSF.

1. Introduction

In the areas of operations research and management science, many decision-making problems involve complex stochastic systems and discrete decision variables. In presence of stochastic uncertainties, many replications of stochastic simulation are often needed to accurately evaluate the objective function associated with a discrete decision variable. Such problems are sometimes referred to as *Discrete Optimization via Simulation*, *Discrete Simulation Optimization*, or *Simulation/Stochastic Optimization with Integer Decision Variables* (see Nelson (2010), Hong et al. (2015), Ragavan et al. (2022)). For complex stochastic systems, even one replication of simulation can be time consuming or costly; see also Xu et al. (2010), Sun et al. (2014), Xu et al. (2016) for related discussions. When the decision space is large, it is often computationally impractical to run simulations for all choices of the decision variables, creating a challenge in finding the optimal or near-optimal choice of decision variables. To circumvent this challenge, problem structure such as convexity or local convexity of the objective function may need to be exploited to lower costs and improve the efficiency to find an optimal or near-optimal choice.

In this paper, we consider large-scale discrete optimization via simulation problems with a convex objective function. The notion of "large-scale" refers to a large number of choices for the discrete decision variable on each dimension. Optimization problems with such features naturally arise in many operations research and management science applications, including queueing networks, supply chain networks, sharing economy operations, financial markets, etc.; see Shaked and Shanthikumar (1988), Wolff and Wang (2002), Altman et al. (2003), Singhvi et al. (2015), Jian et al. (2016), Freund et al. (2017) for example. Particularly in the area of supply chain management, a significant amount of models are proved to be discrete convex: lost-sales inventory systems with positive lead time (Zipkin 2008); serial inventory systems (Huh and Janakiraman 2010); single-stage inventory systems with positive order lead time (Pang et al. 2012); capacitated inventory systems with remanufacturing (Gong and Chao 2013); more applications are discussed in Chen and Li (2020). Overall in these papers, the authors consider various decision-making settings and prove convexity for commonly used objective functions in the corresponding settings. In these applications, the convexity is proved, but finer structure such as strong convexity often does not hold or is very difficult to prove. In addition, there may be many choices of decision variables whose associated objective values are close to the optimal objective value, and the gap between optimal and sub-optimal solutions is hard to measure or estimate a priori. For the algorithms designed

in this work, we take the view that this gap information is not available and the algorithms are designed to work for arbitrarily small unknown gap.

In this work, we develop provably efficient simulation-optimization algorithms that are guaranteed with an arbitrary probability $1-\delta$ to find a near-optimal choice of the decision variable that renders an objective value ϵ-close to the optimal solution, where ϵ is an arbitrary user-specified precision level. This criterion is called (ϵ,δ)-*Probability of Good Selection* ((ϵ,δ)-PGS) in the simulation literature; see Ma and Henderson (2017) and Hong et al. (2020). Although the asymptotic regime $\delta \ll 1$ is of more interest in many theoretical works, this work provides bounds on the simulation cost that hold for all $\epsilon \geq 0$ and $\delta \in (0,1]$. To quantify the computational cost for the proposed algorithms that are guaranteed to find ϵ-optimal solutions with high probability, we take the view that the simulation cost is the dominant contributor to the computational cost; see also Ma and Henderson (2019). The simulation cost of an algorithm is measured as the total number of simulation replications run at all possible decisions visited by the algorithm until it stops. When designing algorithms to solve large-scale discrete optimization via simulation problems, the dependence of the simulation cost on the problem size (or, the number of alternatives/solutions/systems in the area of ranking and selection) is crucial to understand; see also discussions in Zhong and Hong (2019).

Three most recent papers Wang et al. (2021), Eckman et al. (2020) and Zhang et al. (2020) also discussed the use of the convexity structure in simulation. Wang et al. (2021) considered a discrete simulation optimization problem with a specific polynomial functional form for the objective function, and focus on how to strategically use gradient information to accelerate the selection of the best. However, they focused on the fixed-budget problem with an approximately quadratic objective function and a one-dimensional decision space, which is different from our problem setting. Eckman et al. (2020) utilized the convexity structure to select a feasible region that contains the optimal given existing simulation samples at different choices; see also Eckman et al. (2021). Because they do not consider an optimization problem and their goal is not to find an optimal or near-optimal solution, the focus of Eckman et al. (2020) is different from ours. For example, they do not provide simulation-optimization algorithms that can find an optimal or near-optimal decision, nor do they analyze simulation costs and their dependence on problem scale. On the other hand, the method and analysis provided by Eckman et al. (2020) and Eckman et al. (2021) can serve effectively as a module to help solve other general simulation problems, such as multi-objective simulation optimization, which is not the focus of our work.

Zhang et al. (2020) proposed subgradient descent algorithms for problems with a high-dimension decision space. Roughly speaking, their algorithms scale well to high-dimensional problems, but are computationally expensive for large-scale problems. However, in practice, many problem settings

have a large scale but a relatively low dimension or even a single dimension; see the examples in the first paragraph of Section 3. In our work, the focus is on designing algorithms that work well for those large-scale problems. Furthermore, the subgradient descent algorithms in Zhang et al. (2020) require prior knowledge about the upper bounds on the Lipschitz constant L and the variance σ^2, and the simulation cost of the subgradient descent algorithm has a polynomial dependence on these upper bounds; see the comparison of results in Table 3. For many real-world discrete simulation via optimization problems, the Lipschitz constant and the variance are unknown and hard to estimate. As a result, both upper bounds are likely to be over-estimated, which will lead to worse simulation costs. In this work, algorithms that do not rely on prior information about L and σ^2 are proposed, which resolve the aforementioned issues. Moreover, the algorithms proposed in this work have a logarithmic dependence or no dependence on the upper bound L. Intuitively, (discrete) convex functions grow at a super-linear rate when the input goes to infinity, i.e., $\lim_{\|x\|\to\infty} |f(x)|/\|x\| = \infty$. In this case, the Lipschitz constant will be large in the regions where $\|x\|$ is large. Therefore, the upper bound on the Lipschitz constant L will be large for large-scale optimization via simulation problems and reducing the dependence on the Lipschitz constant is important. Another idea is to adaptively adjust the stepsize (or parameters that play a similar role), which is common for stochastic optimization for machine learning problems, e.g., ADAM, AdaGrad and RMSProp algorithms. However, the convergence of those algorithms is established for smooth objective functions. In our case, the Lovász extension is a non-smooth function, which prohibits the application of most adaptive methods. To the best of our knowledge, the only techniques in literature that considered a similar setting are the R-SPLINE (Wang et al. 2013) the ADALINE algorithms (Ragavan et al. 2022), which only provided an asymptotic convergence result.

1.1. Contributions

The major methodology in algorithm design in this paper can be classified as *stochastic localization methods*, in the sense that we "localize" potentially near-optimal solutions in a subset and adaptively shrink the subset (denoted as $\mathcal{S}$ in our proposed algorithms) at each step. At iteration k, the stochastic localization algorithms guarantee that the set $\mathcal{S}_k$ contains an ϵ-optimal solution with high probability. At the beginning of the algorithm ($k = 0$), the set $\mathcal{S}_0$ is equal to the feasible set $\mathcal{X}$, which can be viewed as a "global" neighbourhood of the optimal solution x^*. After several iterations, the size of $\mathcal{S}_k$ is reduced by a lot and can be viewed as a "local" neighbourhood of x^*. We describe this process as the *localization* of an approximately optimal solution. The design of algorithms relies on and addresses the challenge from the fact that the feasible set is a discrete set. Intuitively, if the feasible set has a finite number of discrete points, the subset of

potentially near-optimal solutions can only be shrunk for a finite number of times, and the number of localization operations cannot exceed the size of the feasible set. The proposed algorithms generally do not rely on prior estimates of the Lipschitz constant and the variance. In addition, the simulation cost of achieving the PGS guarantee does not depend on the Lipschitz constant. We note that the expected simulation cost has an inevitable dependence on the variance σ^2. To avoid requiring prior knowledge about the variance in the Gaussian case, after designing algorithms that do not require information about the Lipschitz constant, we propose in the appendix an adaptive scheme to address the challenge of unknown variances. The idea of localization also appears in prior literature of discrete optimization (Günlük 1999) or more specifically discrete optimization via simulation, such as empirical stochastic branch-and-bound (Xu and Nelson 2013), nested partition (Shi et al. 2000) and COMPASS (Hong and Nelson 2006, Xu et al. 2010). The line search procedure in R-SPLINE (Wang et al. 2013) and ADALINE (Ragavan et al. 2022) is able to capture the local convexity of the objective function. However, existing works do not utilize the global information implied by the convexity structure and do not provide complexity analysis of the proposed algorithms. In contrast, we propose specially-designed algorithms for discrete convex objective functions and provide an estimate of the simulation costs; see our comparisons to the Industrial-strength COMPASS algorithm in EC.8.

To show the usefulness of the localization operation, we first consider an important case of discrete simulation via optimization problems, where the decision space is the "one-dimensional" set $\{1, 2, \ldots, N\}$. Here, N is an arbitrary positive integer that represents the problem scale. Without the convexity structure, the problem setting is mathematically equivalent to the problem of *ranking and selection*; see Hong et al. (2020) for a comprehensive review. In this work, the objective function is assumed to be discrete convex on the decision space, but no other structure information such as strong convexity or the knowledge of a minimal gap between the optimal and sub-optimal solutions is known. Utilizing the idea of *localization*, we overcome the shortcoming of the subgradient descent algorithm that its simulation cost has a quadratic dependence on the problem scale. We propose two localization algorithms. As a natural generalization of the classical bi-section algorithm, we design the tri-section sampling (TS) algorithm to find an (ϵ, δ)-PGS solution. We prove that, when δ is small, $O(\log(N)\epsilon^{-2}\log(1/\delta))$ serves as an upper bound on the simulation cost for the TS algorithm for any one-dimensional convex problem, which represents the same logarithmic dependence on the scale as the bi-section algorithm. Note that when the convexity structure is not exploited, the optimal dependence on N can be linear. We then design the shrinking uniform sampling (SUS) algorithm that beats the TS algorithm. The SUS algorithm is proved to enjoy the upper bound on the simulation cost as $O[\epsilon^{-2}(\log(N) + \log(1/\delta))]$ when δ is small. Using the asymptotic criterion in Kaufmann et al. (2016), namely $\delta \to 0$ with other parameters fixed, the SUS

algorithm asymptotically achieves the optimal performance and, therefore, is the first algorithm to achieve a matching upper bound on simulation costs for ranking and selection problems with general convex structure. This theoretical superiority of the SUS algorithm is also verified in numerical experiments. We remark that our major contribution is the SUS algorithm rather than the TS algorithm, though the analysis provided for these two algorithms may be separately useful in broader settings.

Next, we turn to the settings of large-scale multi-dimensional problems with the "d-dimensional" discrete decision space $\{1,2,\ldots,N\}\times\{1,2,\ldots,N\}\times\ldots\times\{1,2,\ldots,N\}$. We note that the scale N can easily be relaxed to be different in each dimension in our algorithm design (e.g., after linear constraints are applied on the decision space), but we unify the use of N in each dimension in the analysis, so as to clearly demonstrate the impact of the scale N. In a multi-dimensional decision space, a common definition of discrete convexity, which guarantees that a local optimum is globally optimal, is the $L^\natural$-convexity (Murota 2003); see Dyer and Proll (1977), Freund et al. (2017) for examples of $L^\natural$-convex functions. We observe that even though the TS algorithm and the SUS algorithm designed for one-dimensional problems can be extended to the multi-dimensional case, the dependence of their simulation cost on the dimension d can be large, even up to an exponential order of dependence, which may prohibit their practical use in high-dimensional problems. This motivates us to consider alternative approaches to design stochastic localization algorithms that have a low dependence on the dimension d.

In this work, we combine the idea of localization with the *subgradient information* in the multi-dimensional case. The subgradient information is constructed by taking simulation samples and plays a crucial role in reducing the dependence of simulation cost on the dimension d. The cutting-plane methods (Vaidya 1996, Bertsimas and Vempala 2004, Lee et al. 2015, Jiang et al. 2020) are based on a similar idea and are known to have a lower order or no dependence on the Lipschitz constant. However, the cutting-plane methods are not robust to noise. Therefore, we develop a novel framework to design stochastic cutting-plane (SCP) algorithms based on deterministic cutting-plane algorithms, with the goal of achieving the PGS guarantee. A novel stochastic separation oracle is designed and analyzed. A straightforward application of the proposed framework leads to SCP algorithms that have an $O(d^3)$ dependence on the dimension and a logarithmic dependence on L, which improves the quadratic dependence of the subgradient-based algorithms in Zhang et al. (2020).

Utilizing the discrete nature of the problem, we further develop the dimension reduction algorithm whose simulation cost is upper bounded by a constant that is independent of Lipschitz constant L and has an $O(d^4)$ dependence on the dimension. This is the first algorithm for discrete optimization via simulation that utilizes the convex structure of the objective to reduce the

simulation cost and does not require knowledge about the Lipschitz constant L. In contrast, the subgradient-based search algorithms developed in Zhang et al. (2020) has a higher order dependence on L and requires the knowledge about the Lipschitz constant, although it has a lower dependence ($O(d^2)$) on the dimension compared to the dimension reduction algorithm. Our developed SCP algorithms may particularly be preferable when the Lipschitz parameter L for a given problem is large or hard to estimate. When a prior estimate of the Lipschitz constant is unavailable, we need to estimate the Lipschitz constant through the stochastic oracle and this leads to two major difficulties. First, the objective function can only be evaluated with noise. This means that we need to simulate $F(x, \xi_x)$ a number of times to get a considerably accurate estimate of the Lipschitz constant and this process can be time-consuming. Second, we need to check a large number of points to estimate the Lipschitz constant. Even in the local neighbourhood $\{y \in \mathbb{Z}^d \mid \|y - x\|_\infty \leq 1\}$, we need to evaluate at least $O(2^d)$ points to get an estimate of the Lipschitz constant. The simulation cost will be prohibitively large even when d is as low as 50. The idea of gradually reducing the problem dimension was proposed in parallel in Jiang (2021), where the author made the algorithm more practical by reducing the number of arithmetic operations to be polynomial. We numerically verify that the dimension reduction algorithm has a better performance than the subgradient descent algorithm in Zhang et al. (2020) both on the synthetic and the queueing simulation optimization examples, especially for the large-scale case.

In terms of dependence on the scale N, we theoretically show that the subgradient descent algorithm and the SCP algorithms all present an $O(N^2)$ dependence on N for their simulation costs. However, the SCP algorithms empirically perform better than the subgradient descent algorithm in applications where N is large. On the other hand, the SUS algorithm, when extended to multi-dimensional problems, still present no dependence on N under the asymptotic criterion (Kaufmann et al. 2016), but however incurs an exponential dependence on d. These analyses can assist practitioners to choose which algorithm to use depending on the knowledge or partial knowledge on d, N and L in the specific problems.

Finally, we propose a novel algorithm that is able to adaptively estimate the variance of the randomness at each feasible decision in the case when the noise is Gaussian. Adaptive variants are highly important since the variance is not known in many real-life applications. The design of the algorithm is based on the property that the lower tail for χ^2-random variables is sub-Gaussian (Wainwright 2019). The adaptive algorithm is suitable for the case when an upper bound on the variance is hard to estimate and over-estimation is inevitable. In addition, the adaptive algorithm provides an approach to improve the simulation cost in the case when location-dependent upper bounds of the variance σ_x^2 is available for all feasible decisions x. This is because the uniform upper bound $\sigma^2 = \max_{x \in \mathcal{X}} \sigma_x^2$ is in general attained by extreme choices of the decision variable and

may be much larger than the variance of a large proportion of feasible decisions. In contrast to common two-stage procedures for the unknown variance case in ranking and selection literature, the proposed adaptive algorithm does not require simulating all choices of the decision variable (which requires $O(N^d)$ simulations) to get an upper bound on the variance. Moreover, using the novel algorithm, the simulation cost is at most increased by a constant factor compared to the known variance case.

The remainder of the paper is outlined as follows. Section 1.2 summarizes the notation. Section 2 introduces the model, framework, optimality criterion, and simulation costs. Section 3 discusses the algorithms and performance analysis developed for one-dimensional large-scale problems. Section 4 discusses the algorithms and performance analysis developed for multi-dimensional large-scale problems. Section 5 provides numerical experiments to compare the proposed algorithms to benchmark methods. Section 6 gives the concluding remarks. The adaptive algorithm for estimating the variance is provided in the appendix.

1.2. Notation

For a stochastic system labeled by its decision variable x, we denote ξ_x as the random object associated with the decision variable. We write $\xi_{x,1}, \xi_{x,2}, \ldots, \xi_{x,n}$ as independent and identically distributed (i.i.d.) copies of ξ_x. The empirical mean of the n independent evaluations for a decision variable labeled by x is denoted as $\hat{F}_n(x) := \frac{1}{n}\sum_{j=1}^{n} F(x, \xi_{x,j})$. The indices set $[N] := \{1, 2, \ldots, N\}$ is defined for every positive integer N. For any set S and positive integer d, we define the product set S^d as $\{(x_1, x_2 \ldots, x_d) : x_i \in S, i \in [d]\}$. For two vectors $x, y \in \mathbb{R}^d$, the maximum operation $x \vee y$, minimum operation $x \wedge y$, the ceiling function $\lceil x \rceil$ and the flooring function $\lfloor x \rfloor$ are all considered as component-wise operations. Let c be the indifference zone parameter in the PCS-IZ criterion. To compare simulation costs, we omit terms that are independent of $d, N, \epsilon, \delta, c$ in $O(\cdot)$ and omit terms independent of δ in $\tilde{O}(\cdot)$. To be more concrete, the notation $f = O(g)$ means that there exist constants $c_1, c_2 > 0$ independent of $N, d, \epsilon, \delta, c$ such that $f \le c_1 g + c_2$. Similarly, the notation $f = \tilde{O}(g)$ means that there exist constants $c_1 > 0$ independent of $N, d, \epsilon, \delta, c$ and constant $c_2 > 0$ independent of δ such that $f \le c_1 g + c_2$. The notation $f = \Theta(g)$ means that there exist constants $c_1, c_2, c_3 > 0$ independent of $N, d, \epsilon, \delta, c$ such that $c_3 g \le f \le c_1 g + c_2$. The notation $f = \tilde{\Theta}(g)$ means that there exist constants $c_1, c_3 > 0$ independent of $N, d, \epsilon, \delta, c$ and constants $c_2, c_4 > 0$ independent of δ such that $c_3 g + c_4 \le f \le c_1 g + c_2$.

2. Model and Framework

We consider a complex stochastic system that involves discrete decision variables in a d-dimensional subspace $\mathcal{X} = [N_1] \times [N_2] \times \cdots \times [N_d]$ in which the N_i's are positive integers. The objective function $f(x)$ for $x \in \mathcal{X}$ is given by

$$f(x) := \mathbb{E}[F(x, \xi_x)],$$

in which ξ_x is a random object belonging to the probability space $(\mathsf{Y}, \mathcal{B}_\mathsf{Y})$ and $F : \mathcal{X} \times \mathsf{Y} \to \mathbb{R}$ is a measurable function. Specifically, the function F captures the full operations logic in the stochastic system and measures the performance of the system. For example, in a queueing system, ξ_x is the arrival times and the service times of customers, and $F(\cdot, \xi_x)$ is the average waiting time of all customers under the situation described by ξ_x. We consider scenarios when the objective function $f(x)$ is not in closed-form and needs to be evaluated by averaging over simulation replications of $F(x, \xi_x)$. The random objects ξ_x's can be different for different choices of decision variables. In this work, we focus on identifying the optimal decision, i.e., finding the decision that has the minimal objective value:

$$\min_{x \in \mathcal{X}} \ f(x). \tag{1}$$

We assume that the objective function has a convex structure.

ASSUMPTION 1. *The objective function $f(x)$ is a convex function on the discrete set $\mathcal{X}$.*

For the exact definition of discrete convexity, we describe in detail in Section 3 for the one-dimensional cases and Section 4 for the multi-dimensional cases.

2.1. Optimality Guarantees and Classes of Algorithms

Our general goal is to design algorithms that guarantee the selection of a good decision that yields a close-to-optimal performance with high probability. Formally, this criterion is defined as *Probability of Good Selection*.

- (ϵ, δ)**-Probability of good selection (PGS).** The solution x returned by an algorithm has an objective value at most ϵ larger than the optimal objective value with probability at least $1 - \delta$.

This PGS guarantee is also referred to as the probably approximately correct selection (PAC) guarantee in the literature (Even-Dar et al. 2002, Kaufmann et al. 2016, Ma and Henderson 2017). While our main focus is to design algorithms that satisfy the PGS optimality guarantee, we also consider the optimality guarantee of *Probability of Correct Selection with Indifference Zone* for completeness.

- **Probability of correct selection with indifference zone (PCS-IZ).** (See Hong et al. (2020)) The problem is assumed to have a unique solution that renders the optimal objective value. The optimal objective value is assumed to be at least $c > 0$ smaller than the objective values at sub-optimal choices of decisions. The gap width c is called the **indifference zone parameter** in Bechhofer (1954). The PCS-IZ guarantee requires that the solution returned by an algorithm be the optimal solution with probability at least $1 - \delta$.

In general, by choosing $\epsilon < c$, algorithms satisfying the PGS guarantee can be readily applied to satisfy the PCS-IZ guarantee. However, algorithms satisfying the PCS-IZ guarantee may fail to satisfy the PGS guarantee; see Eckman and Henderson (2018) and Hong et al. (2020). The failing probability δ in either PGS or PCS-IZ is usually chosen to be small to ensure a high probability result. Hence, we assume henceforth that δ is small enough and focus on the asymptotic expected simulation cost. In addition, we assume that the probability distribution for the stochastic simulation output $F(x, \xi_x)$ is sub-Gaussian.

ASSUMPTION 2. *The distribution of $F(x, \xi_x) - f(x)$ is zero-mean sub-Gaussian with the known upper bound σ^2 on the parameter for any $x \in \mathcal{X}$.*

We note that a special case of Assumption 2 is when the distribution follows the Gaussian distribution. In that case, the parameter σ^2 can be chosen as the upper bound on the variance of the distribution. For more general distributions with a finite variance, the mean estimator in Lee and Valiant (2020) can be used in place of the empirical mean estimator and the results in this work can be directly generalized. We assume that Assumption 2 holds in the remainder of the paper. In EC.7, we consider the Gaussian noise case when the variance is unknown. We propose a novel algorithm to adaptively estimate the variance σ^2 in the Gaussian case. The triad of the decision space $\mathcal{X}$, the space of randomness $(\mathsf{Y}, \mathcal{B}_\mathsf{Y})$ and the function $F(\cdot, \cdot)$ is called the **model** of problem (1). We define the set of all models for which function $f(\cdot)$ is convex on set $\mathcal{X}$ as $\mathcal{MC}(\mathcal{X})$, or simply $\mathcal{MC}$. Next, we define the class of simulation-optimization algorithms that are proved to find solutions satisfying certain optimality guarantee for a given set of models.

DEFINITION 1. Given an optimality guarantee $\mathcal{O}$ and a set of models $\mathcal{M}$, a simulation-optimization algorithm is called an $(\mathcal{O}, \mathcal{M})$-algorithm if, for any model $\mathbb{M} \in \mathcal{M}$, the algorithm returns a solution to $\mathbb{M}$ that satisfies the optimality guarantee $\mathcal{O}$.

For example, the class of $(\text{PGS}, \mathcal{MC})$-algorithms guarantees a high-precision solution with high probability for any convex model.

2.2. Simulation Costs

For optimization via simulation problems, the view that the simulation cost of generating replications of $F(x, \xi_x)$ is the dominant contributor to the computational cost is widely held; see Luo et al. (2015), Ni et al. (2017), Ma and Henderson (2017, 2019). Therefore, for the purpose of comparing different simulation-optimization algorithms that satisfy certain optimality guarantee, the performance of each algorithm is measured by the total number of evaluations of $F(x, \xi_x)$ at different points x. The number of evaluations during an optimization process is called the *simulation*

cost. Besides providing a measure to compare different algorithms, simulation costs can provide insights into how the computational cost depends on the scale and dimension of the problem. Moreover, understanding the simulation costs can provide information to facilitate the setup of parallel procedures for large-scale problems. The main focus of this paper is to develop provably efficient simulation-optimization algorithms for a certain optimality guarantee and provide an upper bound on the simulation cost to achieve that guarantee. We note that our proposed algorithms do not require additional structures of the selection problem in addition to convexity. Now, we give the rigorous definition of the expected simulation cost for a given set of models $\mathcal{M}$ and given optimality guarantee $\mathcal{O}$.

DEFINITION 2. Given the optimality guarantee $\mathcal{O}$ and a set of models $\mathcal{M}$, the **expected simulation cost** is defined as

$$T(\mathcal{O},\mathcal{M}) := \inf_{A \text{ is } (\mathcal{O},\mathcal{M})} \sup_{\mathbb{M}\in\mathcal{M}} \mathbb{E}\left[\tau_A\right],$$

where τ_A is a random variable that represents the number of simulation evaluations of $F(\cdot,\cdot)$ for each implementation of Algorithm A. Intuitively, the process that Algorithm A finds a potentially ϵ-optimal solution includes a series of evaluations of $F(\cdot,\cdot)$ and decisions on which choices of decision variables to simulate, which can be viewed as a random process. An implementation of Algorithm A is basically an instance of this random process.

The notion of simulation cost in this paper is largely focused on

$$T(\epsilon,\delta,\mathcal{MC}) := T((\epsilon,\delta)\text{-}PGS,\mathcal{MC}).$$

We mention that the upper bounds derived in this paper also hold almost surely, while the lower bounds only hold in expectation.

To better present the dependence of the expected simulation cost on the scale and dimension of the problem, we assume that $N_1 = N_2 = \cdots = N_d$.

ASSUMPTION 3. *The feasible set of decision variables is $\mathcal{X} = [N]^d$, where $N \geq 2$ and $d \geq 1$.*

With Assumption 3 in hand, we will present the dependence of the expected simulation cost on N and d. We note that the results in this work can be naturally extended to the case when each dimension has a different number of feasible choices of decision variables. Furthermore, if the objective function f is defined on a $L^\natural$-convex set (i.e., the indicator function of the set is a $L^\natural$-convex function, which we will define later), the algorithms proposed in this paper can be directly extended with small modifications. A typical example of a $L^\natural$-convex set is the capacity-constrained set

$$\left\{(x_1,\ldots,x_d) \;\middle|\; x_i \in [N],\ \forall i \in [d],\ \textstyle\sum_i x_i \leq C\right\}$$

under a linear transform, where $C > 0$ is the capacity constraint; see Section 5 for more details.

3. Simulation-optimization Algorithms and Complexity Analysis: One-dimensional Case

We first consider a special class of optimization via simulation problems where the dimension of the decision variable is one, but there are a large number of choices of decision variable. This class of one-dimensional problems, despite of the less generality compared to multi-dimensional large-scale problems, have applications when the one-dimensional decision variable is a choice of overall resource level. For example, large delivery companies often need to decide the total number of trucks that should be recruited for operations in a self-contained region. A service system may need to decide the total number of staff members needed to host a special event. Such decisions often involve a trade-off between service satisfaction and resource costs. The convexity in the objective function often comes from the marginal decay of contribution to service satisfaction as the resource level increases; see the optimal allocation example and Figure 2 in Section 5 for more details.

In the one-dimensional case, the feasible set is $\mathcal{X} = [N] = \{1, 2, \ldots, N\}$. The discrete convexity for a function f can be defined similarly to the ordinary continuous convexity through the discrete midpoint convexity property, namely,

$$f(x+1) + f(x-1) \geq 2f(x), \quad \forall x \in \{2, \ldots, N-1\}.$$

If the function $f(x)$ is convex on $\mathcal{X}$, it has a convex linear interpolation on the continuous interval $[1, N]$, defined as

$$\tilde{f}(x) := [f(x_0+1) - f(x_0)] \cdot (x - x_0) + f(x_0), \quad \forall x \in [x_0, x_0+1],\ x_0 \in [N-1]. \tag{2}$$

The definition of discrete convexity in a multi-dimensional decision space is called the $L^\natural$-convexity (Murota 2003). We defer the discussion of $L^\natural$-convex functions for the multi-dimensional case to Section 4.

In this section, we propose simulation-optimization algorithms that are guaranteed to find solutions that satisfy the PGS guarantee, provided that the objective function has a convex structure. For every developed simulation-optimization algorithm, we provide an upper bound on the expected simulation cost to achieve the PGS guarantee. We also provide a lower bound on the expected simulation cost that reflects the best achievable performance for any algorithm. Under the asymptotic criterion in Kaufmann et al. (2016), one of our proposed algorithms can attain the best achievable asymptotic performance.

In contrast to the multi-dimensional case, where the subgradient descent algorithm achieves satisfying performance (Zhang et al. 2020), the subgradient descent algorithm is not efficient for large-scale one-dimensional problems. This is because of the $O(N^2)$ dependence in the simulation

cost. In addition, the subgradient descent algorithm relies on the Lipschitz constant of the objective function, which is shown to be unnecessary for discrete problems in this section. Utilizing the localization operation, the algorithms proposed in this section do not have the aforementioned issues. Therefore, the algorithms in this section provide better alternatives to the subgradient descent algorithm for one-dimensional problems. The analysis of the one-dimensional case also shows the limitation of subgradient-based search methods and provides a hint on how to improve algorithms for multi-dimensional problems.

3.1. Tri-section Sampling Algorithm and Upper Bound on Expected Simulation Cost

We first propose the tri-section sampling (TS) algorithm for the PGS guarantee. The idea of the TS algorithm is from the classical bi-section method and the golden section method. A similar TS algorithm is proposed in Agarwal et al. (2011) for stochastic continuous convex optimization, which controls the regret instead of the objective value. However, their algorithm does not utilize the prior information that the optimal solution is an integral point and thus the simulation cost has a polynomial dependence on the Lipschitz constant. In addition, although an algorithm that minimizes the regret can be used to minimize the objective function value, the resulting simulation cost may be larger than that of specialized optimization algorithms and has an inferior dependence on the dimension d in the multi-dimensional case. The pseudo-code of the proposed TS algorithm is listed in Algorithm 1. The 3-quantiles of an interval $[L, U]$ are $(2L+U)/3$ and $(L+2U)/3$. Since we are looking for integral solutions, we round these quantiles to integers. In the procedure of Algorithm 1, one step is to compute confidence intervals that satisfy certain confidence guarantees. We now provide one feasible approach to construct such confidence intervals, which is based on Hoeffding's inequality for sub-Gaussian random variables. Define

$$h(n, \sigma, \alpha) := \sqrt{\frac{2\sigma^2}{n} \log\left(\frac{2}{\alpha}\right)}.$$

Recall that σ^2 is the upper bound on the sub-Gaussian parameters of all choices of decision variables. With this function $h(\cdot)$ in hand, whenever n independent simulations of the decision x are available, one can construct a $(1-\alpha)$-confidence interval for $f(x)$ as

$$\left[\hat{F}_n(x) - h(n, \sigma, \alpha), \hat{F}_n(x) + h(n, \sigma, \alpha)\right].$$

This is because the distribution of the empirical mean $\hat{F}_n(x)$ is sub-Gaussian with parameter σ^2/n and Hoeffding's inequality gives

$$\mathbb{P}\left[|\hat{F}_n(x) - f(x)| > h(n, \sigma, \alpha)\right] \leq 2\exp\left(-\frac{n h(n, \sigma, \alpha)^2}{2\sigma^2}\right) = \alpha.$$

If the variance σ_x^2 of a single choice of decision variable x is known, the confidence interval may be sharpened by replacing σ with σ_x; see Section EC.7. We note that the analysis in this work can be generalized to more general distributions, such as the sub-exponential distributions, by replacing $h(n,\sigma,\alpha)$ with other concentration bounds.

Algorithm 1 Tri-section sampling algorithm for the PGS guarantee

Input: Model $\mathcal{X}=[N],(\mathsf{Y},\mathcal{B}_{\mathsf{Y}}),F(x,\xi_x)$, optimality guarantee parameters ϵ,δ.

Output: An (ϵ,δ)-PGS solution x^* to problem (1).

Set upper and lower bounds of the current interval $x_L \leftarrow 1, x_U \leftarrow N$.
Set maximal number of comparisons $T_{max} \leftarrow \log_{1.5}(N)+2$.
while $x_U - x_L > 2$ **do** ▷ Iterate until there are at most 3 points.
    Compute 3-quantiles of the interval $q_{1/3} \leftarrow \lfloor 2x_L/3 + x_U/3 \rfloor$ and $q_{2/3} \leftarrow \lceil x_L/3 + 2x_U/3 \rceil$.
    Simulate n independent copies of $F(q_{1/3},\xi_{1/3})$ and $F(q_{2/3},\xi_{2/3})$, where n is the smallest integer such that $h[n,\sigma,1-\delta/(2T_{max})] \le \epsilon/8$.
    Compute the empirical means $\hat{F}_n(q_{1/3}), \hat{F}_n(q_{2/3})$.
    if $\hat{F}_n(q_{1/3}) - \epsilon/8 \ge \hat{F}_n(q_{2/3}) + \epsilon/8$ **then**
        Update $x_L \leftarrow q_{1/3}$.
    else if $\hat{F}_n(q_{1/3}) + \epsilon/8 \le \hat{F}_n(q_{2/3}) - \epsilon/8$ **then**
        Update $x_U \leftarrow q_{2/3}$.
    else
        Update $x_L \leftarrow q_{1/3}$ and $x_U \leftarrow q_{2/3}$.
    end if
end while
Simulate $\tilde{n}$ independent copies of $F(x,\xi_x)$ for each $x \in \{x_L,\dots,x_U\}$, where $\tilde{n}$ is the smallest integer such that $h[\tilde{n},\sigma,1-\delta/(2T_{max})] \le \epsilon/2$. ▷ Now $x_U - x_L \le 2$.
Return the point in $\{x_L,\dots,x_U\}$ with the minimal empirical mean.

Intuitively, the algorithm iteratively shrinks the size of the set containing a potentially near-optimal choice of decision variables. We provide an example of the TS algorithm in Figure 1. In this example, we suppose that the current set is [10] and then, the two 3-quantiles are 4 and 7. Without loss of generality, we assume that $\epsilon_0 := f(7) - f(4) \ge 0$ and the global minimum is in the left set [4]. We consider two different cases. First, if we know that $\epsilon_0 > 0$ holds with high probability, no solution in $\{7,\dots,10\}$ can be a global optimum and we can shrink the set to [6]. On the other hand, if we know that $\epsilon_0 = O(\epsilon)$ holds with high probability, we can construct a linear lower bound for the objective function in [4] and $\{7,\dots,10\}$. In the both sets, the decrease of the lower bound

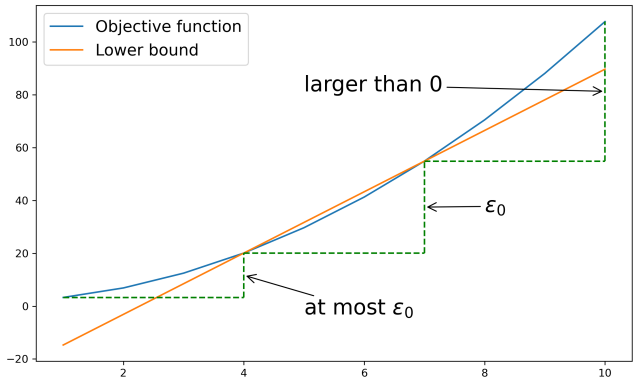


Figure 1 An example of the iteration of the TS algorithm.

is at most ϵ_0. Therefore, we have the relation $\min_{x\in\{4,5,6,7\}} f(x) \le \min_{x\in[10]} f(x) + \epsilon_0$, which implies that $\{4,\dots,7\}$ contains ϵ_0-optimal solutions with high probability and we can shrink the set to $\{4,\dots,7\}$ in the next iteration.

The algorithm shrinks the length of the current interval by at least $1/3$ for each iteration. Thus, the total number of iterations is at most $O(\log_{1.5}(N))$ to shrink the set until there are at most 3 points. Then, the algorithm solves a sub-problem with at most 3 points. We can prove that Algorithm 1 achieves the PGS guarantee for any given convex problem without knowing further structural information, i.e., Algorithm 1 is an $[(\epsilon,\delta)\text{-PGS}, \mathcal{MC}]$-algorithm. By estimating the simulation cost of the algorithm, an upper bound on the expected simulation cost to achieve the PGS guarantee follows.

THEOREM 1. *Suppose that Assumptions 1-3 hold. Algorithm 1 is an $[(\epsilon,\delta)$-PGS$, \mathcal{MC}]$-algorithm. Furthermore, we have*

$$T(\epsilon,\delta,\mathcal{MC}) = O\left[\frac{\log(N)}{\epsilon^2}\log\left(\frac{\log(N)}{\delta}\right)+\log(N)\right] = \tilde{O}\left[\frac{\log(N)}{\epsilon^2}\log\left(\frac{1}{\delta}\right)\right].$$

Proof of Theorem 1. The proof is provided in EC.2.1. □

We provide an explanation on the additional $\log(N)$ term. We note that in practice, the number of simulation samples taken in each iteration must be an integer, while the simulation cost is treated as a real number in our complexity analysis. Hence, the practical simulation cost of each iteration should be the smallest integer larger than the theoretical simulation cost, which introduces an extra $O(1)$ term. Then, the total expected simulation cost of Algorithm 1 should contain an extra $O(\log(N))$ term, which is not related to δ and is relatively small compared to the main term when δ is small.

REMARK 1. The term in the $\tilde{O}(\cdot)$ notation reflects the asymptotic simulation cost when $\delta \to 0$. The asymptotic simulation cost is commonly used in multi-armed bandits literature to compare the

computational complexities of different algorithms (Lai and Robbins 1985, Burnetas and Katehakis 1996, Karnin et al. 2013, Jamieson et al. 2014, Chen et al. 2016, Kaufmann et al. 2016). In practice, the failing probability δ is usually not small enough to enter the asymptotic regime and thus the simulation cost of algorithms may deviate from the asymptotic simulation cost. Therefore, we provide both the non-asymptotic and the asymptotic simulation costs for all algorithms.

3.2. Shrinking Uniform Sampling Algorithm and Upper Bound on Expected Simulation Cost

We have shown that the expected simulation cost of TS algorithm for the PGS guarantee has a $\log(N)$ dependence on N. Then, one may naturally ask: is there any algorithm for the PGS guarantee whose simulation cost has a better dependence on N? The answer is affirmative. In this subsection, the shrinking uniform sampling (SUS) algorithm for the PGS guarantee is proposed, which is proven to have a simulation cost as $O[\epsilon^{-2}(\log(N)+\log(1/\delta))]$, which grows as $\epsilon^{-2}\log(1/\delta)$ in the asymptotic regime $\delta \to 0$. Similarly, utilizing the idea of localization, the SUS algorithm maintains a set of active points and shrinks the set in each iteration until there are at most 2 points. However, instead of only sampling at 3-quantiles points of the current interval, the SUS algorithm samples all points in the current active set but with much fewer simulations. We give the pseudo-code in Algorithm 2.

Algorithm 2 Shrinking uniform sampling algorithm for the PGS guarantee

Input: Model $\mathcal{X}=[N],(\mathsf{Y},\mathcal{B}_{\mathsf{Y}}),F(x,\xi_x)$, optimality guarantee parameters ϵ,δ.

Output: An (ϵ,δ)-PGS solution x^* to problem (1).

Set the active set $\mathcal{S}\leftarrow\mathcal{X}$.
Set the step size $s\leftarrow 1$, maximal number of comparisons $T_{max}\leftarrow N$.
Set number of samples $n_x\leftarrow 0$ simulated at x for all $x\in\mathcal{X}$.
while the size of $\mathcal{S}$ is at least 3 **do** ▷ Iterate until $\mathcal{S}$ has at most 2 points.
    for $x\in\mathcal{S}$ **do**
        Simulate independent copies of $F(x,\xi_x)$ such that $h[n_x,\sigma,1-\delta/(2T_{max})]\le|\mathcal{S}|\cdot\epsilon/80$.
    end for
    Compute the empirical mean (using all simulated samples) $\hat{F}_{n_x}(x)$ for all $x\in\mathcal{S}$.
    if $\hat{F}_{n_x}(x)+h[n_x,\sigma,1-\delta/(2T_{max})]\le\hat{F}_{n_y}(y)-h[n_y,\sigma,1-\delta/(2T_{max})]$ for some $x,y\in\mathcal{S}$ **then** ▷ **Type-I Operation**
        if $x<y$ **then**
            Remove all points $z\in\mathcal{S}$ with the property $z\ge y$ from $\mathcal{S}$.
        else
            Remove all points $z\in\mathcal{S}$ with the property $z\le y$ from $\mathcal{S}$.

14: **end if**
15: **else** ▷ **Type-II Operation**
16: Update the step size $s \leftarrow 2s$.
17: Update $\mathcal{S} \leftarrow \{x_{min}, x_{min}+s, \ldots, x_{min}+ks\}$, where $x_{min} = \min_{x\in\mathcal{S}} x$ and $k = \lceil |\mathcal{S}|/2 \rceil - 1$.
18: **end if**
19: **end while** ▷ Now $\mathcal{S}$ has at most 2 points.
20: Simulate $\tilde{n}$ independent copies of $F(x,\xi_x)$ for each $x \in \{x_L, \ldots, x_U\}$, where $\tilde{n}$ is the smallest integer such that $h[\tilde{n}, \sigma, 1-\delta/(2T_{max})] \le \epsilon/4$.
21: Return the point in $\mathcal{S}$ with minimal empirical mean.

There are two kinds of shrinkage operations in Algorithm 2, which we denote as Type-I and Type-II Operations. Intuitively, Type-I Operations are implemented when we can compare and differentiate the function values of two points with high probability, and Type-II Operations are implemented when all points have similar function values. In the latter case, we prove that there exists a neighboring point to the optimum that has a function value at most $\epsilon/2$ larger than the optimum. Hence, we can discard every other point in $\mathcal{S}$ (the set in the algorithm that contains a potential good selection) with at least one $\epsilon/2$-optimal point remaining in the active set. We give a rough estimate to the expected simulation cost of Algorithm 2. We assign an order to points in $\mathcal{X}$ by the time they are discarded from $\mathcal{S}$. Points discarded in the same iteration are ordered randomly. Then, for the last k-th discarded point x_k, there are at least k points in $\mathcal{S}$ when x_k is discarded. By the second termination condition in Line 17, the confidence half-width at x_k is at least $k\epsilon/80$. If the Hoeffding bound is used, simulating $\tilde{O}(\epsilon^{-2}k^{-2}\log(1/\delta))$ times is enough to achieve the confidence half-width. Recalling the fact that $\sum_k k^{-2} < \pi^2/6 = O(1)$, if we sum the simulation cost over $k \in [N]$, the total expected simulation cost is bounded by $\tilde{O}(\epsilon^{-2}\log(1/\delta))$ and is independent of N. We note that we are able to reuse the samples in the previous rounds since we use the union bound to bound the total failing probability across iterations, which does not require the independence between samples in different iterations. In addition, we mention that the bound $|\mathcal{S}|\epsilon/80$ in lines 10 and 17 is not optimal and we choose this bound since the proof is simpler using this upper bound, and the expected simulation cost is only a constant factor worse than that of the case when the optimal bound is chosen. The following theorem proves that Algorithm 2 indeed achieves the PGS guarantee for any convex problem and provides a rigorous upper bound on the expected simulation cost $T(\epsilon, \delta, \mathcal{MC})$.

THEOREM 2. *Suppose that Assumptions 1-3 hold. Algorithm 2 is an* $[(\epsilon,\delta)$*-PGS*$, \mathcal{MC})]$*-algorithm. Furthermore, we have*

$$T(\epsilon,\delta,\mathcal{MC}) = O\left[\frac{1}{\epsilon^2}\log\left(\frac{N}{\delta}\right) + N\right] = \tilde{O}\left[\frac{1}{\epsilon^2}\log\left(\frac{1}{\delta}\right)\right].$$

Proof of Theorem 2. The proof is provided in EC.2.2. □

If we consider the asymptotic regime $\delta \ll 1$ (which is considered in Kaufmann et al. (2016)), the expected simulation cost of the SUS algorithm grows as $\epsilon^{-2}\log(1/\delta)$. This dependence is asymptotic and holds in the sense that the required failing probability δ tends to be very small. When δ is moderately large, the cost can depend on N. We demonstrate in the numerical experiments this asymptotic independence.

3.3. Lower Bound on Expected Simulation Cost

In this subsection, we consider the lower bounds on the expected simulation costs for all of the simulation-optimization algorithms that satisfy certain optimality guarantee for general convex problems. The lower bounds show the fundamental limit behind the simulation-optimization algorithms for general selection problems with a convex structure. By comparing those lower bounds with the upper bounds established for specific simulation-optimization algorithms, we can conclude that the SUS algorithm is optimal up to a constant factor. The lower bound on $T(\epsilon, \delta, \mathcal{MC})$, i.e., the expected simulation cost for achieving the PGS guarantee, is derived in Corollary EC.1, which is also presented in the following corollary.

Corollary 1. *Suppose that Assumptions 1-3 hold. We have*

$$T(\epsilon, \delta, \mathcal{MC}) \geq \Theta\left[\frac{1}{\epsilon^2}\log\left(\frac{1}{\delta}\right)\right].$$

Combining with the upper bounds derived in Sections 3.1 and 3.2, we conclude that the TS algorithm has a $\log(N)$ order gap for the PGS guarantee, while the SUS algorithm is optimal up to a constant in the asymptotic regime $\delta \ll 1$. However, the space complexities of the TS algorithm and the SUS algorithms are $O(\log(N))$ and $O(N)$, respectively. The space complexity of an algorithm refers to the amount of memory used by a program to execute the algorithm. This observation implies that we need to consider the trade-off between the simulation cost and the space complexity when choosing the best algorithm.

Before concluding this section, we note that the subgradient descent algorithm in Zhang et al. (2020) requires the knowledge of the Lipschitz constant and has a simulation cost as $\tilde{O}(N^2\epsilon^{-2}\log(1/\delta))$, which is $O(N^2)$ larger than that of the TS and the SUS algorithms. This observation implies that subgradient-based search methods may not be able to fully utilize the discrete nature and the convex structure of problem (1), especially for low-dimensional problems. Therefore, the proposed algorithms in this section provide a non-trivial improvement for solving one-dimensional convex optimization via simulation problems and hint a potential improvement direction (namely, localization-based methods) for multi-dimensional problems.

4. Simulation-optimization Algorithms and Complexity Analysis: Multi-dimensional Case

In this section, we propose simulation-optimization algorithms to achieve the PGS guarantee for convex discrete optimization via simulation problems with multi-dimensional decision variables. The decision space is considered as $\mathcal{X} = [N]^d$. In the multi-dimensional case, the discrete convexity of f is defined by the so-called $L^\natural$-convexity, which is defined by the mid-point convexity for discrete variables. The exact definition will be given in Section 4.1. The $L^\natural$-convexity can lead to the property that the discrete convex function has a convex extension along with an explicit subgradient defined on the convex hull of $\mathcal{X}$.

We outline the intuition underlying the algorithm design of this section before discussing the details. Since we have observed the power of localization from the one-dimensional case, the major approach is to design multi-dimensional algorithms based on the same idea. The first idea of applying the localization technique is to extend the TS algorithm to the multi-dimensional case. A direct generalization of the TS algorithm results in the zeroth-order stochastic ellipsoid method (Agarwal et al. 2011) and the zeroth-order random walk method (Liang et al. 2014), whose computational complexities have $O(d^{33})$ and $O(d^{14})$ dependence on the dimension, respectively. On the other hand, we show in the appendix that the SUS method can be naturally extended to the multi-dimensional case. The multi-dimensional SUS algorithm also has an expected simulation cost independent of the scale N using the asymptotic criterion in Kaufmann et al. (2016) (i.e., when δ is sufficiently small). However, the expected simulation cost has an exponential dependence on the dimension d and, therefore, the SUS algorithm is only suitable for low-dimensional problems.

We thus take an alternative approach and combine the localization operation with the *subgradient information*, which is known to be useful for high-dimensional problems. In this work, we design stochastic cutting-plane methods, which utilize properties of $L^\natural$-convex functions and the Lovász extension to evaluate unbiased stochastic subgradients at each point via finite difference. More specifically, we develop a new framework to design stochastic cutting-plane methods and thus reduce the dependence of the simulation cost on d. A straightforward application of our proposed framework leads to stochastic cutting-plane methods whose simulation cost has a $O(d^3)$ dependence on d. In addition, stochastic cutting-plane methods have only a logarithmic dependence on the Lipschitz constant L, while the subgradient-based method in Zhang et al. (2020) has a higher-order dependence on L. Further utilizing the *discrete nature* of problem (1), we develop the dimension reduction algorithm, whose simulation cost is upper bounded by a constant that is independent of the Lipschitz constant. In addition, the dimension reduction algorithm does not require any prior knowledge about the Lipschitz constant, which makes it suitable for the case when prior knowledge about the objective function is limited.

4.1. Discrete Convex Functions in Multi-dimensional Space

Similar to the one-dimensional case, discrete convex functions in multi-dimensional space are characterized by the discrete midpoint convexity property and have a convex piecewise linear extension. In Murota (2003), this collection of functions is named $L^\natural$-convex functions and is proved to have the property that local optimality implies global optimality. The exact definition of $L^\natural$-convex functions is given below.

DEFINITION 3. A set $\mathcal{X} \subset \mathbb{Z}^d$ is called a $L^\natural$-convex set if for all $x, y \in \mathcal{X}$, we have $\lceil (x+y)/2 \rceil, \lfloor (x+y)/2 \rfloor \in \mathcal{X}$. A function $f(x) : \mathcal{X} \mapsto \mathbb{R}$ is called a $L^\natural$-convex function if the feasible set $\mathcal{X}$ is $L^\natural$-convex and the discrete midpoint convexity property holds:

$$f(x) + f(y) \geq f(\lceil (x+y)/2 \rceil) + f(\lfloor (x+y)/2 \rfloor), \quad \forall x, y \in \mathcal{X}.$$

The set of models such that $f(x)$ is $L^\natural$-convex on $\mathcal{X}$ is denoted as $\mathcal{MC}$. The set of models such that $f(x)$ is $L^\natural$-convex with an indifference zone parameter c is denoted as $\mathcal{MC}_c$.

REMARK 2. As noted in Zhang et al. (2020), the feasible set $\mathcal{X} = [N]^d$ is a $L^\natural$-convex set and thus we only need the discrete midpoint convexity property to define $L^\natural$-convex functions on $\mathcal{X}$. In the case when $d = 1$, the $L^\natural$-convexity is equivalent to the discrete convexity defined in Section 3. Hence, the definition of $\mathcal{MC}$ is consistent with Section 3.

The following property shows that $L^\natural$-convex functions can be viewed as a generalization of submodular functions.

LEMMA 1 **(Murota (2003))**. *Suppose that the function $f(x) : \mathcal{X} \mapsto \mathbb{R}$ is $L^\natural$-convex. Then, the translation submodularity holds:*

$$f(x)+f(y) \geq f((x-\alpha\mathbf{1}) \vee y) + f(x \wedge (y+\alpha\mathbf{1})), \quad \forall x, y \in \mathcal{X},\ \alpha \in \mathbb{N} \text{ s.t. } (x-\alpha\mathbf{1}) \vee y,\ x \wedge (y+\alpha\mathbf{1}) \in \mathcal{X}.$$

By the translation submodularity, the $L^\natural$-convex function restricted to a cube $x + \{0,1\}^d \subset \mathcal{X}$ is a submodular function. Therefore, the Lovász extension (Lovász 1983) can be constructed as the convex piecewise linear extension inside each cube. In addition, $L^\natural$-convex functions are integrally convex functions (Murota 2003). Hence, we can obtain a continuous convex function on $[1, N]^d$ by piecing together the Lovász extension in each cube. More importantly, we can calculate a subgradient of the convex extension with $O(d)$ function value evaluations. Hence, $L^\natural$-convex functions provide a good framework for extending the continuous convex optimization theory to the discrete case. In the remainder of this subsection, we specify this intuition of $L^\natural$-convex functions in a rigorous way. We first define the Lovász extension of submodular functions and give an explicit subgradient of the Lovász extension at each point.

DEFINITION 4. Suppose that $f(x):\{0,1\}^d \mapsto \mathbb{R}$ is a submodular function. For any $x=(x_1,\ldots,x_d)\in[0,1]^d$, we say that a permutation $\alpha_x:[d]\mapsto[d]$ is a **consistent permutation** of x, if

$$x_{\alpha_x(1)} \geq x_{\alpha_x(2)} \geq \cdots \geq x_{\alpha_x(d)}.$$

We define $S^{x,0}:=(0,\ldots,0)\in\mathcal{X}$. For each $i\in[d]$, the **i-th neighbouring points** of x is defined as

$$S^{x,i}:=\sum_{j=1}^{i} e_{\alpha_x(j)}\in\mathcal{X}, \tag{3}$$

where vector e_k is the k-th vector in the standard basis of $\mathbb{R}^d$. We define the **Lovász extension** $\tilde{f}(x):[0,1]^d\mapsto\mathbb{R}$ as

$$\tilde{f}(x):=f\left(S^{x,0}\right)+\sum_{i=1}^{d}\left[f\left(S^{x,i}\right)-f\left(S^{x,i-1}\right)\right]x_{\alpha_x(i)}. \tag{4}$$

We note that the value of the Lovász extension does not rely on the choice of the consistent permutation. We list several well-known properties of the Lovász extension and refer their proofs to Lovász (1983), Fujishige (2005).

LEMMA 2. *Suppose that Assumptions 1-3 hold. Then, the following properties hold for $\tilde{f}(x)$:*

(i) For any $x\in\mathcal{X}$, it holds that $\tilde{f}(x)=f(x)$.

(ii) The minimizers of $\tilde{f}(x)$ satisfy $\arg\min_{x\in[0,1]^d}\ \tilde{f}(x)=\arg\min_{x\in\mathcal{X}}\ f(x)$.

(iii) The function $\tilde{f}(x)$ is a convex function on $[0,1]^d$.

(iv) A subgradient $g\in\partial\tilde{f}(x)$ is given by

$$g_{\alpha_x(i)}:=f\left(S^{x,i}\right)-f\left(S^{x,i-1}\right),\quad \forall i\in[d]. \tag{5}$$

It is proved in Zhang et al. (2020) that, with $2d$ simulation runs, we can generate a stochastic subgradient at point x by

$$\hat{g}_{\alpha_x(i)}:=F\left(S^{x,i},\xi_i^1\right)-F\left(S^{x,i-1},\xi_{i-1}^2\right),\quad \forall i\in[d]. \tag{6}$$

Before we examine the properties of the Lovász extension, we discuss the possibility of using of common random number (CRN) in the computation of $\hat{g}$.

REMARK 3. We note that in the computation of $\hat{g}$, if the use of CRN can be successfully implemented and if the resulting sample-path functions were themselves convex, then the use of CRN can contribute to a significant variance reduction on the stochastic subgradient. Such variance reduction should then translate to the improvement of efficiency (or, equivalently, reduction of expected simulation costs) of the proposed simulation-optimization algorithms. To theoretically analyze the reduction of simulation costs in terms of the dependence on the dimension d and scale N, more

concrete assumptions need to be made on how exactly the CRN is implemented. For example, if we use CRN across different choices of value for the decision variable, it is not always the case that the associated random objects with one choice and another choice are exactly the same. One probably is able to use CRN on the "common" part of the random objects and not use CRN on the "uncommon" part of the random objects. We will then need an assumption on quantifying the common part and the uncommon part. As another example, we would need to specify whether or not a single simulation run can simultaneously generate multiple evaluations of the function at neighboring choices of value for the decision variable, supposing that on this single simulation run the use of CRN is exploited to its best extent. A separate challenge arises in that it is not always easy to implement CRN for complicated stochastic systems. We would like to briefly discuss this in the current work and leave a more detailed analysis as future work.

Then, we show that the Lovász extension in the neighborhood of each point can be pieced together to form a convex function on $\text{conv}(\mathcal{X}) = [1, N]^d$. We define the local neighborhood of each point $y \in [1, N-1]^d$ as the cube

$$\mathcal{C}_y := y + [0, 1]^d.$$

We denote the objective function $f(x)$ restricted to $\mathcal{C}_y \cap \mathcal{X}$ as $f_y(x)$, which is submodular by the translation submodularity of $f(x)$. For point $x \in \mathcal{C}_y$, we denote α_x as a consistent permutation of $x - y$ in $[0, 1]^d$ and, for each $i \in \{0, 1, \ldots, d\}$, the corresponding i-th neighboring point of x is defined as

$$S^{x,i} := y + \sum_{j=1}^{i} e_{\alpha_x(j)}.$$

We note that the definition of $S^{x,i}$ is consistent with that in (3) since the corresponding point y is $(0, \ldots, 0)$ in (3). Then, the Lovász extension of $f_y(x)$ in $\mathcal{C}_y$ can be calculated as

$$\tilde{f}_y(x) := f\left(S^{x,0}\right) + \sum_{i=1}^{d} \left[f\left(S^{x,i}\right) - f\left(S^{x,i-1}\right)\right] x_{\alpha_x(i)}.$$

Now, we piece together the Lovász extension in each cube by defining

$$\tilde{f}(x) := \tilde{f}_y(x), \quad \forall x \in [1, N]^d,\ y \in [N-1]^d \quad \text{s.t. } x \in \mathcal{C}_y. \tag{7}$$

It is proved in Murota (2003) and Zhang et al. (2020) that $\tilde{f}(x)$ is well-defined and is a convex function.

LEMMA 3. *The function $\tilde{f}(x)$ in (7) is well-defined and convex on $\mathcal{X}$.*

Utilizing properties (i) and (ii) of Lemma 2, problem (1) is equivalent to the relaxed problem

$$f^* := \min_{x\in[1,N]^d} \tilde{f}(x), \tag{8}$$

which is convex according to Lemma 3. Moreover, the subgradient (5) and stochastic subgradient (6) are valid for the convex extension $\tilde{f}(x)$. Similarly, (stochastic) subgradients can be computed in the neighboring cube of each point and it does not matter which cube is chosen for points belonging to multiple cubes. Finally, the linear-time rounding process proposed in Zhang et al. (2020) reduces the problem of finding high-precision solutions of problem (1) to that of the relaxed problem (8). The pseudo-code is provided in the following algorithm.

Algorithm 3 Rounding process to a feasible solution

Input: Model $\mathcal{X}, \mathcal{B}_Y, F(x,\xi_x)$, optimality guarantee parameters ϵ, δ, $(\epsilon/2, \delta/2)$-PGS solution $\bar{x}$ to problem (8).

Output: An (ϵ,δ)-PGS solution x^* to problem (1).

1: Compute a consistent permutation of $\bar{x}$, denoted as α.
2: Compute the neighbouring points of $\bar{x}$, denoted as $S^0,\ldots,S^d$.
3: Simulate n times at S^i for all i, where n is the smallest integer such that $h[n,\sigma,1-\delta/(4d)] \le \epsilon/4$.
4: Return the optimal point $x^* \leftarrow \arg\min_{S\in\{S^0,\ldots,S^d\}} \hat{F}_n(S)$.

The following theorem verifies the correctness and estimates the simulation cost of Algorithm 3.

LEMMA 4 **(Zhang et al. (2020))**. *Suppose that Assumptions 1-3 hold. The solution returned by Algorithm 3 satisfies the (ϵ,δ)-PGS guarantee and the simulation cost of Algorithm 3 is at most*

$$O\left[\frac{d}{\epsilon^2}\log\left(\frac{d}{\delta}\right)+d\right] = \tilde{O}\left[\frac{d}{\epsilon^2}\log\left(\frac{1}{\delta}\right)\right].$$

The rounding process for the (c,δ)-PCS-IZ guarantee follows by choosing $\epsilon = c/2$.

4.2. Stochastic Cutting-plane Methods: Stochastic Separation Oracles

Now, we consider designing simulation-optimization algorithms with simulation costs having a polynomial dependence on the problem parameters d and N. In addition, we reiterate that the goal is to design algorithms that do not require the information about the Lipschitz constant L and the simulation cost is upper bounded by a constant that is independent of L. Intuitively, the subgradient information is useful for high-dimensional problems, while the localization operation is good at utilizing the discrete nature of the problem and getting rid of the dependence on the Lipschitz constant. Therefore, one may expect subgradient-based localization methods to satisfy the aforementioned requirements. Using the definitions and tools introduced in Section 4.1, we

are able to design the desired algorithm in two steps. In this subsection, we first introduce the definition of stochastic separation oracles and give a novel framework to design stochastic cutting-plane methods via deterministic cutting-plane methods. Straightforward extensions of deterministic cutting-plane methods require prior knowledge about L and the simulation cost has a logarithmic dependence on L. Hence, the following assumption is required.

ASSUMPTION 4. *The ℓ_∞-Lipschitz constant L is known a priori. Namely, we have*

$$|f(x) - f(y)| \leq L, \quad \forall x, y \in \mathcal{X}, \quad \text{s.t. } \|x - y\|_\infty \leq 1.$$

In the next subsection, we incorporate the stochastic cutting-plane methods with the dimension reduction operation. The resulting algorithm, named as the dimension reduction algorithm, does not require prior information about L and the simulation cost is upper bounded by a constant that is independent of L. We note that the design of the dimension reduction algorithm is the main objective of this section and stochastic cutting-plane methods mainly serve as an example of our novel framework.

In each iteration of a cutting-plane algorithm, a cutting hyperplane is generated to shrink the subset of potentially optimal choices of decision variables. In other words, the cutting hyperplane is used to localize the optimal solution. When the volume is small enough, the Lipschitz continuity implies that the all points in the polytope have their objective values close to the optimal value. In general, cutting-plane methods have a higher order of dependence on the dimension than subgradient-based search methods (Lee et al. 2015, Jiang et al. 2020). Hence, we expect that the simulation cost of cutting-plane methods will have a higher-order dependence on the problem dimension compared to that of subgradient-based search methods. As a counterpart of separation oracles, we introduce the stochastic separation oracle, named as the (ϵ, δ)-separation oracle, to characterize the accuracy of separation oracles in the stochastic case.

DEFINITION 5. A **(ϵ, δ)-separation oracle ((ϵ, δ)-$\mathcal{SO}$)** is a function on $[1, N]^d$ with the property that for any input $x \in [1, N]^d$, it outputs a stochastic vector $\hat{g}_x \in \mathbb{R}^d$ such that the inequality

$$f(y) \geq f(x) - \epsilon, \quad \forall y \in [1, N]^d \cap H$$

holds with probability at least $1 - \delta$, where the half space H is defined as $\{z : \langle \hat{g}_x, z - x \rangle \geq 0\}$.

Before we state algorithms, we give a concrete example of (ϵ, δ)-$\mathcal{SO}$ oracles and provide an upper bound on the expected simulation cost of evaluating each oracle. We define the averaged subgradient estimator $\hat{g}^n$ as

$$\hat{g}^n_{\alpha_x(i)} := \hat{F}_n\left(S^{x,i}\right) - \hat{F}_n\left(S^{x,i-1}\right), \quad \forall i \in [d], \tag{9}$$

where α_x is a consistent permutation of x, $n \geq 1$ is the number of samples, and $\hat{F}_n$ is the empirical mean of n independent evaluations of F. The following lemma gives a lower bound on n to guarantee that $\hat{g}^n$ is an (ϵ, δ)-$\mathcal{SO}$ oracle.

LEMMA 5. *Suppose that Assumptions 1-3 hold. If we choose n such that*

$$n = \Theta\left[\frac{dN^2}{\epsilon^2}\log\left(\frac{1}{\delta}\right)\right],$$

then $\hat{g}^n$ is an (ϵ, δ)-$\mathcal{SO}$ oracle. Moreover, the expected simulation cost of generating an (ϵ, δ)-$\mathcal{SO}$ oracle is at most

$$O\left[\frac{d^2N^2}{\epsilon^2}\log\left(\frac{1}{\delta}\right) + d\right] = \tilde{O}\left[\frac{d^2N^2}{\epsilon^2}\log\left(\frac{1}{\delta}\right)\right].$$

Proof of Lemma 5. The proof is provided in EC.6.1. □

We note that the condition in Lemma 5 provides a sufficient condition of the $\mathcal{SO}$ oracle. In practice, the value of n can be much smaller than the bound in Lemma 5; see numerical examples in Section 5. To show the usefulness of the stochastic separation oracle, we extend Vaidya's cutting-plane method (Vaidya 1996) to a stochastic cutting-plane method that can find high-precision solutions with high probability in the stochastic case. Vaidya's cutting-plane method maintains a polytope that contains the optimal points and iteratively reduces the volume of polytope by generating a separation oracle at the approximate volumetric center. We provide the pseudo-code of deterministic Vaidya's method in EC.4 for the self-contained purpose. Other deterministic cutting-plane methods based on reducing the volume of a polytope can also be extended to the stochastic case using our novel framework, and we consider Vaidya's method mainly for its simplicity.

It is desirable to prove that by substituting the separation oracles with stochastic separation oracles, Vaidya's cutting-plane method can be used to find high-precision solutions with high probability. The pseudo-code of the stochastic cutting-plane method is given in Algorithm 4.

Algorithm 4 Stochastic cutting-plane method for the PGS guarantee

Input: Model $\mathcal{X}, (\mathsf{Y}, \mathcal{B}_{\mathsf{Y}}), F(x, \xi_x)$, optimality guarantee parameters ϵ and δ, Lipschitz constant L, (ϵ, δ)-$\mathcal{SO}$ oracle $\hat{g}$.

Output: An (ϵ, δ)-PGS solution x^* to problem (1).

1: Set the initial polytope $P \leftarrow [1, N]^d$.
2: Set the constant $\rho \leftarrow 10^{-7}$. ▷ Constant ρ corresponds to ϵ in Vaidya (1996).
3: Set the number of iterations $T_{max} \leftarrow \lceil 2d/\rho \cdot \log[dNL/(\rho\epsilon)] \rceil$.
4: Initialize the set of points used to query separation oracles $\mathcal{S} \leftarrow \emptyset$.
5: Initialize the volumetric center $z \leftarrow (N+1)/2 \cdot (1, 1, \ldots, 1)^T$.
6: **for** $T = 1, 2, \ldots, T_{max}$ **do**

Decide adding or removing a cutting plane by Vaidya's method.
if add a cutting plane **then**
  Evaluate an $(\epsilon/8, \delta/4)$-$\mathcal{SO}$ oracle $\hat{g}_z$ at z.
  if $\hat{g}_z = 0$ **then**
    Round z to an integral solution by Algorithm 3 and return the rounded solution.
  end if
  Add the current point z to $\mathcal{S}$.
else if remove a cutting plane **then**
  Remove corresponding point z from $\mathcal{S}$.
end if
Update the approximate volumetric center z by a Newton-type method.
end for ▷ There are at most $O(d)$ points in $\mathcal{S}$ by Vaidya's method.
Find an $(\epsilon/4, \delta/4)$-PGS solution $\hat{x}$ of problem $\min_{x\in\mathcal{S}} f(x)$.
Round $\hat{x}$ to an integral solution by Algorithm 3.

We note that if the approximate volumetric center z is not in $[1, N]^d$, then we choose a violated constraint $x_i \geq 1$ or $x_i \leq N$ and return e_i or $-e_i$ as the separating vector, respectively. For arithmetic operations, each iteration of Algorithm 4 requires $O(d)$ inversions and multiplications of $d \times d$ matrices. Each inversion and multiplication can be finished within $O(d^\omega)$ arithmetic operations, where $\omega < 2.373$ is the matrix exponent (Alman and Williams 2020). Hence, Algorithm 4 needs $O(d^{\omega+1})$ arithmetic operations for each iteration. The calculation of the number of iterations T_{max} is provided in EC.6.2. The correctness and the expected simulation cost of Algorithm 4 are studied in the following theorem.

THEOREM 3. *Suppose that Assumptions 1-4 hold. Algorithm 4 returns an (ϵ, δ)-PGS solution and we have*

$$T(\epsilon, \delta, \mathcal{MC}) = O\left[\frac{d^3N^2}{\epsilon^2}\log\left(\frac{dLN}{\epsilon}\right)\log\left(\frac{1}{\delta}\right) + d^2\log\left(\frac{dLN}{\epsilon}\right)\right] = \tilde{O}\left[\frac{d^3N^2}{\epsilon^2}\log\left(\frac{dLN}{\epsilon}\right)\log\left(\frac{1}{\delta}\right)\right].$$

Proof of Theorem 3. The proof is provided in EC.6.2. □

REMARK 4. We note that another popular deterministic cutting-plane method, the random walk-based cutting-plane method (Bertsimas and Vempala 2004), can also be extended to the stochastic case and achieves a better expected simulation cost

$$\tilde{O}\left[\frac{d^3N^2}{\epsilon^2}\log\left(\frac{LN}{\epsilon}\right)\log\left(\frac{1}{\delta}\right)\right]$$

at the expense of $\tilde{O}[d^6 + \log^2(1/\delta)]$ arithmetic operations in each iteration. We provide the pseudo-code in EC.4 for the self-contained purpose. Here, the $O[\log^2(1/\delta)]$ factor is required to ensure the

high-probability approximation to the centroid. Moreover, we note that the fast implementation of Vaidya's method in Jiang et al. (2020) reduces number of arithmetic operations in each iteration to $O(d^2)$.

REMARK 5. Stochastic cutting-plane methods can also be applied to problems that are defined on $[N]^d$ with linear constraints $\{x \in \mathbb{Z}^d : Ax \leq b\}$, since we can choose the initial polytope to be $\mathcal{X} := [1, N]^d \cap \{Ax \leq b\}$. The results in this section still hold if we replace N with $\max_{x,y\in\mathcal{X}} \|x-y\|_\infty$.

From Theorem 3, we can see that the upper bound on the expected simulation cost only has a logarithmic dependence on the Lipschitz constant L, which is better than the quadratic dependence of the subgradient-based algorithms in Zhang et al. (2020). In the next subsection, we further improve the dependence and develop a dimension reduction algorithm, whose simulation cost is upper bounded by a constant that is independent of the Lipschitz constant L.

4.3. Stochastic Cutting-plane Methods: Dimension Reduction Algorithm

In this subsection, we develop the dimension reduction algorithm, which does not require the knowledge about the Lipschitz constant L and whose simulation cost is upper bounded by a constant that is independent of L. The idea behind the dimension reduction algorithm is based on the following observation: if a convex body $P \subset \mathbb{R}^d$ has a volume $\text{vol}(P)$ smaller than $(d!)^{-1} = O[\exp(-(d+1/2)\log(d)+d)]$, then all integral points inside P must lie on a hyperplane. Otherwise, if there exist $d+1$ integral points $x_0, \ldots, x_d \in P$ that are not on the same hyperplane, then the convex body P contains the polytope $\text{conv}\{x_0, \ldots, x_d\}$, which has the volume

$$\frac{1}{d!} |\det(x_1 - x_0, \ldots, x_d - x_0)| \geq \frac{1}{d!},$$

where $\text{conv}(\cdot)$ is the convex hull and $\det(\cdot)$ is the determinant of matrices. This leads to a contradiction since we assume that $\text{vol}(P) < (d!)^{-1}$. Hence, we may use Vaidya's method or the random walk method to reduce the volume of the search polytope P to $O[\exp(-(d+1/2)\log(d)+d)]$, and then we reduce the problem dimension by projecting the polytope onto the hyperplane that all remaining points lie on. After $d-1$ dimension reductions, we have a one-dimensional convex problem and algorithms in Section 3 can be applied. This idea is summarized in Algorithm 5.

Algorithm 5 Dimension reduction algorithm for the PGS guarantee

Input: Model $\mathcal{X}, (\mathsf{Y}, \mathcal{B}_\mathsf{Y}), F(x, \xi_x)$, optimality guarantee parameters ϵ and δ, (ϵ, δ)-$\mathcal{SO}$ oracle $\hat{g}$.

Output: An (ϵ, δ)-PGS solution x^* to problem (1).

1: Set the initial polytope $P \leftarrow [1, N]^d$.

2: Initialize the set of points used to query separation oracles $\mathcal{S} \leftarrow \emptyset$.

3: **for** $d' = d, d-1, \ldots, 2$ **do** ▷ The current dimension d' is gradually reduced.
4: Initialize Vaidya's cutting-plane method.
5: **while** the volume of P is larger than $(d'!)^{-1}$ **do**
6: Take one step of Vaidya's cutting-plane method with $(\epsilon/4, \delta/4)$-$\mathcal{SO}$ oracle.
▷ Vaidya's cutting-plane method decides a suitable cutting plane H.
7: Add the point where the stochastic separation oracle is called to $\mathcal{S}$.
8: Shrink the volume of P using the cutting plane H.
9: **end while**
10: Find the hyperplane H that contains all integral points in P.
▷ If P contains no integral points, then an arbitrary hyperplane works.
11: Project P onto the hyperplane H. ▷ Reduce the dimension by 1.
12: **end for**
13: Find an $(\epsilon/4, \delta/4)$-PGS solution of the last one-dim problem and add the solution to $\mathcal{S}$.
14: Find the $(\epsilon/4, \delta/4)$-PGS solution $\hat{x}$ of problem $\min_{x \in \mathcal{S}} f(x)$.
15: Round $\hat{x}$ to an integral solution by Algorithm 3.

We note that the application of Vaidya's method in Line 6 refers to implementing the cutting-plane algorithm for one iteration. Namely, only a single cutting hyperplane will be generated. Importantly, the implementation of Vaidya's method in this step does not require the knowledge about the Lipschitz constant, since the Lipschitz constant is only used to calculate the total number of steps in Algorithm 4. In addition, Vaidya's cutting-plane method can be replaced with other deterministic cutting-plane methods. Furthermore, many cutting-plane methods, including Vaidya's method and random-walk-based method, guarantee that the volume of the polytope P is decreased at a constant rate. For example, the random walk-based cutting-plane method reduces the volume at the rate $1 - e^{-1}$ and after k iterations, the volume of P is at most $(1 - e^{-1})^k N^d$. Thus, we can terminate the cutting-plane method when the upper bound is lower than $(d!)^{-1}$.

We note that the search of hyperplane H in Line 10 does not require evaluations of the function $F(x, \xi_x)$ and therefore, it will not affect the simulation cost of Algorithm 5. The simplest algorithm to find the hyperplane H is to enumerate all hyperplanes generated by the points in $[N]^d$ and check if the condition $P \cap \mathbb{Z}^d \subset H$ is satisfied. This naive algorithm terminates in finite time but may require an exponential number of arithmetic operations. In Jiang (2021), the author reduced the problem of finding a hyperplane H to the problem of finding an approximate solution to the Shortest Vector Problem in lattices. When the volume of the current polytope P is small enough, it is proved that a set of Lenstra–Lenstra–Lovász (LLL)-reduced basis (Lenstra et al. 1982) contains the normal vector of the hyperplane H, namely, the vector $c \in \mathbb{Z}^d$ such that

$$\langle c, x - y \rangle = 0, \quad \forall x, y \in P \cap \mathbb{Z}^d.$$

The LLL algorithm (Lenstra et al. 1982), which only requires a polynomial number of arithmetic operations, can be applied to find the desired LLL-reduced basis. We show that their results can be extended to the stochastic case and combined with the framework in Section 4.2 to generate an algorithm that only requires a polynomial number of arithmetic operations.

Intuitively, the dimension reduction algorithm implements the stochastic cutting-plane method at each dimension from d to 1. Therefore, the total simulation cost is on the same order as the summation of i^3 for $i \in [d]$, which is on the order of $O(d^4)$. More rigorously, we provide the correctness and the simulation cost of Algorithm 5 in the following theorem.

THEOREM 4. *Suppose that Assumptions 1-3 hold. Algorithm 5 returns an (ϵ, δ)-PGS solution and we have*

$$T(\epsilon, \delta, \mathcal{MC}) = O\left[\frac{d^3N^2(d+\log(N))}{\epsilon^2}\log\left(\frac{1}{\delta}\right) + d^2(d+\log(N))\right] = \tilde{O}\left[\frac{d^3N^2(d+\log(N))}{\epsilon^2}\log\left(\frac{1}{\delta}\right)\right].$$

Proof of Theorem 4. The proof is provided in EC.6.3. □

We note that the idea of gradually reducing the dimension is proposed in our work and Jiang (2021) independently, although the author of Jiang (2021) has made the algorithm more practical. More specifically, if we allow exponentially many arithmetic operations, the LLL algorithm is not necessary. In that case, we can reduce the number of separation oracles to $O(d^2)$ and the computational complexity can be reduced to $\tilde{O}[d^4N^2\epsilon^{-2}\log(1/\delta)]$. From Theorem 4, we can see that the expected simulation cost of Algorithm 5 is upper bounded by a constant that is independent of the Lipschitz constant L. In addition, Algorithm 5 does not require any prior estimation of the upper bound of the Lipschitz constant. Since the estimated Lipschitz constant is likely to be much larger than the real Lipschitz constant, the error in the estimation will lead to a significant increase in the the simulation cost. Therefore, the dimension reduction algorithm is suitable when an estimate of the upper bound of the Lipschitz constant is difficult to obtain or the upper bound of the Lipschitz constant is large.

5. Numerical Experiments

In this section, we implement our proposed simulation-optimization algorithms that are guaranteed to find high-confidence high-precision solutions. Through these numerical experiments, we show that the localization methods proposed in this manuscript outperform benchmark algorithms on large-scale problems. We also compare our proposed algorithms to benchmark algorithms that do not utilize the $L^\natural$-convexity, e.g. the Industrial-strength COMPASS algorithm (Xu et al. 2010) and the R-SPLINE algorithm (Wang et al. 2013); see Section EC.8 for additional numerical results and discussions.

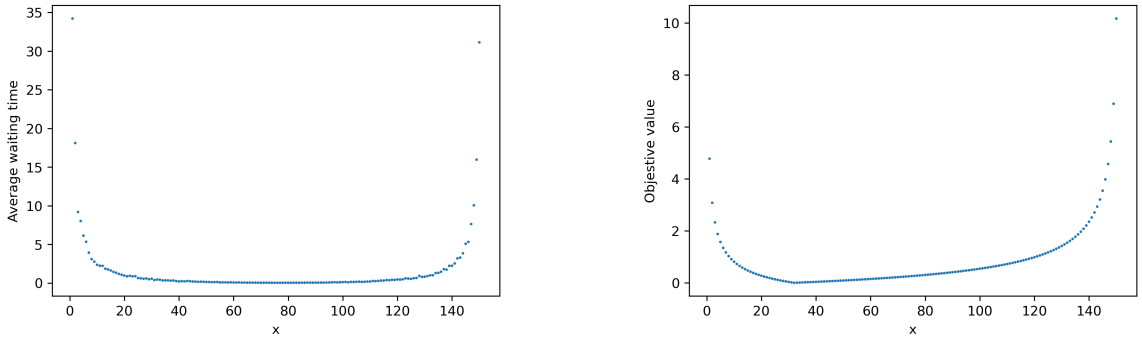
First, we consider the problem of finding the optimal allocation of a total number of N staffs to two queues so that the average waiting time for all of the arrivals from the two queues is minimized. Given the optimality parameters ϵ and δ, we empirically show that the TS algorithm and the SUS algorithm have respectively $O(\log N)$ and $O(1)$ dependence on the scale N, which supports our theoretical results. In addition, we construct a synthetic one-dimensional convex function with a similar landscape to show that the returned solution satisfies the high-probability guarantee. Second, we construct a multi-dimensional stochastic function, whose expectation is a separable convex function, i.e., functions of the form $f(x) = \sum_{i=1}^{d} f^i(x_i)$ for convex functions $f^1(x), \ldots, f^d(x)$, to test and compare the subgradient descent algorithm (Zhang et al. 2020) with the stochastic localization methods proposed in this work for different values of the scale N and dimension d, especially for large N. Similar to the one-dimensional case, we consider functions with a closed-form to check the coverage rate of the proposed algorithms. Finally, the multi-dimensional resource allocation problem in service systems is considered to compare the performance of proposed algorithms on practical problems.

5.1. Staffing Two Queues under Resource Constraints

Consider a service system that operates over a time horizon $[0, T]$ with two streams of customers arriving at the system. One example is that the system receives service requests from both online app-based customers and offline walk-in customers, and each stream needs dedicated servers assigned. The first stream of customers arrives according to a doubly stochastic non-homogeneous Poisson process $N_1 := (N_1(t) : t \in [0, T])$, with the customer service times being independent and identically distributed according to a distribution S_1. The second stream of customers obeys the same model with the process $N_2 := (N_2(t) : t \in [0, T])$ and distribution S_2. The two streams of customers form two separate queues and their arrival processes can be correlated. Suppose that the decision maker needs to staff the two queues separately. There are in total a number of $N+1$ homogeneous servers that work independently in parallel. Each server can handle the service requested by customers from either stream, one at a time. Suppose that no change on the staffing plan can be made once the system starts working. Assume that the system operates based on a first-come-first-serve routine, with unlimited waiting room in each queue, and that customers never abandon.

The decision maker's objective is to select the staffing level $x \in [N]$ for the first queue and the staffing level $N + 1 - x$ for the second queue, in order to minimize the expected average waiting time for all customers from the two streams over the time horizon $[0, T]$. In the numerical example, we consider $N \in \{10, 20, \ldots, 150\}$ and $T = 2$. The arrival processes N_1 and N_2 are non-homogeneous processes with random intensity functions $\Gamma_1 \cdot \lambda_1(t)$ and $\Gamma_2 \cdot \lambda_2(t)$, in which

$$\lambda_1(t) := 75 + 25\sin(0.3t), \quad \lambda_2(t) := 80 + 40\sin(0.2t).$$



(a) (b)

Figure 2 The landscapes of objective functions in the one-dimensional case. **(a)** The empirical average waiting time with $N = 150$. **(b)** The landscape of the synthetic convex function with scale $N = 150$ and optimum $x^* = 31$.

Positive-valued random variables Γ_1 and Γ_2 are defined as

$$\Gamma_1 := X + Z, \quad \Gamma_2 := Y - Z,$$

where X, Y are independent uniform random variables on $[0.75, 1.25]$ and Z is an independent uniform random variable on $[-0.5, 0.5]$. The service time distribution S_1 is log-normal distributed with mean 0.75 and variance 0.1. The service time distribution S_2 is gamma distributed with mean 0.65 and variance 0.1. Figure 2 plots an empirical average waiting time as a function of the discrete decision variable x. It can be observed that the landscape around the optimum is extremely flat and such property may cause challenges for algorithms that aim to exactly select the optimal solution (i.e., the PCS guarantee). In practice, the decision maker may be indifferent about a very small difference in the averaging waiting time performance, when the small difference does not impact much on customers satisfaction. Instead, algorithms that are designed for the (ϵ, δ)-PGS guarantee do not suffer from the extremely flat landscape around the global optimum.

5.2. Separable Convex Function Minimization

We consider the problem of minimizing a stochastic function whose expectation is a separable $L^\natural$-convex function of the form

$$f_{c,x^*}(x) := \sum_{i=1}^{d} c_i g(x_i; x_i^*),$$

where $c_i \in [0.75, 1.25]$, $x_i^* \in \{1, \ldots, \lfloor 0.3N \rfloor\}$ for all $i \in [d]$ and

$$g(x; x^*) := \begin{cases} \sqrt{\frac{x^*}{x}} - 1 & \text{if } x \le x^* \\ \sqrt{\frac{N+1-x^*}{N+1-x}} - 1 & \text{if } x > x^* \end{cases}, \quad \forall x, x^* \in [N],$$

It can be observed that the function $f_{c,x^*}(x)$ is the sum of separable convex functions and therefore is $L^\natural$-convex. Moreover, the function $f_{c,x^*}(x)$ has the optimum x^* associated with the optimal value

0. The objective function has a similar landscape as the average waiting time; see Figure 2. For stochastic evaluations, we add Gaussian noise with mean 0 and variance 1. The advantage of this numerical example is that the expected objective function has a closed form, and we are able to exactly compute the optimality gap of the solutions returned by the proposed algorithms.

5.3. Resource Allocation Problem in Service Systems

We consider the 24-hour operation of a service system with a single stream of incoming customers. The customers arrive according to a doubly stochastic non-homogeneous Poisson process with the intensity function

$$\Lambda(t) := 0.5\lambda N \cdot (1 - |t - 12|/12), \quad \forall t \in [0, 24],$$

where λ is a positive constant and N is a positive integer. Each customer requests a service with the service time independent and identically distributed according to the log-normal distribution with mean $1/\lambda$ and variance 0.1. We divide the 24-hour operation into d time slots with length $24/d$ for some positive integer d. For the i-th time slot, there are $x_i \in [N]$ of homogeneous servers that work independently in parallel and the number of servers cannot be changed during the slot. Assume that the system operates based on a first-come first-serve routine, with an unlimited waiting room in each queue, and that customers never abandon.

The decision maker's objective is to select the staffing level $x := (x_1, \ldots, x_d)$ such that the total waiting time of all customers is minimized. Namely, by letting $f(x)$ be the expected total waiting time under the staffing plan x, the optimization problem can be written as

$$\min_{x \in [N]^d} f(x). \tag{10}$$

It has been proved in Altman et al. (2003) that the function $f(\cdot)$ is multimodular. We define the linear transformation

$$g(y) := (y_1, y_2 - y_1, \ldots, y_d - y_{d-1}) \quad \forall y \in \mathbb{R}^d.$$

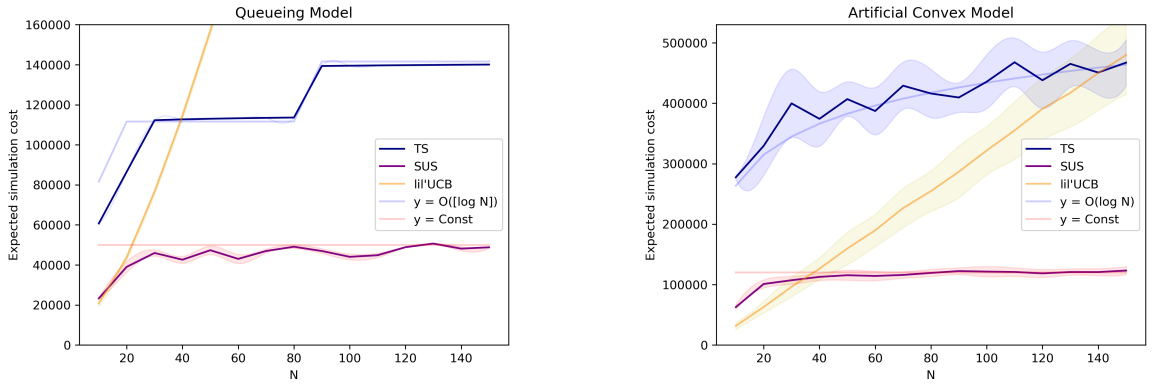
Then, Murota (2003) has proved that

$$h(y) := f \circ g(y) = f(y_1, y_2 - y_1, \ldots, y_d - y_{d-1})$$

is a $L^\natural$-convex function on the $L^\natural$-convex set

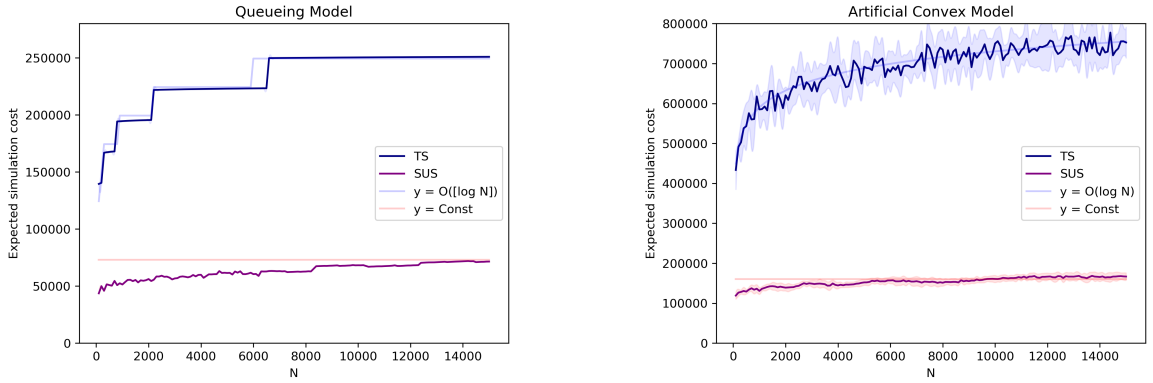
$$\mathcal{Y} := \{y \in [Nd]^d \mid y_1 \in [N],\ y_{i+1} - y_i \in [N],\ i = 1, \ldots, d-1\}.$$

The optimization problem (10) has the trivial solution $x_1 = \cdots = x_d = N$. However, in reality, it is also necessary to keep the staffing cost low. Therefore, we add the staffing cost term $R(x_1, \ldots, x_d) :=$



(a) (b)

Figure 3 The expected simulation cost of TS, SUS and lil'UCB algorithms in the one-dimensional case. **(a)** Optimal allocation problem. **(b)** One-dimensional separable convex function minimization.



(a) (b)

Figure 4 The expected simulation cost of TS and SUS algorithms for problems with a larger scale. **(a)** Optimal allocation problem. **(b)** One-dimensional separable convex function minimization.

$C/d \cdot \sum_{i=1}^{d} x_i = C/d \cdot y_d$ to the objective function, where C is a positive constant. The optimization problem can be written as

$$\min_{y \in \mathcal{Y}} h(y) + C/d \cdot y_d. \tag{11}$$

The proposed algorithms can be extended to this problem by considering the Lovász extension $\tilde{h}(y)$ on the set

$$\tilde{\mathcal{Y}} := \{y \in [1, Nd]^d \mid y_1 \in [1, N],\ y_{i+1} - y_i \in [1, N],\ i = 1, \ldots, d-1\}.$$

5.4. Numerical Results: Tri-section Sampling Algorithm and Shrinking Uniform Sampling Algorithm

We first compare the performance of the TS algorithm and the SUS algorithm on the optimal allocation problem in Section 5.1 and the closed-form convex function minimization problem in

Section 5.2. As a comparison to the existing algorithms, we also implement the state-of-the-art algorithm for the best arm identification problem, namely the lil'UCB algorithm (Jamieson et al. 2014). The best arm identification problem is equivalent to problem (1) without any convexity structure. We consider problems with dimension $d = 1$ and scale $N \in \{10, 20, \ldots, 150\}$. The expected simulation cost is computed by averaging 400 independent solving processes. For the optimal allocation problem, we set the optimality parameters for the PGS guarantee as $\epsilon = 1$ and $\delta = 10^{-6}$. An upper bound on the variance is estimated as $\sigma^2 = 10$. For the convex function minimization problem, we generate each c_i from the uniform distribution on $[0.75, 1.25]$ and x_i^* from the discrete uniform distribution on $\{1, 2, \ldots, \lfloor 0.3N \rfloor\}$. The optimality parameters are chosen as $\epsilon = 0.2$ and $\delta = 10^{-6}$ and the variance is set to be $\sigma^2 = 1$.

It is observed that both algorithms satisfy the given PGS guarantee on the synthetic convex function minimization problem, namely, the ϵ-optimality is satisfied for all implementations. We then plot the estimated expected simulation costs in Figure 3. For the optimal allocation problem, the expected simulation costs of the TS and SUS algorithms approximately have $O(\lceil \log N \rceil)$ and almost $O(1)$ dependence on the scale N, respectively. The expected simulation cost of the SUS algorithm is almost independent of N and this verifies our theoretical analysis. For the synthetic convex function minimization problem, same as the queueing example, the estimated expected simulation costs of the TS algorithm and SUS algorithm have $O(\log N)$ and almost $O(1)$ dependence on N, respectively. This again verifies our theoretical analysis. Moreover, both algorithms outperform the lil'UCB algorithm, when N is large. The numerical results show that our proposed algorithms can efficiently solve large-scale one-dimensional convex problems.

Furthermore, we also compare the TS and the SUS algorithms on problems with a larger scale. Specifically, we consider both problems with $N \in \{100, 200, \ldots, 15000\}$ under the same setting. The results are plotted in Figure 4 and we can see that the outcomes of the algorithms comply with our theoretical results.

5.5. Numerical Results: Subgradient Descent and Localization Methods

We next compare the performances of the truncated stochastic subgradient descent algorithm (Zhang et al. 2020) and stochastic localization methods proposed in this work. We first consider the separable convex function minimization problem, where we can compute the optimality gap and verify the ϵ-optimality. The dimension and scale of the separable convex model are chosen as $d \in \{2, 6, 10, 15\}$ and $N \in \{50, 500, 5000\}$. The optimality guarantee parameters are chosen as $\epsilon = d$ and $\delta = 10^{-6}$, respectively. The empirical choice of ϵ ensures that any ϵ-optimal solution x^0 satisfies $\|x^0 - x^*\|_1 \leq d^{5/6} \ll N$. We compute the average simulation cost of 100 independently generated models to estimate the expected simulation cost. Moreover, early stopping conditions are designed

to terminate algorithms early when little progress is made at any iteration. For the subgradient descent method, we maintain the empirical mean of stochastic objective function values up to the current iteration and terminate the algorithm if the empirical mean does not decrease by $O(\epsilon/\sqrt{N})$ after $O[d\epsilon^{-2}\log(1/\delta)]$ consecutive iterations. For stochastic cutting-plane methods, we terminate the algorithm if the empirical mean of the objective function of the last 5 iterations does not decrease by ϵ/d. For the dimension reduction method, we terminate the algorithm early if the polytope is empty. Furthermore, we have observed that using $(N\epsilon/4, \delta/4)$-$\mathcal{SO}$ oracles in localization methods is sufficient for producing high-probability guarantees on this example.

We summarize the results in Table 1. We define the coverage rate to be the percentage of implementations that produce an ϵ-optimal solution. The coverage rates of the algorithms are all equal to 100% and thus, the PGS guarantee is likely to be satisfied by all of the algorithms. We note that, similar to the one-dimensional case, the standard deviation of the simulation cost is smaller than 10% of the estimated simulation cost in all settings. The performances of localization methods are better than the subgradient descent algorithm in all settings especially for the large-scale instances. The simulation cost of the random walk-based cutting-plane method is better than the Vaidya's cutting-plane method, which may be a result of the extra $\log(d)$ term in the simulation cost; see the discussion in Remark 4. The dimension reduction method has the best performance on examples with $N = 500, 5000$ and has the advantage of not requiring any knowledge about the Lipschitz constant. From the experimental results, we can see that the empirical performances of proposed algorithms are sometimes better than their theoretical guarantees. One possible explanation for the better empirical performance is that during the implementation of the stochastic localization methods, the diameter of the feasible set (i.e., the set $\mathcal{S}$ in Algorithms 4 and 5) will decrease and become much smaller than N after a few iterations. In contrast, for the theoretical analysis, we need to consider the worst case and assume that the diameter is still N for the shrunken set. Therefore, the number of simulations to generate an accurate stochastic separation oracle is overestimated in Lemma 5 and the simulation costs of the stochastic localization algorithms have a better dependence on the problem scale N in practice.

We then consider the multi-dimensional resource allocation problem. We first fix the dimension (number of time slots) to be $d = 4$ and compare the performance with the scale $N \in \{10, 20, 30, 40, 50\}$, and we then fix the scale to be $N = 10$ and compare the performance with the dimension $d \in \{4, 8, 12, 16, 20, 24\}$. The parameters of the problem are chosen as $\lambda = 1$ and $C = 10$, and the optimality guarantee parameters are $\epsilon = N/2$ and $\delta = 10^{-6}$. We also compare the algorithms with a smaller precision parameter $\epsilon = N/10 + 1$ in Section EC.8.3. An upper bound on the variance is estimated as $\sigma^2 = 30\sqrt{N}$. For each problem setup, we average the results of 10 independent implementations to estimate the expected simulation cost and the objective value of

Table 1 Simulation cost of different algorithms on separable convex functions.

Params.		**Search Methods**	**Localization Methods (this work)**		
		SubGD	Vaidya's	Random Walk	Dim Reduction
d	N	Cost	Cost	Cost	Cost
2	50	1.08e3	2.74e2	1.66e2	**1.56e2**
2	500	2.54e4	6.54e2	2.32e2	**2.08e2**
2	5000	3.97e5	1.13e4	5.29e2	**4.66e2**
6	50	5.00e3	4.13e2	**3.36e2**	4.05e2
6	500	4.75e4	1.34e3	1.65e3	**6.45e2**
6	5000	2.72e6	8.15e4	4.75e3	**8.25e2**
10	50	8.46e3	7.98e2	**7.70e2**	8.34e2
10	500	6.32e4	6.57e3	2.16e3	**1.48e3**
10	5000	7.76e6	2.42e5	8.03e3	**2.02e3**
15	50	1.23e4	1.50e3	**1.91e3**	2.18e3
15	500	2.83e5	2.66e4	1.06e4	**3.19e3**
15	5000	1.85e7	1.96e6	1.55e5	**4.85e3**

Table 2 Simulation cost and objective value of different algorithms on the resource allocation problem.

Params.		**Search Methods**		**Localization Methods (this work)**					
		SubGD		Vaidya's		Random Walk		Dim Reduction	
d	N	Cost	Obj.	Cost	Obj.	Cost	Obj.	Cost	Obj.
4	10	3.06e5	2.13e1	9.89e4	2.19e1	6.92e4	2.47e1	**2.42e4**	2.40e1
4	20	1.08e5	3.41e1	3.64e4	3.42e1	2.45e4	3.73e1	**1.40e4**	3.44e1
4	30	7.79e4	4.59e1	1.94e4	4.65e1	1.33e4	5.10e1	**9.21e3**	4.59e1
4	40	5.06e4	5.73e1	1.24e4	5.86e1	8.68e3	6.35e1	**6.31e3**	5.75e1
4	50	4.50e4	6.91e1	9.24e3	6.98e1	6.22e3	7.49e1	**4.03e3**	6.67e1
8	10	1.20e6	2.01e1	7.27e5	2.12e1	5.53e5	2.17e1	**1.48e5**	2.12e1
12	10	2.69e6	1.90e1	2.49e6	2.07e1	1.86e6	2.13e1	**6.10e5**	2.01e1
16	10	4.78e6	1.83e1	6.64e6	2.02e1	4.43e6	2.04e1	**1.59e6**	1.91e1
20	10	7.45e6	1.78e1	1.38e7	2.01e1	8.65e6	2.04e1	**3.21e6**	1.81e1
24	10	1.43e7	1.71e1	2.42e7	1.99e1	1.49e7	2.04e1	**8.54e6**	1.76e1

the returned solution. The results are summarized in Table 2. Similarly, the standard deviation of the simulation cost is smaller than 10% of the estimated simulation cost in all settings. It is observed that the dimension reduction method achieves the best performance in all cases, although its simulation costs have a faster growth rate than other methods. The stochastic cutting-plane methods also outperform the subgradient descent algorithm when the dimension is 4. The truncated stochastic subgradient descent method returns the smallest objective values except the case when $(d, N) = (4, 50)$, and the objective values returned by other algorithms are not much larger than the truncated stochastic subgradient descent method. This is possible since we are searching for ϵ-optimal solutions and an optimality gap smaller than $\epsilon = N/2$ is acceptable.

In summary, based on the results from numerical results, the SUS algorithm and the dimension reduction method provide a more efficient choice for large-scale convex discrete optimization via simulation problems, and they have the advantage that no prior information about the objective function is required except the $L^\natural$-convexity.

Table 3 Upper bounds on the expected simulation cost for algorithms that achieve the PGS guarantee. Here, M is an absolute constant. All constants except $d, N, \epsilon, \delta, c, M$ are omitted in the $\tilde{O}(\cdot)$ notation. In comparison, the expected simulation cost without convexity is $O(N^d \epsilon^{-2} \log(1/\delta))$.

Algorithms	**Expected Simulation Cost**
Tri-section Sampling (Section 3.1)	$\tilde{O}(\log(N)\epsilon^{-2}\log(1/\delta))$
Shrinking Uniform Sampling (Section 3.2)	$\tilde{O}(\epsilon^{-2}\log(1/\delta))$ (best achievable performance)
Subgradient-based (Zhang et al. 2020)	$\tilde{O}(d^2N^2L^2\epsilon^{-2}\log(1/\delta))$
Stochastic Cutting-plane (Section 4.2)	$\tilde{O}(d^3N^2\epsilon^{-2}\log(dNL/\epsilon)\log(1/\delta))$
Dimension Reduction (Section 4.3)	$\tilde{O}(d^3N^2(d+\log(N))\epsilon^{-2}\log(1/\delta))$
Shrinking Uniform Sampling (Section EC.3)	$\tilde{O}(M^d\epsilon^{-2}\log(1/\delta))$

6. Conclusion

In this paper, algorithms based on the idea of localization are proposed for large-scale convex discrete optimization via simulation problems. The simulation-optimization algorithms are theoretically guaranteed to identify a solution whose corresponding objective value is close to the optimal objective value up to a given precision with high probability. Moreover, the efficiency of the developed algorithms is evaluated by obtaining upper bounds on the expected simulation cost. We summarize the performances of our algorithms in Table 3. Specifically, in the one-dimensional case, we propose the SUS method, which has an expected simulation cost as $O[\epsilon^{-2}(\log(N)+\log(1/\delta))]$, which attains the best achievable performance under the asymptotic criterion (Kaufmann et al. 2016), i.e., when $\delta \to 0$. For the multi-dimensional case, we combine the idea of localization with subgradient information. The dimension reduction algorithm is designed using a new framework to extend deterministic cutting-plane methods. The expected simulation cost is proven to be upper bounded by a constant that is independent of the Lipschitz constant. In addition, the dimension reduction algorithm does not require prior knowledge about the Lipschitz constant. Finally, an adaptive algorithm (described in EC.7) is designed to avoid the requirement that the variance of the noise should be estimated a priori. Numerical results on both synthetic and queueing models demonstrate that the proposed algorithms have better performances compared to benchmark methods especially when the problem scale is large. In summary, the stochastic localization algorithms are preferred when either (i) the problem scale is large or (ii) the Lipschitz constant is large or difficult to estimate. On the other hand, if the problem scale is moderate but the dimension is high, the subgradient-based search methods are preferred.

References

Agarwal A, Foster DP, Hsu DJ, Kakade SM, Rakhlin A (2011) Stochastic convex optimization with bandit feedback. Advances in Neural Information Processing Systems, 1035–1043.

Alman J, Williams VV (2020) A refined laser method and faster matrix multiplication. arXiv preprint arXiv:2010.05846 .

Altman E, Gaujal B, Hordijk A (2003) Discrete-event control of stochastic networks: Multimodularity and regularity (springer).

Bechhofer RE (1954) A single-sample multiple decision procedure for ranking means of normal populations with known variances. The Annals of Mathematical Statistics 16–39.

Bertsimas D, Vempala S (2004) Solving convex programs by random walks. Journal of the ACM (JACM) 51(4):540–556.

Burnetas AN, Katehakis MN (1996) Optimal adaptive policies for sequential allocation problems. Advances in Applied Mathematics 17(2):122–142.

Chen L, Gupta A, Li J (2016) Pure exploration of multi-armed bandit under matroid constraints. Conference on Learning Theory, 647–669.

Chen X, Li M (2020) Discrete convex analysis and its applications in operations: A survey. Production and Operations Management .

Dyer M, Proll L (1977) Note—on the validity of marginal analysis for allocating servers in m/m/c queues. Management Science 23(9):1019–1022.

Eckman DJ, Henderson SG (2018) Fixed-confidence, fixed-tolerance guarantees for selection-of-the-best procedures. Technical report, Working paper, Cornell University, School of Operations Research and ....

Eckman DJ, Plumlee M, Nelson BL (2020) Plausible screening using functional properties for simulations with large solution spaces, working paper.

Eckman DJ, Plumlee M, Nelson BL (2021) Flat chance! using stochastic gradient estimators to assess plausible optimality for convex functions. 2021 Winter Simulation Conference (WSC), 1–18 (IEEE).

Even-Dar E, Mannor S, Mansour Y (2002) Pac bounds for multi-armed bandit and markov decision processes. International Conference on Computational Learning Theory, 255–270 (Springer).

Freund D, Henderson SG, Shmoys DB (2017) Minimizing multimodular functions and allocating capacity in bike-sharing systems. International Conference on Integer Programming and Combinatorial Optimization, 186–198 (Springer).

Fujishige S (2005) Submodular functions and optimization (Elsevier).

Gong X, Chao X (2013) Optimal control policy for capacitated inventory systems with remanufacturing. Operations Research 61(3):603–611.

Günlük O (1999) A branch-and-cut algorithm for capacitated network design problems. Mathematical programming 86(1):17–39.

Hong LJ, Fan W, Luo J (2020) Review on ranking and selection: A new perspective. arXiv preprint arXiv:2008.00249 .

Hong LJ, Nelson BL (2006) Discrete optimization via simulation using compass. Operations Research 54(1):115–129.

Hong LJ, Nelson BL, Xu J (2015) Discrete optimization via simulation. Handbook of simulation optimization, 9–44 (Springer).

Huh WT, Janakiraman G (2010) On the optimal policy structure in serial inventory systems with lost sales. Operations Research 58(2):486–491.

Jamieson K, Malloy M, Nowak R, Bubeck S (2014) lil'ucb: An optimal exploration algorithm for multi-armed bandits. Conference on Learning Theory, 423–439.

Jian N, Freund D, Wiberg HM, Henderson SG (2016) Simulation optimization for a large-scale bike-sharing system. 2016 Winter Simulation Conference (WSC), 602–613 (IEEE).

Jiang H (2021) Minimizing convex functions with integral minimizers 976–985.

Jiang H, Lee YT, Song Z, Wong SCw (2020) An improved cutting plane method for convex optimization, convex-concave games, and its applications. Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, 944–953.

Karnin Z, Koren T, Somekh O (2013) Almost optimal exploration in multi-armed bandits. International Conference on Machine Learning, 1238–1246.

Kaufmann E, Cappé O, Garivier A (2016) On the complexity of best-arm identification in multi-armed bandit models. The Journal of Machine Learning Research 17(1):1–42.

Lai TL, Robbins H (1985) Asymptotically efficient adaptive allocation rules. Advances in applied mathematics 6(1):4–22.

Lee JC, Valiant P (2020) Optimal sub-gaussian mean estimation in $\mathbb{R}$. arXiv preprint arXiv:2011.08384 .

Lee YT, Sidford A, Wong SCw (2015) A faster cutting plane method and its implications for combinatorial and convex optimization. 2015 IEEE 56th Annual Symposium on Foundations of Computer Science, 1049–1065 (IEEE).

Lenstra AK, Lenstra HW, Lovász L (1982) Factoring polynomials with rational coefficients. Mathematische annalen 261(ARTICLE):515–534.

Lerasle M (2019) Selected topics on robust statistical learning theory. arXiv preprint arXiv:1908.10761 .

Liang T, Narayanan H, Rakhlin A (2014) On zeroth-order stochastic convex optimization via random walks. arXiv preprint arXiv:1402.2667 .

Lovász L (1983) Submodular functions and convexity. Mathematical programming the state of the art, 235–257 (Springer).

Luo J, Hong LJ, Nelson BL, Wu Y (2015) Fully sequential procedures for large-scale ranking-and-selection problems in parallel computing environments. Operations Research 63(5):1177–1194.

Ma S, Henderson SG (2017) An efficient fully sequential selection procedure guaranteeing probably approximately correct selection. 2017 Winter Simulation Conference (WSC), 2225–2236 (IEEE).

Ma S, Henderson SG (2019) Predicting the simulation budget in ranking and selection procedures. ACM Transactions on Modeling and Computer Simulation 29(3):Article 14, 1–25.

Murota K (2003) Discrete convex analysis. Society for Industrial and Applied Mathematics (Citeseer).

Nelson BL (2010) Optimization via simulation over discrete decision variables. Risk and Optimization in an Uncertain World, 193–207 (Informs).

Ni EC, Ciocan DF, Henderson SG, Hunter SR (2017) Efficient ranking and selection in high performance computing environments. Operations Research 65(3):821–836.

Ostrovsky E, Sirota L (2014) Exact value for subgaussian norm of centered indicator random variable. arXiv preprint arXiv:1405.6749 .

Pang Z, Chen FY, Feng Y (2012) A note on the structure of joint inventory-pricing control with leadtimes. Operations Research 60(3):581–587.

Ragavan PK, Hunter SR, Pasupathy R, Taaffe MR (2022) Adaptive sampling line search for local stochastic optimization with integer variables. Mathematical Programming 196(1-2):775–804.

Shaked M, Shanthikumar JG (1988) Stochastic convexity and its applications. Advances in Applied Probability 20(2):427–446.

Shi L, et al. (2000) Nested partitions method for stochastic optimization. Methodology and Computing in Applied probability 2(3):271–291.

Singhvi D, Singhvi S, Frazier PI, Henderson SG, O'Mahony E, Shmoys DB, Woodard DB (2015) Predicting bike usage for new york city's bike sharing system. AAAI Workshop: Computational Sustainability (Citeseer).

Sun L, Hong LJ, Hu Z (2014) Balancing exploitation and exploration in discrete optimization via simulation through a gaussian process-based search. Operations Research 62(6):1416–1438.

Vaidya PM (1996) A new algorithm for minimizing convex functions over convex sets. Mathematical programming 73(3):291–341.

Wainwright MJ (2019) High-dimensional statistics: A non-asymptotic viewpoint, volume 48 (Cambridge University Press).

Wang H, Pasupathy R, Schmeiser BW (2013) Integer-ordered simulation optimization using r-spline: Retrospective search with piecewise-linear interpolation and neighborhood enumeration. ACM Transactions on Modeling and Computer Simulation (TOMACS) 23(3):1–24.

Wang T, Xu J, Hu JQ, Chen CH (2021) Optimal computing budget allocation for regression with gradient information. Automatica 134:109927.

Wolff RW, Wang CL (2002) On the convexity of loss probabilities. Journal of applied probability 402–406.

Xu J, Nelson BL, Hong JL (2010) Industrial strength compass: A comprehensive algorithm and software for optimization via simulation. ACM Transactions on Modeling and Computer Simulation (TOMACS) 20(1):1–29.

Xu WL, Nelson BL (2013) Empirical stochastic branch-and-bound for optimization via simulation. Iie Transactions 45(7):685–698.

Xu Y, Lin Q, Yang T (2016) Accelerated stochastic subgradient methods under local error bound condition. arXiv preprint arXiv:1607.01027 .

Zhang H, Zheng Z, Lavaei J (2020) Discrete convex simulation optimization. arXiv preprint arXiv:2010.16250 .

Zhong Y, Hong LJ (2019) Knockout-tournament procedures for large-scale ranking and selection in parallel computing environments. accepted .

Zipkin P (2008) On the structure of lost-sales inventory models. Operations research 56(4):937–944.

Proofs of Statements

EC.1. Algorithms and Complexity Analysis for the PCS-IZ Guarantee

In this section, we provide modified simulation-optimization algorithms for the PCS-IZ guarantee. We assume that the objective value of any sub-optimal choice of decision variables is at least c larger than the optimal objective value, where the indifference zone parameter $c > 0$ is known a priori. Let $\mathcal{MC}_c$ be the set that includes all convex models with the indifference zone parameter c. Then, the expected simulation cost for the PCS-IZ criterion is denoted as $T(\delta, \mathcal{MC}_c) := T((c,\delta)\text{-}PCS\text{-}IZ, \mathcal{MC}_c)$.

EC.1.1. Modified Tri-section Sampling Algorithm for the PCS-IZ Guarantee

We first consider the one-dimensional case. When the prior information about the indifference zone parameter c is available, we can modify the TS algorithm to achieve a better simulation cost. The modified algorithm also consists of two parts: the shrinkage of intervals and a sub-problem with at most 3 points. The improvement is achieved by a weaker condition for the comparison of objective values at two 3-quantiles. We give the modified algorithm in Algorithm 6 and omit those lines that are the same as Algorithm 1.

Algorithm 6 Tri-section sampling algorithm for the PCS-IZ guarantee

Input: Model $\mathcal{X} = [N], (\mathsf{Y}, \mathcal{B}_{\mathsf{Y}}), F(x, \xi_x)$, optimality guarantee parameter δ, indifference zone parameter c.

Output: An (c,δ)-PCS-IZ solution x^* to problem (1).

1: Set upper and lower bounds of current interval $x_L \leftarrow 1, x_U \leftarrow N$.

2: Set maximal number of comparisons $T_{max} \leftarrow \log_{1.5}(N) + 2$.

3: **while** $x_U - x_L > 2$ **do** ▷ Iterate until there are at most 3 decisions.

...

5: Simulate n independent copies of $F(q_{1/3}, \xi_{1/3})$ and $F(q_{2/3}, \xi_{2/3})$, where n is the smallest integer such that $h[n, \sigma, 1 - \delta/(2T_{max})] \le (q_{2/3} - q_{1/3}) \cdot c/5$.

...

14: **end while**

15: Simulate $\tilde{n}$ independent copies of $F(x, \xi_x)$ for all $x \in \{x_L, \ldots, x_U\}$, where $\tilde{n}$ is the smallest integer such that $h[\tilde{n}, \sigma, 1 - \delta/(2T_{max})] \le c/3$. ▷ Now $x_U - x_L \le 2$.

16: Return the point in $\{x_L, \ldots, x_U\}$ with minimal empirical mean.

The following theorem proves the correctness and the expected simulation cost of the modified TS algorithm.

THEOREM EC.1. *Suppose that Assumptions 1-3 hold. The modified TS algorithm is a* $[(c,\delta)$*-PCS-IZ*$,\mathcal{MC}_c]$*-algorithm. Furthermore, we have*

$$T(\delta,\mathcal{MC}_c)=O\left[\frac{1}{c^2}\log\left(\frac{\log(N)}{\delta}\right)+\log(N)\right]=\tilde{O}\left[\frac{1}{c^2}\log\left(\frac{1}{\delta}\right)\right].$$

Proof of Theorem EC.1. The proof of is provided in EC.1.4. □

By Theorem EC.1, the expected simulation cost for the PCS-IZ guarantee is asymptotically independent of the number of points N, when the failing probability δ is sufficiently small. If the SUS algorithm is used for the PCS-IZ guarantee, the algorithm also achieves an $\tilde{O}(c^{-2}\log(1/\delta))$ expected simulation cost by setting the optimality parameter $\epsilon=c/2$. This is because the objective values of sub-optimal solutions are larger than that of the optimal solution by at least c and because the solution satisfying the $(c/2,\delta)$-PGS guarantee also satisfies the (c,δ)-PCS-IZ guarantee. Hence, for both the modified TS algorithm and the SUS algorithm, the asymptotic simulation cost has an upper bound that is independent of N. However, we note that the space complexity of the modified TS algorithm is only $\tilde{O}(\log(N))$, whereas the SUS algorithm requires $O(N)$ memory space. Therefore, the modified TS algorithm is preferred for the PCS-IZ guarantee.

EC.1.2. Modified Stochastic Cutting-plane Methods for the PCS-IZ Guarantee

In the multi-dimensional case, we develop modified stochastic cutting-plane methods for the PCS-IZ guarantee. Using the same adaptive acceleration scheme as in Zhang et al. (2020), the indifference zone parameter can help reduce the dependence of the simulation cost on the problem scale N. We give the pseudo-code of the accelerated stochastic cutting-plane method in Algorithm 7.

Algorithm 7 Stochastic cutting-plane method for the PCS-IZ guarantee

Input: Model $\mathcal{X},(\mathsf{Y},\mathcal{B}_{\mathsf{Y}}),F(x,\xi_x)$, optimality guarantee parameter δ, indifference zone parameter c, Lipschitz constant L, (ϵ,δ)-$\mathcal{SO}$ oracle $\hat{g}$.

Output: An (c,δ)-PCS-IZ solution x^* to problem (1).

1: Set the initial guarantee $\epsilon_0\leftarrow cN/4$.
2: Set the number of epochs $E\leftarrow\lceil\log_2(N)\rceil+1$.
3: Set the initial searching space $\mathcal{Y}_0\leftarrow[1,N]^d$.
4: **for** $e=0,\ldots,E-1$ **do**
5: Use Algorithm 4 to get an $(\epsilon_e,\delta/(2E))$-PGS solution x_e in $\mathcal{Y}_e$.
6: Update guarantee $\epsilon_{e+1}\leftarrow\epsilon_e/2$.
7: Update the searching space $\mathcal{Y}_{e+1}\leftarrow\mathcal{N}(x_e,2^{-e-2}N)$.

8: **end for**

9: Round x_{E-1} to an integral point by Algorithm 3.

We can prove the correctness and estimate the expected simulation cost of the accelerated algorithm in the same way as Theorem 8 in Zhang et al. (2020). Thus, we omit the proof.

THEOREM EC.2. *Suppose that Assumptions 1-4 hold. The accelerated stochastic cutting-plane method returns a (c,δ)-PCS-IZ solution and we have*

$$\begin{aligned} T(c,\delta,\mathcal{MC}_c) &= O\left[\frac{d^3\log(N)}{\epsilon^2}\log(\frac{dLN}{\epsilon})\log\left(\frac{1}{\delta}\right)+d^2\log(N)\log(\frac{dLN}{\epsilon})\right] \\ &= \tilde{O}\left[\frac{d^3\log(N)}{\epsilon^2}\log(\frac{dLN}{\epsilon})\log\left(\frac{1}{\delta}\right)\right]. \end{aligned}$$

By substituting Algorithm 4 with Algorithm 5 in the above algorithm, the acceleration scheme can be applied to Algorithm 5 to reduce the number of required simulation runs when the indifference zone parameter c is known. We give the reduced expected simulation cost for achieving the PCS-IZ guarantee and omit the proof.

THEOREM EC.3. *Suppose that Assumptions 1-3 hold. The accelerated dimension reduction method returns an (c,δ)-PCS-IZ solution and we have*

$$\begin{aligned} T(c,\delta,\mathcal{MC}_c) &= O\left[\frac{d^3\log(N)(d+\log(N))}{\epsilon^2}\log\left(\frac{1}{\delta}\right)+d^2\log(N)(d+\log(N))\right] \\ &= \tilde{O}\left[\frac{d^3\log(N)(d+\log(N))}{\epsilon^2}\log\left(\frac{1}{\delta}\right)\right]. \end{aligned}$$

EC.1.3. Lower Bound on Expected Simulation Cost

In this subsection, we derive the lower bounds on the expected simulation costs for all of the simulation-optimization algorithms that satisfy certain optimality guarantee for general convex problems. In the proof for the lower bound result, we construct two convex models that have similar distributions at each point but have distinct optimal solutions. Then, the information-theoretical inequality in Kaufmann et al. (2016) can be used to provide a lower bound on the simulation costs for all algorithms.

We first present the results in Kaufmann et al. (2016) for completeness. Given a simulation-optimization algorithm and a model $\mathbb{M}$, we define random variable $N_x(\tau)$ to be the number of times that $F(x,\xi_x)$ is sampled when the algorithm terminates, where τ is the stopping time of the algorithm. Then, it follows from the definition that

$$\mathbb{E}_M[\tau] = \sum_{x\in\mathcal{X}} \mathbb{E}_{\mathbb{M}}\left[N_x(\tau)\right],$$

where $\mathbb{E}_{\mathbb{M}}$ is the expectation when the model $\mathbb{M}$ is given. Similarly, we can define $\mathbb{P}_{\mathbb{M}}$ as the probability when the model $\mathbb{M}$ is given. We denote the filtration up to the stopping time τ as $\mathcal{F}_\tau$. The following lemma is proved in Kaufmann et al. (2016) and is the major tool for deriving lower bounds in this paper.

LEMMA EC.1 **(Kaufmann et al. (2016))**. *For any two models $\mathbb{M}_1, \mathbb{M}_2$ and any event $\mathcal{E} \in \mathcal{F}_\tau$, we have*

$$\sum_{x\in\mathcal{X}} \mathbb{E}_{\mathbb{M}_1}[N_x(\tau)]\,\mathrm{KL}(\nu_{1,x},\nu_{2,x}) \geq d(\mathbb{P}_{\mathbb{M}_1}(\mathcal{E}),\mathbb{P}_{\mathbb{M}_2}(\mathcal{E})), \tag{EC.1}$$

where $d(x,y) := x\log(x/y)+(1-x)\log((1-x)/(1-y))$, $\mathrm{KL}(\cdot,\cdot)$ is the Kullback–Leibler divergence (KL divergence), and $\nu_{k,x}$ is the distribution of model $\mathbb{M}_k$ at point x for $k=1,2$.

We first give a lower bound for the PCS-IZ guarantee.

THEOREM EC.4. *Suppose that Assumptions 1-3 hold. We have*

$$T(\delta,\mathcal{MC}_c) \geq \Theta\left[\frac{1}{c^2}\log\left(\frac{1}{\delta}\right)\right].$$

Proof of Theorem EC.4. The proof is provided in EC.1.5. □

The lower bound on $T(\epsilon,\delta,\mathcal{MC})$, i.e., the expected simulation cost for achieving the PGS guarantee, can be derived in a similar way by substituting c with 2ϵ in the construction of two models.

COROLLARY EC.1. *Suppose that Assumptions 1-3 hold. We have*

$$T(\epsilon,\delta,\mathcal{MC}) \geq \Theta\left[\frac{1}{\epsilon^2}\log\left(\frac{1}{\delta}\right)\right].$$

Combining with the upper bounds derived in Sections 3.1, 3.2 and EC.1, we conclude that the TS algorithm and the SUS algorithm are optimal up to a constant for the PCS-IZ guarantee in the asymptotic regime $\delta \ll 1$. Considering the space complexity, the TS algorithm is preferred for the PCS-IZ guarantee, while for the PGS guarantee we need to consider the trade-off between the simulation cost and the space complexity when choosing the best algorithm.

EC.1.4. Proof of Theorem EC.1

We first estimate the simulation cost of each iteration and the sub-problem.

LEMMA EC.2. *Suppose that Assumptions 1-3 hold. The simulation cost for each iteration of Algorithm 6 is at most $100\sigma^2c^{-2}(q_{2/3}-q_{1/3})^{-2}\log[4T_{max}/\delta]$, where $T_{max} := \log_{1.5}(N)+2$. The simulation cost of the sub-problem is at most $54\sigma^2c^{-2}\log[4T_{max}/\delta]$.*

Proof. The proof is similar to the proof of Lemma EC.5 and we only give a sketch of the proof. By the definition of $h(n,\sigma,\alpha)$, simulating

$$n=\frac{50\sigma^2}{(q_{2/3}-q_{1/3})^2c^2}\log\left(\frac{4T_{max}}{\delta}\right)$$

times on quantiles $q_{1/3}$ and $q_{2/3}$ is enough to ensure that the confidence half-width is at most $(q_{2/3}-q_{1/3})\cdot c/5$. It implies that the last condition in Line 8 is satisfied and the simulation cost of each iteration is at most $100\sigma^2c^{-2}(q_{2/3}-q_{1/3})^{-2}\log[4T_{max}/\delta]$. For the last sub-problem, simulating

$$\tilde{n}=\frac{18\sigma^2}{c^2}\log\left(\frac{4T_{max}}{\delta}\right)$$

times for each point is enough to ensure that the confidence half-width is at most $c/3$. Since there are at most 3 points in the sub-problem, the simulation cost for the sub-problem is at most $54\sigma^2c^{-2}\log[4T_{max}/\delta]$. □

Using Lemma EC.2, we can estimate the total simulation cost of Algorithm 6.

Lemma EC.3. *Suppose that Assumptions 1-3 hold. The expected simulation cost of Algorithm 6 is bounded by*

$$\frac{459\sigma^2}{c^2}\log\left(\frac{4T_{max}}{\delta}\right)=O\left[\frac{1}{c^2}\log\left(\frac{1}{\delta}\right)\right],$$

where $T_{max}:=\log_{1.5}(N)+2$.

Proof. We denote the upper bound and the lower bound at the beginning of the k-th iteration as x_{U_k} and x_{L_k}, respectively. By Lemma EC.2, the simulation cost for the k-th iteration is at most $100\sigma^2c^{-2}(q^k_{2/3}-q^k_{1/3})^{-2}\log[4T_{max}/\delta]$, where $q^k_{1/3}$ and $q^k_{2/3}$ are the 3-quantiles for the k-th iteration. By the definition of 3-quantiles, it follows that $q^k_{2/3}-q^k_{1/3}\geq(x_{U_k}-x_{L_k})/3$ and therefore

$$\frac{100\sigma^2}{(q^k_{2/3}-q^k_{1/3})^2c^2}\log\left(\frac{4T_{max}}{\delta}\right)\leq\frac{900\sigma^2}{(x_{U_k}-x_{L_k})^2c^2}\log\left(\frac{4T_{max}}{\delta}\right). \tag{EC.2}$$

Hence, we only need to bound the sum $\sum_{k=1}^{T}(x_{U_k}-x_{L_k})^{-2}$, where T is the number of iterations of Algorithm 6. By inequality (EC.7), we know

$$x_{U_k}-x_{L_k}\geq\frac{3}{2}(x_{U_{k+1}}-x_{L_{k+1}})-1,\quad\forall k\in\{1,2,\ldots,T-1\}.$$

We can rewrite the above inequality as $x_{U_k}-x_{L_k}-2\geq 3/2\cdot(x_{U_{k+1}}-x_{L_{k+1}}-2)$. Since T is the last iteration, it holds that $x_{U_T}-x_{L_T}\geq 4$ and therefore

$$x_{U_k}-x_{L_k}-2\geq\left(\frac{3}{2}\right)^{T-k}(x_{U_T}-x_{L_T}-2)\geq 2\cdot\left(\frac{3}{2}\right)^{T-k}.$$

Summing over $k=1,2,\ldots,T$, we get the bound

$$\sum_{k=1}^{T}(x_{U_k}-x_{L_k})^{-2}\leq\sum_{k=1}^{T}\left(2\cdot\left(\frac{3}{2}\right)^{T-k}+2\right)^{-2}\leq\sum_{k=1}^{T}\frac{1}{4}\cdot\left(\frac{3}{2}\right)^{-2(T-k)}=\frac{9}{20}\left[1-\left(\frac{4}{9}\right)^{T}\right]\leq\frac{9}{20}.$$

Combining with inequality (EC.2), the simulation cost for T iterations is at most

$$\frac{900\sigma^2}{c^2}\log\left(\frac{4T_{max}}{\delta}\right)\cdot\sum_{k=1}^{T}\left(x_{U_k}-x_{L_k}\right)^{-2}\leq\frac{405\sigma^2}{c^2}\log\left(\frac{4T_{max}}{\delta}\right).$$

Considering the simulation cost of the sub-problem, the total simulation cost of Algorithm 6 is at most

$$\frac{405\sigma^2}{c^2}\log\left(\frac{4T_{max}}{\delta}\right)+\frac{54\sigma^2}{c^2}\log\left(\frac{4T_{max}}{\delta}\right)=\frac{459\sigma^2}{c^2}\log\left(\frac{4T_{max}}{\delta}\right).$$

□

Finally, we verify the correctness of Algorithm 6 and get an upper bound on $T(\delta,\mathcal{MC}_c)$.

Proof of Theorem EC.1. Similar to the proof of Theorem 1, we use the induction method to prove that Event-I happens for the k-th iteration with probability at least $1-(k-1)\delta/T_{max}$. For the first iteration, the solution to problem (1) is in $\mathcal{X}=\{1,2,\ldots,N\}$ with probability 1. We assume that the claim is true for the first $k-1$ iterations, and consider the k-th iteration. If one of the first two conditions holds when the current iteration terminates, then, by the same analysis as the proof of Theorem 1, we know that Event-I happens for the k-th iteration with probability at least $1-(k-1)\delta/T_{max}$. Hence, we only need consider the case when only the last condition holds when the current iteration terminates. Since the first two conditions do not hold, we know

$$\left|\hat{F}_n(q_{1/3})-\hat{F}_n(q_{2/3})\right|\leq(q_{2/3}-q_{1/3})\cdot 2c/5. \tag{EC.3}$$

In addition, it holds that

$$\left|f(q_{1/3})-\hat{F}_n(q_{1/3})\right|\leq(q_{2/3}-q_{1/3})\cdot c/5,\quad \left|f(q_{2/3})-\hat{F}_n(q_{2/3})\right|\leq(q_{2/3}-q_{1/3})\cdot c/5$$

with probability at least $1-\delta/T_{max}$. Combining with inequality (EC.3), we know that

$$\left|f(q_{1/3})-f(q_{2/3})\right|\leq(q_{2/3}-q_{1/3})\cdot 4c/5<(q_{2/3}-q_{1/3})\cdot c \tag{EC.4}$$

holds with probability at least $1-\delta/T_{max}$. We assume that the above event and Event-I for the $(k-1)$-th iteration both hold, which has a joint probability of at least $1-\delta/T_{max}-(k-2)\delta/T_{max}=1-(k-1)\delta/T_{max}$. If the solution to problem (1) is not in $\{q_{1/3},\ldots,q_{2/3}\}$, then function $f(x)$ is monotone on $\{q_{1/3},\ldots,q_{2/3}\}$ and

$$\left|f(q_{1/3})-f(q_{2/3})\right|=\sum_{x=q_{1/3}}^{q_{2/3}-1}\left|f(x)-f(x+1)\right|.$$

Since the indifference zone parameter is c and the function $f(x)$ is convex, the function value difference between any two neighbouring points is at least c, which implies that

$$\sum_{x=q_{1/3}}^{q_{2/3}-1}|f(x)-f(x+1)|\geq(q_{2/3})-q_{1/3})\cdot c.$$

However, the above inequality contradicts inequality (EC.4) and thus the solution to problem (1) is in $\{q_{1/3}, \ldots, q_{2/3}\}$. Hence, Event-I happens for the k-th iteration with probability at least $1-(k-1)\delta/T_{max}$.

Suppose that there are T iterations in Algorithm 6. Since the updating rule of intervals is not changed, Lemma EC.4 gives $T \le T_{max}-1$. By the induction method, the solution to problem (1) is in $\{x_{L_{T+1}}, \ldots, x_{U_{k+1}}\}$ with probability at least $1-T\cdot\delta/T_{max} \ge 1-\delta+\delta/T_{max}$. Using the same analysis as Theorem 1, the point returned by the sub-problem is at most $2c/3$ larger than the optimal value with probability at least $1-\delta$. By the assumption that the indifference zone parameter is c, all feasible points have function values at least c larger than the optimal value. This implies that the solution returned by Algorithm 6 is optimal with probability at least $1-\delta+\delta/T_{max}-\delta/T_{max} \ge 1-\delta$ and Algorithm 6 is a $[(c,\delta)\text{-PCS-IZ}, \mathcal{MC}_c]$-algorithm. □

EC.1.5. Proof of Theorem EC.4

Proof of Theorem EC.4. We construct the two models $\mathbb{M}_1, \mathbb{M}_2 \in \mathcal{MC}$ as

$$\nu_{1,x} := \mathcal{N}\left[cx, \sigma^2\right], \quad \nu_{2,x} := \mathcal{N}\left[c\left(|x-2|+2\right), \sigma^2\right], \quad \forall x \in \mathcal{X}.$$

Given a $[(c,\delta)\text{-PCS-IZ}, \mathcal{MC}_c]$-algorithm, the algorithm returns point 1 with probability at least $1-\delta$ when applied to model $\mathbb{M}_1$, and returns point 2 with probability at least $1-\delta$ when applied to model $\mathbb{M}_2$. We choose $\mathcal{E}$ as the event that the algorithm returns point 1 as the solution. Then, we know

$$\mathbb{P}_{\mathbb{M}_1}(\mathcal{E}) \ge 1-\delta, \quad \mathbb{P}_{\mathbb{M}_2}(\mathcal{E}) \le \delta.$$

Using the monotonicity of function $d(x,y)$, we get

$$d(\mathbb{P}_{\mathbb{M}_1}(\mathcal{E}), \mathbb{P}_{\mathbb{M}_2}(\mathcal{E})) \ge d(1-\delta,\delta) \ge \log(1/2.4\delta). \tag{EC.5}$$

Since the distributions $\nu_{1,x}$ and $\nu_{2,x}$ are Gaussian with variance σ^2, the KL divergence can be calculated as

$$\mathrm{KL}(\nu_{1,x}, \nu_{2,x}) = \frac{\left[cx - c\left(|x-2|+2\right)\right]^2}{2\sigma^2} = \begin{cases} 2c^2\sigma^{-2} & \text{if } x=1 \\ 0 & \text{otherwise.} \end{cases}$$

Hence, the summation can be calculated as

$$\sum_{x\in\mathcal{X}} \mathbb{E}_{\mathbb{M}_1}\left[N_x(\tau)\right] \mathrm{KL}(\nu_{1,x}, \nu_{2,x}) = \frac{2c^2}{\sigma^2} \cdot \mathbb{E}_{\mathbb{M}_1}\left[N_1(\tau)\right]. \tag{EC.6}$$

Substituting (EC.5) and (EC.6) into inequality (EC.1), we know

$$\frac{2c^2}{\sigma^2} \cdot \mathbb{E}_{\mathbb{M}_1}\left[N_1(\tau)\right] \ge \log(1/2.4\delta),$$

which implies that

$$\mathbb{E}_{\mathbb{M}_1}[\tau] \ge \mathbb{E}_{\mathbb{M}_1}\left[N_1(\tau)\right] \ge \frac{\sigma^2}{2c^2} \cdot \log\left(\frac{1}{2.4\delta}\right) = \Theta\left[\frac{1}{c^2}\log\left(\frac{1}{\delta}\right)\right].$$

□

EC.2. Proofs in Section 3

EC.2.1. Proof of Theorem 1

We first estimate the simulation cost of Algorithm 1. The following lemma gives an upper bound on the total number of iterations.

LEMMA EC.4. *Suppose that Assumptions 1-3 hold. The number of iterations of Algorithm 1 is at most* $\log_{1.5}(N)+1$.

Proof. If the total number of points N is at most 3, then there is no iteration. In the following proof, we assume $N \geq 4$. We first calculate the shrinkage of interval length after each iteration. We denote the upper and the lower bound at the beginning of the k-th iteration as x_{U_k} and x_{L_k}, respectively. Then, we know there are $n_k := x_{U_k} - x_{L_k} + 1$ points in the k-th iteration and the algorithm starts with $x_{L_1} = 1, x_{U_1} = N$. We define the 3-quantiles $q_{1/3} := \lfloor 2x_{L_k}/3 + x_{U_k}/3 \rfloor$ and $q_{2/3} := \lceil x_{L_k}/3 + 2x_{U_k}/3 \rceil$. By the updating rule, the next interval is

$$[x_{L_k}, q_{2/3}] \quad \text{or} \quad [q_{1/3}, x_{U_k}] \quad \text{or} \quad [q_{1/3}, q_{2/3}].$$

By discussing three cases when $n_k \in 3\mathbb{Z}$, $n_k \in 3\mathbb{Z}+1$ and $n_k \in 3\mathbb{Z}+2$, we know the next interval has at most $2n_k/3+1$ points, i.e.,

$$n_{k+1} \leq 2n_k/3 + 1. \tag{EC.7}$$

Rewriting the inequality, we get the relation $n_{k+1} - 3 \leq 2(n_k - 3)/3$. Combining with the fact that $n_1 = N$, it follows that

$$n_k \leq \left(\frac{2}{3}\right)^{k-1} (N-3) + 3.$$

Suppose Algorithm 1 terminates after T iterations. Then, it holds that $n_T \geq 4$ and $n_{T+1} \leq 3$. Hence, we know

$$4 \leq n_T \leq \left(\frac{2}{3}\right)^{T-1} (N-3) + 3,$$

which implies

$$T \leq \log_{1.5}(N-3) + 1 < \log_{1.5}(N) + 1.$$

□

In the next lemma, we estimate the simulation cost of each iteration.

LEMMA EC.5. *Suppose that Assumptions 1-3 hold. The simulation cost of each iteration of Algorithm 1 is at most* $256\sigma^2\epsilon^{-2}\log(4T_{max}/\delta)$, *where* $T_{max} := \log_{1.5}(N)+2$. *The simulation cost of the sub-problem is at most* $24\sigma^2\epsilon^{-2}\log(4T_{max}/\delta)$.

Proof. We first estimate the simulation cost at each iteration. If we choose $n = n_{\epsilon,\delta}$, the confidence half-width is $\epsilon/4$ and the condition $h[n, \sigma, 1 - \delta/(2T_{max})] \leq \epsilon/8$ is satisfied. Hence, the simulation cost of each iteration is at most $2n_{\epsilon,\delta}$.

For the last sub-problem, we choose

$$\tilde{n} = \tilde{n}_{\epsilon,\delta} := \frac{8\sigma^2}{\epsilon^2} \log\left(\frac{4T_{max}}{\delta}\right)$$

and it follows that

$$h[\tilde{n}_{\epsilon,\delta}, \sigma, 1 - \delta/(2T_{max})] \leq \epsilon/2.$$

Hence, simulating $\tilde{n}_{\epsilon,\delta}$ times on each point is sufficient and the simulation cost is at most $3\tilde{n}_{\epsilon,\delta}$. □

Combining Lemmas EC.4 and EC.5, we get the total simulation cost of Algorithm 1.

LEMMA EC.6. *Suppose that Assumptions 1-3 hold. The expected simulation cost of Algorithm 1 is at most*

$$\frac{256T_{max}\sigma^2}{\epsilon^2} \log\left(\frac{4T_{max}}{\delta}\right) = O\left[\frac{\log(N)}{\epsilon^2} \log\left(\frac{1}{\delta}\right)\right],$$

where $T_{max} := \log_{1.5}(N) + 2$.

Proof. By Lemmas EC.4 and EC.5, the total simulation cost of the first part is at most

$$[\log_{1.5}(N) + 1] \cdot \frac{256\sigma^2}{\epsilon^2} \log\left(\frac{4T_{max}}{\delta}\right) \leq [T_{max} - 1] \cdot \frac{256\sigma^2}{\epsilon^2} \log\left(\frac{4T_{max}}{\delta}\right)$$

and the simulation cost of the second part is at most

$$\frac{24\sigma^2}{\epsilon^2} \log\left(\frac{4T_{max}}{\delta}\right).$$

Combining two parts, we know the total simulation cost is at most

$$[T_{max} - 1] \cdot \frac{256\sigma^2}{\epsilon^2} \log\left(\frac{4T_{max}}{\delta}\right) + \frac{24\sigma^2}{\epsilon^2} \log\left(\frac{4T_{max}}{\delta}\right) \leq \frac{256T_{max}\sigma^2}{\epsilon^2} \log\left(\frac{4T_{max}}{\delta}\right).$$

□

Finally, we verify the correctness of Algorithm 1 and get an upper bound on $T(\epsilon, \delta\mathcal{MC})$.

Proof of Theorem 1. We denote $T_{max} := \log_{1.5}(N) + 2$. We also denote the upper and the lower bound at the beginning of the k-th iteration as x_{U_k} and x_{L_k}, respectively. We use the induction method to prove that, for the k-th iteration, at least one of the following two events happens with probability at least $1 - (k-1) \cdot \delta/T_{max}$:

- **Event-I.** A solution to problem (1) is in $\{x_{L_k}, \ldots, x_{U_k}\}$,
- **Event-II.** For any $x \in \{x_{L_k}, \ldots, x_{U_k}\}$, it holds $f(x) \leq \min_{y \in \mathcal{X}} f(y) + \epsilon$.

When $k=1$, all solutions to problem (1) are in $\mathcal{X}=\{x_{L_1},\dots,x_{U_1}\}$ and Event-I happens with probability 1. Suppose the claim is true for the first $k-1$ iterations. We consider the k-th iteration. For the $(k-1)$-th iteration, if Event-II happens with probability at least $1-(k-2)\cdot\delta/T_{max}$, then Event-II happens for the k-th iteration with the same probability. This is because the interval $\{x_{L_k},\dots,x_{U_k}\}$ is a subset of $\{x_{L_{k-1}},\dots,x_{U_{k-1}}\}$ and all points in the new interval satisfy the condition of Event-II.

Hence, we only need to consider the case when only Event-I for the $(k-1)$-th iteration happens with probability at least $1-(k-2)\delta/T_{max}$. We assume Event-I happens and consider conditional probabilities in the following of the proof. We denote

$$q_{1/3}:=\lfloor 2x_{L_{k-1}}/3+x_{U_{k-1}}/3\rfloor,\quad q_{2/3}:=\lfloor x_{L_{k-1}}/3+2x_{U_{k-1}}/3\rfloor$$

and discuss by two different cases.

Case I. Suppose one of the first two conditions holds when the current iteration terminates. Since the two conditions are symmetrical, we assume without loss of generality that the first condition holds. Then, the new interval is $[q_{1/3},x_{U_{k-1}}]$ and, by the definition of confidence interval, we know

$$f(q_{1/3})\geq f(q_{2/3})$$

holds with probability at least $1-\delta/T_{max}$. We assume the above event and Event-I for the $(k-1)$-th iteration both happen, which has joint probability at least $1-\delta/T_{max}-(k-2)\delta/T_{max}=1-(k-1)\delta/T_{max}$. By the convexity of $f(x)$, it holds that

$$f(x)\geq f(q_{1/3})+\frac{x-q_{1/3}}{q_{1/3}-q_{2/3}}\left[f(q_{1/3})-f(q_{2/3})\right]\geq f(q_{1/3}),\quad \forall x\in\{x_{L_{k-1}},\dots,q_{1/3}\}.$$

Hence, the minimum of $f(x)$ in $\{x_{L_{k-1}},\dots,x_{U_{k-1}}\}$ is attained by a point in $\{q_{1/3},\dots,x_{U_{k-1}}\}$. Combining with the assumption that there exists a solution to problem (1) in $\{x_{L_{k-1}},\dots,x_{U_{k-1}}\}$, we know that there exists a solution to problem (1) in $\{q_{1/3},\dots,x_{U_{k-1}}\}$. Thus, Event-I for the k-th iteration happens with probability at least $1-(k-1)\delta/T_{max}$.

Case II. Suppose only the last condition holds when the current iteration terminates. Since the first two conditions do not hold, we have

$$\left|\hat{F}_n(q_{1/3})-\hat{F}_n(q_{2/3})\right|\leq\epsilon/4. \tag{EC.8}$$

In addition, by the definition of confidence interval, it holds

$$\left|f(q_{1/3})-\hat{F}_n(q_{1/3})\right|\leq\epsilon/8,\quad \left|f(q_{2/3})-\hat{F}_n(q_{2/3})\right|\leq\epsilon/8$$

with probability at least $1-\delta/T_{max}$. Combining with inequality (EC.8), we know

$$\left|f(q_{1/3})-f(q_{2/3})\right| \le \epsilon/2 \tag{EC.9}$$

holds with probability at least $1-\delta/T_{max}$. We assume that the above event and Event-I for the $(k-1)$-th iteration both happen, which has joint probability at least $1-\delta/T_{max}-(k-2)\delta/T_{max} = 1-(k-1)\delta/T_{max}$. We prove that if Event-I for the k-th iteration does not happen, then Event-II for the k-th iteration happens. Under the condition that Event-I does not happen, we assume without loss of generality that solutions to problem (1) are in $\{x_{L_{k-1}},\dots,q_{1/3}-1\}$. Using the convexity of function $f(x)$, we know

$$f(x) \ge f(q_{1/3}) - \frac{q_{1/3}-x}{q_{2/3}-q_{1/3}}\left[f(q_{1/3})-f(q_{2/3})\right], \quad \forall x \in \{x_{L_{k-1}},\dots,q_{1/3}\}.$$

Choosing

$$x \in \left(\operatorname*{arg\,min}_{y\in\mathcal{X}} \ f(y)\right) \cap \{x_{L_{k-1}},\dots,x_{U_{k-1}}\} \neq \emptyset,$$

we get

$$\begin{aligned}\min_{y\in\mathcal{X}} \ f(y) &\ge f(q_{1/3}) - \frac{q_{1/3}-x}{q_{2/3}-q_{1/3}}\left[f(q_{1/3})-f(q_{2/3})\right] \\ &\ge f(q_{1/3}) - \frac{q_{1/3}-x_{L_{k-1}}}{q_{2/3}-q_{1/3}}\cdot\epsilon/2 \ge f(q_{1/3})-\epsilon/2,\end{aligned}$$

where the last inequality is from the definition of 3-quantiles. Combining with inequality (EC.9), we get

$$\min_{y\in\mathcal{X}} \ f(y) \ge f(q_{2/3})-\epsilon.$$

By the convexity of $f(x)$, it holds that

$$\max_{x\in\{q_{1/3},\dots,q_{2/3}\}} \ f(x) = \max\{f(q_{1/3}),f(q_{2/3})\} \le \min_{y\in\mathcal{X}} \ f(y)+\epsilon,$$

which means Event-II for the k-th iteration happens.

Combining the two cases, we know the claim holds for the k-th iteration. Suppose there are T iterations in Algorithm 1. By Lemma EC.4, we have $T \le T_{max}-1$. By the induction method, the last interval $\{x_{L_{T+1}},\dots,x_{U_{T+1}}\}$ satisfies the condition in Event-I or Event-II with probability at least $1-T\cdot\delta/T_{max} \ge 1-\delta+\delta/T_{max}$. If Event-II happens with probability at least $1-\delta+\delta/T_{max}$, then regardless of the point chosen in the sub-problem, the solution returned by the algorithm has value at most ϵ larger than the optimal value with probability at least $1-\delta+\delta/T_{max} \ge 1-\delta$. Hence, the solution satisfies the (ϵ,δ)-PGS guarantee. Otherwise, we assume Event-I happens with probability at least $1-\delta+\delta/T_{max}$. Then, a solution to problem (1) is in $\{x_{L_{T+1}},\dots,x_{U_{T+1}}\}$. We choose

$$x^* \in \left(\operatorname*{arg\,min}_{x\in\mathcal{X}} \ f(x)\right) \cap \{x_{L_{T+1}},\dots,x_{U_{T+1}}\}$$

and suppose the algorithm returns

$$x^{**} \in \underset{x \in \{x_{L_{T+1}}, \ldots, x_{U_{T+1}}\}}{\arg\min} \hat{F}_n(x).$$

By the definition of confidence interval, it holds

$$f(x^{**}) \le \hat{F}_n(x^{**}) + \epsilon/2, \quad f(x^*) \ge \hat{F}_n(x^*) - \epsilon/2$$

with probability at least $1 - \delta/T_{max}$. Under the above event, we get

$$f(x^{**}) \le \hat{F}_n(x^{**}) + \epsilon/2 \le \hat{F}_n(x^*) + \epsilon/2 \le f(x^*) + \epsilon.$$

Recalling that Event-I happens with probability at least $1 - \delta + \delta/T_{max}$, the point x^{**} satisfies the above relation with probability at least $1 - \delta$ and therefore satisfies the (ϵ, δ)-PGS guarantee. Combining with the first case, we know Algorithm 1 is an $[(\epsilon, \delta)\text{-PGS}, \mathcal{MC}]$-algorithm. □

EC.2.2. Proof of Theorem 2

We first estimate the simulation cost of Algorithm 2.

LEMMA EC.7. *Suppose that Assumptions 1-3 hold. The expected simulation cost for Algorithm 2 is at most*

$$\frac{25600\sigma^2}{\epsilon^2} \log\left[\frac{4N}{\delta}\right] = O\left[\frac{1}{\epsilon^2} \log\left(\frac{1}{\delta}\right)\right].$$

Proof. Denote $T_{max} := N$. Suppose that there are T iterations in Algorithm 2. We denote $\mathcal{S}_k$ as the active set at the beginning of the k-th iteration. Since each iteration reduces the size of $\mathcal{S}_k$ by at least 1, it follows that

$$|\mathcal{S}_k| \ge |\mathcal{S}_{T+1}| + T + 1 - k, \quad \forall k \in [T+1].$$

By the same analysis as Lemma EC.5, we know that for the k-th iteration, simulating

$$n(|\mathcal{S}_k|) := \frac{12800\sigma^2}{|\mathcal{S}_k|^2\epsilon^2} \log\left(\frac{4T_{max}}{\delta}\right)$$

times is sufficient to achieve $1 - \delta/(2T_{max})$ confidence half-width $|\mathcal{S}_k|/80 \cdot \epsilon$. By the condition on Line 6, each point discarded during the k-th iteration is simulated at most $n(|\mathcal{S}_k|)$ times. Hence, the total number of simulations on points discarded during the k-th iteration is at most

$$\begin{aligned}(|\mathcal{S}_k| - |\mathcal{S}_{k+1}|) \cdot n(|\mathcal{S}_k|) &= \frac{|\mathcal{S}_k| - |\mathcal{S}_{k+1}|}{|\mathcal{S}_k|^2} \cdot \frac{12800\sigma^2}{\epsilon^2} \log\left(\frac{4T_{max}}{\delta}\right) \\ &\le \left(\frac{1}{|\mathcal{S}_{k+1}|} - \frac{1}{|\mathcal{S}_k|}\right) \cdot \frac{12800\sigma^2}{\epsilon^2} \log\left(\frac{4T_{max}}{\delta}\right),\end{aligned}$$

where the inequality is because of $|\mathcal{S}_k| \geq |\mathcal{S}_{k+1}|$. Summing over $k = 1, 2, \ldots, T$, we get the number of simulations on all discarded points during iterations is at most

$$\sum_{k=1}^{T} \left(\frac{1}{|\mathcal{S}_{k+1}|} - \frac{1}{|\mathcal{S}_k|}\right) \cdot \frac{12800\sigma^2}{\epsilon^2} \log\left(\frac{4T_{max}}{\delta}\right) = \left(\frac{1}{|\mathcal{S}_{T+1}|} - \frac{1}{|\mathcal{S}_1|}\right) \cdot \frac{12800\sigma^2}{\epsilon^2} \log\left(\frac{4T_{max}}{\delta}\right)$$
$$\leq \left(1 - \frac{1}{N}\right) \cdot \frac{12800\sigma^2}{\epsilon^2} \log\left(\frac{4T_{max}}{\delta}\right) \leq \frac{12800\sigma^2}{\epsilon^2} \log\left(\frac{4T_{max}}{\delta}\right).$$

For points in the last active set $\mathcal{S}_{T+1}$, the number of simulations is bounded by

$$|\mathcal{S}_{T+1}| \cdot n(|\mathcal{S}_{T+1}|) = \frac{12800\sigma^2}{|\mathcal{S}_{T+1}|\epsilon^2} \log\left(\frac{4T_{max}}{\delta}\right) \leq \frac{12800\sigma^2}{\epsilon^2} \log\left(\frac{4T_{max}}{\delta}\right).$$

Combining two parts, we know that the simulation cost of Algorithm 2 is at most

$$\frac{25600\sigma^2}{\epsilon^2} \log\left(\frac{4T_{max}}{\delta}\right).$$

□

Then, we prove the correctness of Algorithm 2. The following lemma plays a critical role in verifying the correctness of Type-II Operations.

LEMMA EC.8. *Suppose function $h(x)$ is convex on $[1, M+a]$, where integer $M \geq 3$ and constant $a \in [0, 1]$. Then, the restriction of function $h(x)$ to $[M]$, which we denote as $\tilde{h}(x)$, is also convex. Furthermore, given a constant $\epsilon > 0$, if it holds that*

$$\max_{x \in [M]} \tilde{h}(x) - \min_{x \in [M]} \tilde{h}(x) \leq M/20 \cdot \epsilon, \tag{EC.10}$$

then we know

$$\min_{y \in [M']} \tilde{h}(2y-1) - \min_{x \in [1, M+a]} h(x) \leq \epsilon/2.$$

where we define $M' := \lceil M/2 \rceil$.

Proof. Since the midpoint convexity of $h(x)$ implies the discrete midpoint convexity of $\tilde{h}$, we know $\tilde{h}(x)$ is also convex. We prove the second claim in three steps.

Step 1. We first prove that

$$\min_{x \in [1, M]} h(x) - \min_{x \in [1, M+a]} h(x) \leq \epsilon/10. \tag{EC.11}$$

Suppose x^* is a minimizer of $h(x)$ on $[1, M+a]$. If $x^* \in [1, M]$, then inequality (EC.11) holds trivially. We assume that $x^* \in (M, M+a]$. By the convexity of $h(x)$, we have

$$h(M) - h(x^*) \leq \frac{x^* - M}{M-1} \cdot [h(M) - h(1)] \leq \frac{1}{M-1} \cdot [h(M) - h(1)] \leq \frac{M/20 \cdot \epsilon}{M/2} = \epsilon/10.$$

Hence, we know

$$\min_{x \in [1, M]} h(x) - \min_{x \in [1, M+a]} h(x) = \min_{x \in [1, M]} h(x) - h(x^*) \leq h(M) - h(x^*) \leq \epsilon/10.$$

Step 2. Next, we prove that

$$\min_{x\in[M]} \tilde{h}(x) - \min_{x\in[1,M]} h(x) \le \epsilon/5. \tag{EC.12}$$

Let x^* be a minimizer of $\tilde{h}(x)$ on $[M]$. By inequality (EC.10), we know

$$\max_{x\in[M]} \tilde{h}(x) = \max\{\tilde{h}(1), \tilde{h}(M)\} \le \tilde{h}(x^*) + M/20 \cdot \epsilon.$$

By the convexity of $h(x)$, there exists a minimizer $x^{**} \in [1, M]$ of $h(x)$ in $(x^* - 1, x^* + 1)$. If $x^{**} = x^*$, then $\min \tilde{h}(x) = \min h(x)$ and inequality (EC.12) holds. Hence, we assume that $x^{**} \neq x^*$ and, without loss of generality, $x^{**} \in (x^*, x^* + 1)$. Since $(M - x^* - 1) + (x^* - 1) = M - 2$, we know $\max\{M - x^* - 1, x^* - 1\} \ge \lceil (M-2)/2 \rceil$. We first consider the case when

$$M - x^* - 1 \ge \lceil (M-2)/2 \rceil.$$

By the convexity of $h(x)$, we have

$$\begin{aligned} h(x^*+1) - h(x^{**}) &\le \frac{x^* + 1 - x^{**}}{M - x^* - 1} \cdot [h(M) - h(x^*+1)] \\ &\le \frac{1}{\lceil (M-2)/2 \rceil} \cdot [h(M) - h(x^*)] \le \frac{M/20 \cdot \epsilon}{\lceil (M-2)/2 \rceil}. \end{aligned}$$

By simple calculations, we get $M/4 \le \lceil (M-2)/2 \rceil$ for all $M \ge 3$ and therefore

$$h(x^*) - h(x^{**}) \le h(x^*+1) - h(x^{**}) \le \epsilon/5,$$

which means inequality (EC.12) holds. Now we consider the case when

$$x^* - 1 \ge \lceil (M-2)/2 \rceil.$$

Similarly, by the convexity of $h(x)$, we have

$$h(x^*) - h(x^{**}) \le \frac{x^{**} - x^*}{x^* - 1} \cdot [h(x^*) - h(1)] \le \frac{1}{\lceil (M-2)/2 \rceil} \cdot [h(x^*) - h(1)] \le \frac{M/20 \cdot \epsilon}{\lceil (M-2)/2 \rceil} \le \epsilon/5.$$

Combining the two cases, we know inequality (EC.12) holds.

Step 3. Finally, we prove that

$$\min_{y\in[M']} \tilde{h}(2y-1) - \min_{x\in[M]} \tilde{h}(x) \le \epsilon/5. \tag{EC.13}$$

Let x^* be a minimizer of $\tilde{h}(x)$. If x^* is an odd number, then $\min_{y\in[M']} \tilde{h}(2y-1) = \min_{x\in[M]} \tilde{h}(x)$ and inequality (EC.13) holds. Otherwise, we assume $x^* = 2y^*$ is an even number. Then, by the convexity of $\tilde{h}(x)$, there exists a minimizer of $\tilde{h}(2y-1)$ in $\{y^*, y^*+1\}$. Without loss of generality, we assume

y^*+1 is a minimizer of $\tilde{h}(2y-1)$. Since $(M-x^*)+(x^*-1)=M-1$, we have $\max\{M-x^*, x^*-1\} \geq \lceil (M-1)/2 \rceil$. We first consider the case when

$$M - x^* \geq \lceil (M-1)/2 \rceil.$$

By the convexity of $\tilde{h}(x)$, we have

$$\tilde{h}(2y^*+1) - \tilde{h}(2y^*) \leq \frac{1}{M-2y^*} \cdot \left[\tilde{h}(M) - \tilde{h}(2y^*)\right] \leq \frac{M/20 \cdot \epsilon}{\lceil (M-1)/2 \rceil}.$$

We can verify that $M/4 \leq \lceil (M-1)/2 \rceil$ for all $M \geq 3$. Hence, it holds that

$$\tilde{h}(2y^*+1) - \tilde{h}(2y^*) \leq \epsilon/5.$$

Then, we consider the case when

$$x^* - 1 \geq \lceil (M-1)/2 \rceil.$$

Similarly, using the convexity of $\tilde{h}(x)$, we have

$$\tilde{h}(2y^*-1) - \tilde{h}(2y^*) \leq \frac{1}{2y^*-1} \cdot \left[\tilde{h}(2y^*) - \tilde{h}(1)\right] \leq \frac{M/20 \cdot \epsilon}{\lceil (M-1)/2 \rceil} \leq \epsilon/5,$$

which implies that

$$\tilde{h}(2y^*+1) - \tilde{h}(2y^*) \leq \tilde{h}(2y^*-1) - \tilde{h}(2y^*) \leq \epsilon/5.$$

Combining the two cases, we know inequality (EC.13) holds.

By inequalities (EC.11), (EC.12) and (EC.13), we have

$$\min_{y \in [M']} \tilde{h}(2y-1) - \min_{x \in [1, M+a]} h(x) \leq \epsilon/10 + \epsilon/5 + \epsilon/5 = \epsilon/2.$$

□

We denote $\mathcal{S}_k$ and d_k as the active set and the step size at the beginning of the k-th iteration, respectively. We define the upper bound and the lower bound for the k-th iteration as

$$x_{x_{L_1}} := 1, \quad x_{L_{k+1}} := \begin{cases} y + d_k & \text{if the second case of Type-I Operation happens} \\ x_{L_k} & \text{otherwise,} \end{cases}$$

$$x_{x_{U_1}} := N, \quad x_{U_{k+1}} := \begin{cases} y - d_k & \text{if the first case of Type-I Operation happens} \\ x_{U_k} & \text{otherwise.} \end{cases}$$

Although not explicitly defined in the algorithm, the interval $\{x_{L_k}, \ldots, x_{U_k}\}$ plays a similar role as in the TS algorithm and characterizes the set of possible solutions. In the following lemma, we prove that the active set $\mathcal{S}_k$ is a good approximation to the interval $\{x_{L_k}, \ldots, x_{U_k}\}$. We note that the following lemma is deterministic.

LEMMA EC.9. *For any iteration k, we have*

$$x_{L_k} = \min\ \mathcal{S}_k \quad \textit{and} \quad x_{U_k} \leq \max\ \mathcal{S}_k + d_k. \tag{EC.14}$$

Proof. We use the induction method to prove the result. When $k=1$, we know $x_{L_1}=1, x_{U_1}=N, \mathcal{S}=[N]$ and $d_1=1$. Hence, the relations in (EC.14) hold. We assume these relations hold for the first $k-1$ iterations. We discuss by two different cases.

Case I. Type-I Operation is implemented during the $(k-1)$-th iteration. If the first case of Type-I Operation happens, then we know $x_{L_k} = x_{L_{k-1}}$ and $x_{U_k} = y - d_{k-1}$. By the updating rule, the step size d_{k-1} is not changed and all points in $\mathcal{S}_{k-1}$ that are at least y are discarded from $\mathcal{S}_{k-1}$. Hence, it follows that $\max\ \mathcal{S}_k = x_{U_k}$ and the inequality $x_{U_k} \leq \max\ \mathcal{S}_k + d_k$ holds. Moreover, since both x_{L_k} and $\min\ \mathcal{S}_{k-1}$ are not changed, the equality $x_{L_k} = \min\ \mathcal{S}_k$ still holds.

Otherwise if the second case of Type-I Operation happens, then we know $x_{L_k} = y + d_{k-1}$ and $x_{U_k} = x_{U_{k-1}}$. Similarly, we can prove that $x_{L_k} = \min\ \mathcal{S}_k$. Moreover, since $d_k = d_{k-1}$ and $\max\ \mathcal{S}_{k-1} + d_{k-1} = \max\ \mathcal{S}_k + d_{k-1}$, it holds

$$x_{U_k} = x_{U_{k-1}} \leq \max\ \mathcal{S}_{k-1} + d_{k-1} = \max\ \mathcal{S}_k + d_k.$$

Case II. Type-II Operation is implemented during the $(k-1)$-th iteration. In this case, bounds $x_{L_{k-1}}$ and $x_{U_{k-1}}$ are not changed. By the update rule, we know the step size $d_k = 2d_{k-1}$ and

$$\min\ \mathcal{S}_k = \min\ \mathcal{S}_{k-1}, \quad \max\ \mathcal{S}_k \in \{\max\ \mathcal{S}_{k-1} - d_{k-1}, \max\ \mathcal{S}_{k-1}\}. \tag{EC.15}$$

Thus, the equality $x_{L_k} = \min\ \mathcal{S}_k$ still holds. By the induction assumption, we know that

$$x_{U_k} \leq x_{U_{k-1}} \leq \max\ \mathcal{S}_{k-1} + d_{k-1}.$$

Combining with the latter relation in (EC.15), we get

$$x_{U_k} \leq \max\ \mathcal{S}_{k-1} + d_{k-1} \leq \max\ \mathcal{S}_k + 2d_{k-1} = \max\ \mathcal{S}_k + d_k.$$

Combining the two cases, we know the relations in (EC.14) hold for the k-th iteration. By the induction method, the relations hold for all iterations. □

Finally, utilizing Lemmas EC.8 and EC.9, we can prove the correctness of Algorithm 2 and get a better upper bound on $T_0(\epsilon, \delta, \mathcal{MC})$.

Proof of Theorem 2. Denote $T_{max} := N$. We use the induction method to prove that, for any iteration k, the two events

- $\min_{x \in \mathcal{S}_k}\ f(x) \leq \min_{x \in \mathcal{X}}\ f(x) + \epsilon/2$.
- $\min_{x \in \{x_{L_k}, x_{L_k}+1, \ldots, x_{U_k}\}}\ f(x) = \min_{x \in \mathcal{X}}\ f(x)$.

happen jointly with probability at least $1 - (k-1)\delta/T_{max}$. When $k = 1$, we know $\mathcal{S}_1 = \mathcal{X}$ and $x_{L_1} = 1, x_{U_1} = N$. Hence, the two events happen with probability 1. Suppose the claim is true for the first $k-1$ iterations. We assume the two events happen for the $(k-1)$-th iteration and consider conditional probabilities in the following proof. We discuss by two different cases.

Case I. Type-I Operation is implemented in the $(k-1)$-th iteration. In this case, there exists $x, y \in \mathcal{S}_{k-1}$ such that $\hat{F}_{n_x}(x) + h_x \leq \hat{F}_{n_y}(y) - h_y$. By the definition of confidence intervals, we know $f(x) \leq f(y)$ holds with probability at least $1 - \delta/T_{max}$. We assume event $f(x) \leq f(y)$ happens jointly with the claim for the $(k-1)$-th iteration, which has probability at least $1-(k-2)\delta/T_{max} - \delta/T_{max} = 1-(k-1)\delta/T_{max}$. If $x < y$, then using the convexity of $f(x)$, we know

$$f(z) \geq f(y) \geq f(x), \quad \forall z \in [N] \quad \text{s.t. } z \geq y,$$

which means all discarded points have function values at least $f(x)$. Hence, the minimums in the claim are not changed, i.e., we have

$$\min_{x \in \mathcal{S}_k} \ f(x) = \min_{x \in \mathcal{S}_{k-1}} \ f(x) \leq \min_{x \in \mathcal{X}} \ f(x) + \epsilon/2$$

and

$$\min_{x \in \{x_{L_k}, x_{L_k}+1, \ldots, x_{U_k}\}} \ f(x) = \min_{x \in \{x_{L_{k-1}}, x_{L_{k-1}}+1, \ldots, x_{U_{k-1}}\}} \ f(x) = \min_{x \in \mathcal{X}} \ f(x).$$

The two events happen with probability at least $1-(k-1)\delta/T_{max}$. If $y < x$, the proof is the same and therefore the claim holds for the k-th iteration.

Case II. Type-II Operation is implemented in the $(k-1)$-th iteration. Since $x_{L_{k-1}}$ and $x_{U_{k-1}}$ are not changed, the first event happens for the k-th iteration. Hence, we only need to verify that the second event happens with high probability. Let x^* and x^{**} be a minimizer and a maximizer of $f(x)$ on $\mathcal{S}_{k-1}$, respectively. By the condition of Type-II Operations, we know

$$|\hat{F}_n \ (x^*) - \hat{F}_n(x^{**})| \leq h_{x^*} + h_{x^{**}}$$

and

$$h_x \leq |\mathcal{S}_{k-1}|/80 \cdot \epsilon, \quad \forall x \in \mathcal{S}_{k-1}.$$

By the definition of confidence intervals, it holds

$$|f(x^*) - \hat{F}_n \ (x^*)| \leq h_{x^*}, \quad |f(x^{**}) - \hat{F}_n \ (x^{**})| \leq h_{x^{**}}$$

with probability at least $1 - \delta/T_{max}$. Under the above event, we have

$$\begin{aligned} |f(x^*) - f(x^{**})| &\leq |f(x^*) - \hat{F}_n \ (x^*)| + |\hat{F}_n \ (x^*) - \hat{F}_n(x^{**})| + |f(x^{**}) - \hat{F}_n \ (x^{**})| \\ &\leq 2(h_{x^*}) + h_{x^{**}} \leq M/20 \cdot \epsilon. \end{aligned} \tag{EC.16}$$

We assume the above event happens jointly with with the claim for the $(k-1)$ -th iteration, which has probability at least $1-(k-1)\delta/T_{max}$. By the induction assumption, the original problem (1) is equivalent to

$$\min_{x \in \{x_{L_{k-1}}, x_{L_{k-1}}+1, \ldots, x_{U_{k-1}}\}} \ f(x).$$

Moreover, if we denote $\tilde{f}$ as the linear interpolation of $f(x)$ defined in (2), then the above problem is equivalent to

$$\min_{x\in\{x_{L_{k-1}},x_{L_{k-1}+1},\ldots,x_{U_{k-1}}\}} f(x) = \min_{x\in[x_{L_{k-1}},x_{U_{k-1}}]} \tilde{f}(x). \tag{EC.17}$$

We define the constant $\tilde{M} := (x_{U_{k-1}} - x_{L_{k-1}})/d_{k-1} + 1$ and the linear transformation

$$T(x) := x_{L_{k-1}} + d_{k-1}(x-1), \quad \forall x \in [1, \tilde{M}].$$

The inverse image $T^{-1}\left([x_{L_{k-1}}, x_{U_{k-1}}]\right)$ is $[1, \tilde{M}]$. Defining the composite function

$$\tilde{g}(x) := \tilde{f}(T(x)), \quad \forall x \in [1, \tilde{M}],$$

we know that the problem (EC.17) is equivalent to

$$\min_{x\in[1,\tilde{M}]} \tilde{g}(x). \tag{EC.18}$$

The inverse image $T^{-1}(\mathcal{S}_{k-1})$ is $[M]$, where $M := |\mathcal{S}_{k-1}|$ is the number of points in $\mathcal{S}_{k-1}$. Lemma EC.9 implies that $a := \tilde{M} - M \in [0,1]$. Recalling inequality (EC.16), we get

$$\max_{x\in[M]} \tilde{g}(x) - \min_{x\in[M]} \tilde{g}(x) \le M/20 \cdot \epsilon.$$

Now, we can apply Lemma EC.8 to get

$$\min_{y\in[M']} \tilde{g}(2y-1) - \min_{x\in[1,M+a]} \tilde{g}(x) = \min_{y\in[M']} \tilde{g}(2y-1) - \min_{x\in[1,\tilde{M}]} \tilde{g}(x) \le \epsilon/2,$$

where $M' := \lceil M/2 \rceil$. We note that the interval $[1, M+a]$ corresponds to the interval $[1, N]$ before scaling the x-axis. Since problem (EC.18) is equivalent to problem (EC.17) and further equivalent to problem (1), it holds that

$$\min_{y\in[M']} \tilde{g}(2y-1) - \min_{x\in[1,\tilde{M}]} \tilde{g}(x) = \min_{y\in[M']} \tilde{g}(2y-1) - \min_{x\in\mathcal{X}} f(x) \le \epsilon/2.$$

By the definition of $\mathcal{S}_k$, we know $T(2[M']-1)$ is $\mathcal{S}_k$ and therefore

$$\min_{y\in[M']} \tilde{g}(2y-1) - \min_{x\in\mathcal{X}} f(x) = \min_{x\in\mathcal{S}_k} f(x) - \min_{x\in\mathcal{X}} f(x) \le \epsilon/2,$$

which implies that the second case happens for the k-th iteration with probability at least $1-(k-1)\delta/T_{max}$.

Combining the above two cases, we know the claim holds for all iterations. Suppose there are T iterations in Algorithm 2. Since each iteration will decrease the active set $\mathcal{S}$ by at least 1, we get $T \le N-1$. Then after the T iterations, we have

$$\min_{x\in\mathcal{S}_{T+1}} f(x) \le \min_{x\in\mathcal{X}} f(x) + \epsilon/2 \tag{EC.19}$$

holds with probability at least $1 - T\cdot\delta/T_{max} \ge 1 - \delta + \delta/T_{max}$. For the sub-problem, using the same analysis as Theorem 1, the point returned by Algorithm 2 satisfies the $(\epsilon/2, \delta/T_{max})$-PGS guarantee. Combining with the relation (EC.19), we know the algorithm returns a solution satisfying the (ϵ,δ)-PGS guarantee. □

EC.3. Multi-dimensional Shrinking Uniform Sampling Algorithm

In this section, we give the multi-dimensional version of the SUS algorithm designed in Section 3.2. Similar to the one-dimensional case, the asymptotic simulation cost of the multi-dimensional algorithm is upper bounded by a constant that does not depend on the problem scale N and the Lipschitz constant of the objective function. Hence, the multi-dimensional algorithm provides a matching simulation cost to the one-dimensional case. However, the expected simulation cost is exponentially dependent on the dimension d. Therefore, the multi-dimensional SUS algorithm is mainly theoretical and only suitable for low-dimensional problems.

The main idea of the generalization to multi-dimensional problems is to view optimization algorithms as (usually biased) estimators to the optimal value, which is elaborated in the following definition.

DEFINITION EC.1. Given a constant $S > 0$, we say that an algorithm is **sub-Gaussian with dimension d and parameter** S if for any d-dimensional $L^\natural$-convex problem, any $\epsilon > 0$ and small enough $\delta > 0$, the algorithm returns an ϵ-optimal solution $\hat{x}$ along with an estimate $\hat{f}^*$ to the optimal value f^* that satisfies $|\hat{f}^* - f^*| \leq \epsilon$ with probability at least $1-\delta$ using at most

$$T(\epsilon, \delta) := \tilde{O}\left[\frac{2S}{\epsilon^2}\log(\frac{2}{\delta})\right]$$

simulation runs.

For example, Theorem 2 shows that the one-dimensional SUS algorithm (Algorithm 2) returns an $(\epsilon/2, \delta/2)$-PGS solution with $\tilde{O}[\epsilon^{-2}\log(1/\delta)]$ simulations. Then, we can simulate the function value at the solution for $O[\epsilon^{-2}\log(1/\delta)]$ times such that the $1-\delta/2$ confidence half-width becomes smaller than $\epsilon/4$. Then, the empirical mean of function values at the solution is at most ϵ distant from f^* with probability at least $1-\delta$. Hence, we know that Algorithm 2 is sub-Gaussian with dimension 1. We denote its associated parameter as S. We note that if we treat algorithms as estimators, the estimators are generally "*biased*" (but consistent). This fact implies that the empirical mean of several estimates to the optimal value does not produce a better optimality guarantee, while the empirical mean of several unbiased estimators usually has a tighter deviation bound.

Now, we inductively construct sub-Gaussian algorithms for multi-dimensional problems. We first define the marginal objective function as

$$f^{d-1}(x) := \min_{y \in [N]^{d-1}} f(y, x). \tag{EC.20}$$

Observe that each evaluation of $f^{d-1}(x)$ requires solving a $(d-1)$-dimensional $L^\natural$-convex subproblem. Hence, if we have an algorithm for $(d-1)$-dimensional $L^\natural$-convex problems, we only need to solve the one-dimensional problem

$$\min_{x \in [N]} f^{d-1}(x) = \min_{x \in [N]} \min_{y \in [N]^{d-1}} f(y, x) = \min_{x \in \mathcal{X}} f(x) \tag{EC.21}$$

Moreover, we can prove that problem (EC.21) is also a convex problem.

LEMMA EC.10. *If function $f(x)$ is $L^\natural$-convex, then function $f^{d-1}(x)$ is $L^\natural$-convex on $[N]$.*

Proof of Lemma EC.10. The proof is provided in EC.3.1. □

Based on the observations above, we can use sub-Gaussian algorithms for $(d-1)$-dimensional problems and Algorithm 2 to construct sub-Gaussian algorithms for d-dimensional problems. We give the pseudo-code in Algorithm 8.

Algorithm 8 Multi-dimensional shrinking uniform sampling algorithm

Input: Model $\mathcal{X}, (\Upsilon, \mathcal{B}_\Upsilon), F(x, \xi_x)$, optimality guarantee parameters ϵ and δ, sub-Gaussian algorithm $\mathcal{A}$ with dimension $d-1$.

Output: An (ϵ, δ)-PGS solution x^* to problem (1).

Set the active set $\mathcal{S} \leftarrow [N]$.
Set the step size $s \leftarrow 1$ and the maximal number of comparisons $T_{max} \leftarrow N$.
Set $N_{cur} \leftarrow +\infty$.
while the size of $\mathcal{S}$ is at least 3 **do** ▷ Iterate until $\mathcal{S}$ has at most 2 points.
  if $|\mathcal{S}| \le N_{cur}/2$ **then** ▷ Update the confidence interval.
    Record current active set size $N_{cur} \leftarrow |\mathcal{S}|$.
    Set the confidence half-width $h \leftarrow N_{cur} \cdot \epsilon/160$.
    For each $x \in \mathcal{S}$, use algorithm $\mathcal{A}$ to get an estimate to $f^{d-1}(x)$ such that

$$\left| \hat{f}^{d-1}(x) - f^{d-1}(x) \right| \le h$$

    holds with probability at least $1 - \delta/(2T_{max})$.
  end if
  if $\hat{f}^{d-1}(x) + h \le \hat{f}^{d-1}(y) - h$ for some $x, y \in \mathcal{S}$ **then** ▷ **Type-I Operation**
    if $x < y$ **then**
      Remove all points $z \in \mathcal{S}$ with the property $z \ge y$ from $\mathcal{S}$.
    else
      Remove all points $z \in \mathcal{S}$ with the property $z \le y$ from $\mathcal{S}$.
    end if
  else ▷ **Type-II Operation**
    Update the step size $s \leftarrow 2s$.
    Update $\mathcal{S} \leftarrow \{x_{min}, x_{min} + s, \ldots, x_{min} + ks\}$, where $x_{min} = \min_{x \in \mathcal{S}} x$ and $k = \lceil |\mathcal{S}|/2 \rceil - 1$.
  end if
end while ▷ Now $\mathcal{S}$ has at most 2 points.

21: For each $x \in \mathcal{S}$, use Algorithm $\mathcal{A}$ to obtain an estimate to $f^{d-1}(x)$ such that

$$\left|\hat{f}^{d-1}(x) - f^{d-1}(x)\right| \leq \epsilon/4$$

holds with probability at least $1 - \delta/(2T_{max})$.

22: Return $x^* \leftarrow \arg\min_{x\in\mathcal{S}} \ \hat{f}^{d-1}(x)$.

We note that the bound $N_{cur} \cdot \epsilon/160$ in Line 7 is tighter than the condition in Algorithm 2. This is also because the "algorithm estimators" are usually biased. Therefore, taking the mean of several algorithm estimators does not necessarily reduce the width of confidence intervals, and the algorithm needs to generate a new estimator with a better confidence interval to guarantee the desired PGS criterion (instead of taking the mean of two less accurate estimators to improve the precision). We prove that Algorithm 8 is sub-Gaussian with dimension d and estimate its parameter.

THEOREM EC.5. *Suppose that Assumptions 1-3 hold, and that Algorithm $\mathcal{A}$ is sub-Gaussian with dimension $d-1$ and parameter S. Then, Algorithm 8 is a sub-Gaussian algorithm with dimension d and parameter MC, where $M > 0$ is an absolute constant.*

Proof of Theorem EC.5. The proof is provided in EC.3.2. □

If we treat $F(x, \xi_x)$ as a sub-Gaussian algorithm with dimension 0 and parameter σ^2, then Theorem EC.5 implies that there exists a sub-Gaussian algorithm with dimension 1 and parameter $\sigma^2 M$. However, the parameter S of Algorithm 2 is usually smaller than $\sigma^2 M$ and therefore Algorithm 2 is preferred in the one-dimensional case. Using the results of Theorem EC.5 and the fact that Algorithm 2 is sub-Gaussian with dimension 1, we can inductively construct sub-Gaussian algorithms with any dimension d.

THEOREM EC.6. *Suppose that Assumptions 1-3 hold and Algorithm 2 is sub-Gaussian with dimension 1 and parameter S. There exists an $[(\epsilon,\delta)$-PGS$, \mathcal{MC}]$-algorithm that is sub-Gaussian with parameter $M^{d-1}S$, where M is the constant in Theorem EC.5. Hence, we have*

$$T(\epsilon, \delta, \mathcal{MC}) = \tilde{O}\left[\frac{M^d}{\epsilon^2}\log\left(\frac{1}{\delta}\right)\right].$$

Furthermore, by choosing $\epsilon = c/2$, it holds that

$$T(\delta, \mathcal{MC}_c) = \tilde{O}\left[\frac{M^d}{c^2}\log\left(\frac{1}{\delta}\right)\right].$$

We note that although the upper bound in Theorem EC.5 is independent of the Lipschitz constant L and independent of N when $\delta \ll 1$, the dependence on d is exponential. Hence, Algorithm 8 is

largely theoretical and only suitable for low-dimensional problems, e.g., problems with $d \leq 3$. On the other hand, if the dimension d is treated as a fixed constant, Algorithm 8 attains the optimal asymptotic performance under the asymptotic criterion in Kaufmann et al. (2016). We also mention that Algorithm 8 does not make a full use of the properties of $L^\natural$-convex functions. Actually, Algorithm 8 is an (ϵ, δ)-PGS algorithm for those functions that are convex in each direction.

EC.3.1. Proof of Lemma EC.10

Proof of Lemma EC.10. Let $k \in \{2, 3, \ldots, N-1\}$. By the definition of $f^{d-1}(x)$, there exists vectors $y_{k-1}, y_{k+1} \in [N]^{d-1}$ such that

$$f^{d-1}(k-1) = f(y_{k-1}, k-1), \quad f^{d-1}(k+1) = f(y_{k+1}, k+1).$$

By the $L^\natural$-convexity of $f(x)$, we have

$$\begin{aligned} f^{d-1}(k-1) + f^{d-1}(k+1) &= f(y_{k-1}, k-1) + f(y_{k+1}, k+1) \\ &\geq f\left(\left\lceil \frac{y_{k-1}+y_{k+1}}{2} \right\rceil, k\right) + f\left(\left\lfloor \frac{y_{k-1}+y_{k+1}}{2} \right\rfloor, k\right) \\ &\geq 2 \min_{y \in [N]^{d-1}} f(y, k) = 2f^{d-1}(k), \end{aligned}$$

which means the discrete midpoint convexity holds at point k. Since we can choose k arbitrarily, we know function $f^{d-1}(x)$ is convex on $[N]$. □

EC.3.2. Proof of Theorem EC.5

Proof of Theorem EC.5. We first verify the correctness of Algorithm 8. The algorithm is the same as Algorithm 2 except the condition for implementing Type-II Operations. Hence, if we can prove that, when Type-II Operations are implemented, it holds

$$h \leq |\mathcal{S}| \cdot \epsilon/80, \tag{EC.22}$$

then the proof of Theorem 2 can be directly applied to this case. If the confidence interval is updated at the beginning of current iteration, then we have

$$h = |\mathcal{S}| \cdot \epsilon/160 < |\mathcal{S}| \cdot \epsilon/80.$$

Otherwise, if the confidence interval is not updated in the current iteration. Then, we have $|\mathcal{S}| > N_{cur}/2$ and therefore

$$h = N_{cur} \cdot \epsilon/160 < 2|\mathcal{S}| \cdot \epsilon/160 = |\mathcal{S}| \cdot \epsilon/80.$$

Combining the two cases, we have inequality (EC.22) and the correctness of Algorithm 8.

Next, we estimate the simulation cost of Algorithm 8. Denote the active sets when we update the confidence interval as $\mathcal{S}_1, \ldots, \mathcal{S}_m$, where $m \geq 1$ is the number of times when the confidence interval is updated. Then, we know $|\mathcal{S}_1| = N$ and $|\mathcal{S}_m| \geq 3$. By the condition for updating the confidence interval, it holds

$$|\mathcal{S}_{k+1}| \leq |\mathcal{S}_k|/2, \quad \forall k \in [m-1],$$

which implies

$$|\mathcal{S}_k| \geq 2^{m-k}|\mathcal{S}_m| \geq 3 \cdot 2^{m-k}, \quad \forall k \in [m].$$

Since the algorithm $\mathcal{A}$ is sub-Gaussian with parameter S, for each $x \in \mathcal{S}_k$, the simulation cost for generating $\hat{f}^{d-1}(x)$ is at most

$$\frac{2S}{h^2}\log(\frac{2T_{max}}{\delta}) = \frac{2S}{160^{-2}|\mathcal{S}_k|^2\epsilon^2}\log(\frac{2T_{max}}{\delta}) = |\mathcal{S}_k|^{-2} \cdot \frac{51200S}{\epsilon^2}\log(\frac{2T_{max}}{\delta}).$$

Hence, the total simulation cost for the k-th update of confidence intervals is at most

$$|\mathcal{S}_k| \cdot |\mathcal{S}_k|^{-2} \cdot \frac{51200S}{\epsilon^2}\log(\frac{2T_{max}}{\delta}) = |\mathcal{S}_k|^{-1} \cdot \frac{51200S}{\epsilon^2}\log(\frac{2T_{max}}{\delta}) \leq 2^{k-m}/3 \cdot \frac{51200S}{\epsilon^2}\log(\frac{2T_{max}}{\delta}).$$

Summing over all iterations, we have the simulation cost of all iterations of Algorithm 8 is at most

$$\sum_{k=1}^{m} 2^{k-m}/3 \cdot \frac{51200S}{\epsilon^2}\log(\frac{2T_{max}}{\delta}) = \left(2 - 2^{1-m}\right) \cdot \frac{51200S}{3\epsilon^2}\log(\frac{2T_{max}}{\delta}) < \frac{102400S}{3\epsilon^2}\log(\frac{2T_{max}}{\delta}).$$

Now we consider the simulation cost of the last subproblem. Since the algorithm 8 is sub-Gaussian with parameter S, the simulation cost of the subproblem is at most

$$2 \cdot \frac{2S}{(\epsilon/4)^2}\log(\frac{2T_{max}}{\delta}) = \frac{64S}{\epsilon^2}\log(\frac{2T_{max}}{\delta}).$$

Hence, the total simulation cost of Algorithm 8 is at most

$$\frac{102400S}{3\epsilon^2}\log(\frac{2T_{max}}{\delta}) + \frac{64S}{\epsilon^2}\log(\frac{2T_{max}}{\delta}) < 17099 \cdot \frac{2S}{\epsilon^2}\log(\frac{2T_{max}}{\delta}).$$

When δ is small enough, we can choose $M = 17100$ and the asymptotic simulation cost of Algorithm 8 is at most

$$\frac{2MC}{\epsilon^2}\log(\frac{2T_{max}}{\delta}),$$

which implies that Algorithm 8 is sub-Gaussian with dimension d and parameter MC. □

EC.4. Deterministic cutting-plane methods

In this section, we give the pseudo-codes of Vaidya's cutting-plane method (Vaidya 1996) and the random walk-based cutting-plane method (Bertsimas and Vempala 2004) for the self-contained purpose.

EC.4.1. Vaidya's cutting-plane method

We first give the pseudo-code for Vaidya's cutting-plane method (Vaidya 1996). We note that examples of Newton-type methods include the original Newton method, quasi-Newton methods and the cubic-regularized Newton method.

Algorithm 9 Vaidya's cutting-plane method

Input: Model $\mathcal{X}, f(x)$, optimality guarantee parameter ϵ, Lipschitz constant L, $\mathcal{SO}$ oracle $\hat{g}$.

Output: An ϵ-solution x^* to problem (1).

Set the initial polytope $P \leftarrow [1, N]^d$.
Set the constant $\rho \leftarrow 10^{-7}$. ▷ Constant ρ corresponds to ϵ in Vaidya (1996).
Set the number of iterations $T_{max} \leftarrow \lceil 2d/\rho \cdot \log[dNL/(\rho\epsilon)] \rceil$.
Initialize the set of points used to query separation oracles $\mathcal{S} \leftarrow \emptyset$.
Initialize the volumetric center $z \leftarrow (N+1)/2 \cdot (1, 1, \ldots, 1)^T$.
for $T = 1, 2, \ldots, T_{max}$ **do**
    Decide adding or removing a cutting plane by checking $\sigma_i(z)$ for $i \in [T]$ (Vaidya 1996).
    if add a cutting plane **then**
        Evaluate the $\mathcal{SO}$ oracle $\hat{g}$ at z.
        if $\hat{g} = 0$ **then**
            Return z as the approximate solution.
        end if
        Add the current point z to $\mathcal{S}$.
    else if remove a cutting plane **then**
        Remove corresponding point z from $\mathcal{S}$.
    end if
    Update the approximate volumetric center z by a Newton-type method.
end for ▷ There are at most $O(d)$ points in $\mathcal{S}$ by Vaidya's method.
Return the solution $\hat{x}$ to problem $\min_{x \in \mathcal{S}} f(x)$.

EC.4.2. Random walk-based cutting-plane method

Next, we list the pseudo-code for the random walk-based cutting-plane method in Bertsimas and Vempala (2004).

Algorithm 10 Deterministic random walk-based cutting-plane method

Input: Model $\mathcal{X}, f(x)$, optimality guarantee parameter ϵ, Lipschitz constant L, $\mathcal{SO}$ oracle $\hat{g}$.

Output: An ϵ-solution x^* to problem (1).

Set the initial polytope $P \leftarrow [1, N]^d$.
Set the number of iterations $T_{max} \leftarrow O[d \log(dLN/\epsilon)]$.
Set the number of samples required to calculate the center $M \leftarrow O(d)$.
Initialize the set of points used to query separation oracles $\mathcal{S} \leftarrow \emptyset$.
Initialize the volumetric center $z \leftarrow (N+1)/2 \cdot (1, 1, \ldots, 1)^T$.
for $T = 1, 2, \ldots, T_{max}$ **do**
    Evaluate the $\mathcal{SO}$ oracle $\hat{g}$ at z.
    Add the current point z to $\mathcal{S}$.
    Add the cutting plane using $\hat{g}$ to P.
    if $P = \emptyset$ **then** ▷ This step requires solving a linear feasibility problem
        break
    end if
    Uniformly sample M points from the new polytope P via random walk.
    Update the approximate volumetric center z to the average of all sampled points.
end for
Return the solution $\hat{x}$ to problem $\min_{x \in \mathcal{S}} \ f(x)$.

EC.5. Dimension reduction algorithm with the LLL algorithm

In this section, we provide a more detailed description for the dimension reduction algorithm that utilizes the LLL algorithm. More specifically, the LLL algorithm approximately solves the Shortest Vector Problem (SVP) in lattice to find the hyperplane H in Algorithm 5. Given a lattice Λ and a positive semi-definite matrix Σ that is full-rank on the span of Λ, the SVP problem is given by

$$\arg\min_{v \in \Lambda} \ v^T \Sigma v.$$

In the statement of the algorithm, we define $[x]$ to be the nearest integer to $x \in \mathbb{R}$.

Algorithm 11 Dimension reduction algorithm for the PGS guarantee

Input: Model $\mathcal{X}, (\mathsf{Y}, \mathcal{B}_{\mathsf{Y}}), F(x, \xi_x)$, optimality guarantee parameters ϵ and δ, (ϵ, δ)-$\mathcal{SO}$ oracle $\hat{g}$.

Output: An (ϵ, δ)-PGS solution x^* to problem (1).

Set the initial polytope $P \leftarrow [1, N]^d$.
Set the initial subspace $W \leftarrow \mathbb{R}^d$ and the initial lattice $\Lambda \leftarrow \mathbb{Z}^d$.
Initialize the set of points used to query separation oracles $\mathcal{S} \leftarrow \emptyset$.
for $d' = d, d-1, \ldots, 2$ **do** ▷ The current dimension d' is gradually reduced.
    Compute the the volumetric center z and covariance matrix Σ by Algorithm 10.

repeat Get an approximate solution v of SVP with Λ and Σ. ▷ Use the LLL algorithm.
  Take one step of Algorithm 10 with $(\epsilon/4, \delta/4)$-$\mathcal{SO}$ oracle.
    ▷ Algorithm 10 decides a suitable cutting plane H.
  Add the point where the stochastic separation oracle is called to $\mathcal{S}$.
  Shrink the volume of P using the cutting plane H.
  Update the the volumetric center z and covariance matrix Σ by Algorithm 10.
until $v^T \Sigma v \leq (10n)^{-2}$
Find the vector $\tilde{v} \in \mathbb{Z}^d$ such that v is the orthogonal projection of $\tilde{v}$ on hyperplane $-z + W$.
    ▷ The hyperplane $-z + W$ passes through the origin.
Construct hyperplane $H \leftarrow \{y \in \mathbb{R}^d \mid \langle y, v \rangle = \langle z, v - \tilde{v} \rangle + [\langle z, \tilde{v} \rangle]\}$.
Project P, W and Λ onto the hyperplane H. ▷ Reduce the dimension by 1.
Update the the volumetric center z and covariance matrix Σ by Algorithm 10.
end for
Find an $(\epsilon/4, \delta/4)$-PGS solution of the last one-dim problem and add the solution to $\mathcal{S}$.
Find the $(\epsilon/4, \delta/4)$-PGS solution $\hat{x}$ of problem $\min_{x \in \mathcal{S}} f(x)$.
Round $\hat{x}$ to an integral solution by Algorithm 3.

EC.6. Proofs in Section 4

EC.6.1. Proof of Lemma 5

Proof of Lemma 5. By the assumption that $F(x, \xi_x) - f(x)$ is sub-Gaussian with parameter σ^2 for any x, we know that $\hat{g}_{\alpha_x(i)} - g_{\alpha_x(i)}$ is the difference of two independent sub-Gaussian random variables and therefore

$$\hat{g}_{\alpha_x(i)} - g_{\alpha_x(i)} \sim \text{subGaussian}\left(2\sigma^2\right), \quad \forall i \in [d],$$

where g is the subgradient of $f(x)$ defined in (5). Then, using the properties of sub-Gaussian random variables, it holds that

$$\hat{g}^n_{\alpha_x(i)} - g_{\alpha_x(i)} \sim \text{subGaussian}\left(\frac{2\sigma^2}{n}\right), \quad \forall i \in [d].$$

Recalling that components of $\hat{g}^n$ are mutually independent, we know

$$\langle \hat{g}^n - g, y - x \rangle = \sum_i \left(\hat{g}^n_{\alpha_x(i)} - g_{\alpha_x(i)}\right) \cdot (y - x)_{\alpha_x(i)} \sim \text{subGaussian}\left(\frac{2\sigma^2}{n} \cdot \|y - x\|_2^2\right).$$

Since $\|y - x\|_2^2 \leq dN^2$, we know

$$\langle \hat{g}^n - g, y - x \rangle \sim \text{subGaussian}\left(\frac{2dN^2\sigma^2}{n}\right).$$

By the Hoeffding bound, it holds

$$|\langle \hat{g}^n - g, y - x\rangle| \leq \sqrt{\frac{4dN^2\sigma^2}{n}\log\left(\frac{2}{\delta}\right)}$$

with probability at least $1-\delta$. If we choose

$$n = \left\lceil \frac{4dN^2\sigma^2}{\epsilon^2}\log\left(\frac{2}{\delta}\right)\right\rceil \leq \frac{4dN^2\sigma^2}{\epsilon^2}\log\left(\frac{2}{\delta}\right) + 1,$$

it follows that

$$|\langle \hat{g}^n - g, y - x\rangle| \leq \epsilon. \tag{EC.23}$$

Since $f(x)$ is a convex function and g is a subgradient at point x, we have $f(y) \geq f(x) + \langle g, y - x\rangle$ for all $y \in [1, N]^d$. Combining with inequality (EC.23) gives

$$f(y) \geq f(x) + \langle \hat{g}^n, y - x\rangle + \langle g - \hat{g}^n, y - x\rangle \geq f(x) + \langle \hat{g}^n, y - x\rangle - \epsilon, \quad \forall y \in [1, N]^d$$

holds with probability at least $1-\delta$. Then, considering the half space $H = \{y : \langle \hat{g}^n, y - x\rangle \leq 0\}$, it holds

$$f(y) \geq f(x) + \langle \hat{g}^n, y - x\rangle - \epsilon \geq f(x) - \epsilon, \quad \forall y \in [1, N]^d \cap H^c$$

with the same probability. Taking the minimum over $[1, N]^d \cap H^c$, it follows that the averaged stochastic subgradient provides an (ϵ, δ)-$\mathcal{SO}$ oracle. Finally, the expected simulation cost of each oracle evaluation is at most

$$d \cdot n \leq \frac{4d^2N^2\sigma^2}{\epsilon^2}\log\left(\frac{2}{\delta}\right) + d = \tilde{O}\left[\frac{d^2N^2}{\epsilon^2}\log\left(\frac{1}{\delta}\right)\right].$$

□

EC.6.2. Proof of Theorem 3

Before we provide the proof of Theorem 3, we show the calculation of the number of iterations T_{max}. With a slight abuse of notations, we use the same notations as Vaidya (1996) only in this calculation. Before the first iteration, we have the volumetric center as

$$\omega = \frac{N+1}{2} \cdot (1, \ldots, 1)^T \in \mathbb{R}^d.$$

Therefore, we can calculate that

$$H(\omega) = \frac{8}{(N-1)^2} \cdot I_d, \quad \rho^0 = \frac{d}{2}\log\left(\frac{8}{(N-1)^2}\right),$$

where I_d is the $d \times d$ identity matrix. By Vaidya (1996), the volume of the polytope at the beginning of the t-th iteration satisfies

$$\begin{aligned}\log(\pi^t) &\le d\log\left(\frac{2d}{\rho}\right) - \rho^0 - \frac{\rho}{2}\cdot t = d\log\left(\frac{2d}{\rho}\right) - \frac{d}{2}\log\left(\frac{8}{(N-1)^2}\right) - \frac{\rho}{2}\cdot t \\ &\le d\log\left(\frac{2d}{\rho}\right) + d\log\left(\frac{N}{2}\right) - \frac{\rho}{2}\cdot t = d\log\left(\frac{Nd}{\rho}\right) - \frac{\rho}{2}\cdot t. \end{aligned} \tag{EC.24}$$

The target set consists of points in the set

$$P(\epsilon) := \left\{x \in [1,N]^d \ : \ \|x - x^*\|_1 \le \epsilon/L\right\},$$

where x^* is the optimal solution of problem (1) and L is the Lipschitz constant of $f(\cdot)$. By a simple analysis, we know that the volume of $P(\epsilon)$ satisfies

$$\mathrm{vol}(P(\epsilon)) \ge \left(\frac{\epsilon}{L}\right)^d.$$

Therefore, we can terminate the algorithm when

$$\log(\pi^{T_{max}}) \le d\log\left(\frac{\epsilon}{L}\right).$$

Combining with inequality (EC.24), we know

$$T_{max} \ge \frac{2d}{\rho}\cdot\log\left(\frac{NdL}{\rho\epsilon}\right)$$

is sufficient for ϵ-approximate solutions.

Proof of Theorem 3. We first prove the correctness of Algorithm 4. If $\hat{g} = 0$ for some iteration, the half space $H = \mathbb{R}^d$ and the definition of $(\epsilon/8, \delta/4)$-$\mathcal{SO}$ implies that

$$f(y) \ge f(z) - \epsilon/8, \quad \forall y \in [1,N]^d$$

holds with probability at least $1 - \delta/4$, where z is the point that the separation oracle is called. Hence, we know z is an $(\epsilon/8, \delta/4)$-PGS solution and obviously satisfies the $(\epsilon/2, \delta/2)$-PGS guarantee. Then, by Theorem 4, the integral solution after the round process is an (ϵ, δ)-PGS solution.

In the following of the proof, we assume $\hat{g} \neq 0$ for all iterations. Let $x^* \in \mathcal{X}$ be a minimizer of problem (1). We consider the set

$$Q := \left(x^* + \left[-\frac{\epsilon}{8L}, \frac{\epsilon}{8L}\right]^d\right) \cap [1,N]^d.$$

We can verify that set Q is not empty and has volume at least $(\epsilon/(8L))^d$. Moreover, for any $x \in Q$, it holds

$$f(x) \le f(x^*) + L\|x - x^*\|_\infty \le f(x^*) + \frac{\epsilon}{8}.$$

By the analysis in Vaidya (1996), the volume of the polytope P is smaller than $(\epsilon/(8L))^d$ after

$$T_{max} := O\left[d\log\left(\frac{8dLN}{\epsilon}\right)\right]$$

iterations. Hence, after T_{max} iterations, the volume of P is smaller than the volume of Q and it must hold $Q\backslash P \neq \emptyset$. Since $Q \subset [1,N]^d$, the constraint $1 \leq x_i \leq N$ is not violated for all $i \in [d]$. Thus, if we choose $x \in Q\backslash P$, there exists a cutting plane $-\hat{g}^T y \geq \beta$ in P such that

$$-\hat{g}^T x < \beta \leq -\hat{g}^T z,$$

where z is the point that the $(\epsilon/8, \delta/4)$-$\mathcal{SO}$ oracle $\hat{g}$ is evaluated and β is the value chosen by Vaidya's method. This implies that x is not in the half space

$$H := \{y : \hat{g}^T y \leq \hat{g}^T z\}.$$

Then, by the definition of $(\epsilon/8, \delta/4)$-$\mathcal{SO}$ oracle and the claim that $x \in [1,N]^d \cap H^c$, we know

$$f(x) \geq f(z) - \epsilon/8$$

holds with probability at least $1 - \delta/4$. On the other hand, the condition $x \in P$ leads to

$$f(x) \leq f(x^*) + \epsilon/8.$$

Combining the last two inequalities gives that

$$\min_{y\in\mathcal{S}} \ f(y) \leq f(z) \leq f(x^*) + \epsilon/4$$

holds with probability at least $1-\delta/4$. Hence, the $(\epsilon/4, \delta/4)$-PGS solution $\hat{x}$ of problem $\min_{y\in\mathcal{S}} \ f(y)$ satisfies

$$f(\hat{x}) \leq f(x^*) + \epsilon/2$$

with probability at least $1 - \delta/2$. Equivalently, the solution $\hat{x}$ is an $(\epsilon/2, \delta/2)$-PGS solution. Using Theorem 4, the integral solution returned by Algorithm 4 is an (ϵ, δ)-PGS solution.

Now, we estimate the expected simulation cost of Algorithm 4. By Lemma 5, the simulation cost of each $(\epsilon/8, \delta/4)$-$\mathcal{SO}$ oracle is at most

$$O\left[\frac{d^2N^2}{\epsilon^2}\log\left(\frac{1}{\delta}\right) + d\right].$$

Since at most one separation oracle is evaluated in each iteration, the total simulation cost of T_{max} iterations is at most

$$O\left[\left(\frac{d^2N^2}{\epsilon^2}\log\left(\frac{1}{\delta}\right) + d\right)\cdot d\log\left(\frac{8dLN}{\epsilon}\right)\right] = \tilde{O}\left[\frac{d^3N^2}{\epsilon^2}\log(\frac{dLN}{\epsilon})\log\left(\frac{1}{\delta}\right)\right].$$

By the property of Vaidya's method, there are $O(d)$ cutting planes in the polytope P. Then, using the same analysis as Zhang et al. (2020), the expected simulation cost of finding an $(\epsilon/4, \delta/4)$-PGS solution of the sub-problem $\min_{y\in\mathcal{S}} f(y)$ is at most

$$\tilde{O}\left[\frac{d^2}{\epsilon^2}\log\left(\frac{1}{\delta}\right)\right].$$

We note that the evaluation of $(\epsilon/8, \delta/4)$-$\mathcal{SO}$ oracles at points in $\mathcal{S}$ provides enough simulations for the sub-problem and therefore the simulation cost of this part can be avoided. Finally, the expected simulation cost of the rounding process is bounded by

$$\tilde{O}\left[\frac{d}{\epsilon^2}\log\left(\frac{1}{\delta}\right)\right].$$

Combining the three parts, the total expected simulation cost of Algorithm 4 is at most

$$\tilde{O}\left[\frac{d^3N^2}{\epsilon^2}\log(\frac{dLN}{\epsilon})\log\left(\frac{1}{\delta}\right)\right].$$

□

EC.6.3. Proof of Theorem 4

Proof of Theorem 4. We first verify the correctness of Algorithm 5. If the optimal solution has been removed during the dimension reduction process, we claim that the optimal solutions are removed from the search set by some cutting plane. This is because the dimension reduction steps will not remove integral points from the current search set (Jiang 2021). Then, by the same proof as Theorem 3, it holds

$$\min_{x\in\mathcal{S}} f(x) \le \min_{x\in\mathcal{X}} f(x) + \epsilon/4 \tag{EC.25}$$

with probability at least $1-\delta/4$. Otherwise if the optimal solution has not been removed from the search set throughout the dimension reduction process, we know the last one-dimensional problem contains the optimal solution. Hence, the $(\epsilon/4, \delta/4)$-PGS solution to the one-dimensional problem is also an $(\epsilon/4, \delta/4)$-PGS solution to the original problem. Since the PGS solution is also added to the set $\mathcal{S}$, we also have relation (EC.25) holds with probability at least $1-\delta/4$. Then, the $(\epsilon/4, \delta/4)$-PGS solution $\bar{x}$ to problem $\min_{x\in\mathcal{S}} f(x)$ satisfies

$$f(\bar{x}) \le \min_{x\in\mathcal{X}} f(x) + \epsilon/2$$

with probability at least $1-\delta/2$, or equivalently $\bar{x}$ is an $(\epsilon/2, \delta/2)$-PGS solution to problem (1). Using the results of Theorem 4, the solution returned by Algorithm 5 is an (ϵ, δ)-PGS solution.

Next, we estimate the expected simulation cost of Algorithm 5. By the results in Jiang (2021), $(\epsilon/4, \delta/4)$-$\mathcal{SO}$ oracles are called at most $O[d(d+\log(N))]$ times. Hence, the size of $\mathcal{S}$ is at most

$O[d(d+\log(N))]$. By the estimates in Lemma 5, the total simulation cost of the dimension reduction process is at most

$$O\left[\frac{d^3N^2(d+\log(N))}{\epsilon^2}\log\left(\frac{1}{\delta}\right)+d^2(d+\log(N))\right]=\tilde{O}\left[\frac{d^3N^2(d+\log(N))}{\epsilon^2}\log\left(\frac{1}{\delta}\right)\right].$$

Moreover, the one-dimensional convex problem has at most N feasible points and Theorem 2 implies that the expected simulation cost for this problem is at most

$$\tilde{O}\left[\frac{1}{\epsilon^2}\log\left(\frac{1}{\delta}\right)\right].$$

Since the size of $\mathcal{S}$ is at most $O[d(d+\log(N))]$, the sub-problem for the set $\mathcal{S}$ takes at most

$$O\left[\frac{d^2(d+\log(N))}{\epsilon^2}\log\left(\frac{1}{\delta}\right)+d^2(d+\log(N))\right]=\tilde{O}\left[\frac{d^2(d+\log(N))}{\epsilon^2}\log\left(\frac{1}{\delta}\right)\right]$$

simulation runs. Finally, Theorem 4 shows that the expected simulation cost of the rounding process is at most

$$\tilde{O}\left[\frac{d}{\epsilon^2}\log\left(\frac{1}{\delta}\right)\right].$$

In summary, the total expected simulation cost of Algorithm 5 is at most

$$\tilde{O}\left[\frac{d^3N^2(d+\log(N))}{\epsilon^2}\log\left(\frac{1}{\delta}\right)\right].$$

□

EC.7. Adaptive Sub-Gaussian Parameter Estimator

In this section, we provide a simple adaptive mean estimator to adaptively estimate the variance of each choice of decision variable under the assumption that the distribution of the randomness is Gaussian. The estimator can be used to further enhance our proposed algorithm and we hope the procedure to be useful for other optimization via simulation problems and algorithms that do not know the variances a priori. Using the adaptive estimator, the prior knowledge about the upper bound on the variance σ^2 is not necessary. In addition, for the multi-dimensional localization algorithms proposed in this work, the simulation cost for the unknown variance case is at most a constant factor larger than the case when an upper bound on the variance is known a priori. Therefore, the algorithm using the adaptive estimator, or the adaptive algorithm, is able to improve the performance of our proposed algorithms if an estimate of the upper bound σ^2 is much larger than the true variance. In this case, the original algorithms will implement an unnecessarily large number of simulation runs to shrink the confidence interval, while the adaptive algorithm is able to automatically learn the true variance and thus save the computational cost. Another situation where the adaptive algorithm is useful is when the variance of the system varies a lot at different

choices of decision variable. In this case, the upper bound of the variance is usually attained at extreme choices of decision variable and is much larger than the variance of a majority of feasible choices. For example, in queueing systems, the distribution of arrival times usually follows the Poisson distribution (or the generalizations of Poisson distributions, such as the jump distribution). The mean and the variance of the Poisson distribution are equal and, thus, the variance is large at solutions where the mean is large. The queueing system contains the Poisson process as a part and will also exhibit certain heteroscedasticity in the variance. Using the upper bound at all points leads to a conservative mean estimator; see our experiment results in Table EC.1.

We now state the proposed adaptive mean estimator. To increase the generality of our results, we make a weaker assumption than the Gaussian case.

ASSUMPTION EC.1. *The distribution of $F(x,\xi_x)-f(x)$ belongs to the family of sub-Gaussian distributions $\mathcal{F}_\kappa$, where $\kappa>0$ is a known constant. For any random variable X whose distribution belongs to $\mathcal{F}_\kappa$, it holds that*

$$\kappa\sigma_X^2 \le \mathrm{Var}(X), \tag{EC.26}$$

where σ_X^2 is the sub-Gaussian parameter of the distribution.

We note that the inverse inequality $\mathrm{Var}(F(x,\xi_x))\le\sigma_x^2$ always holds for all sub-Gaussian distributions. However, there does not exist a universal constant $\kappa>0$ such that inequality (EC.26) holds for all sub-Gaussian distributions. Therefore, Assumption EC.1 cannot be implied by Assumption 2 and additional prior knowledge about the distribution is required for the estimation of κ. We provide three special cases when the value of κ can be estimated:

1. Suppose that the distribution of $F(x,\xi_x)$ is Gaussian. In this case, the constant $\kappa=1$, i.e., we have the relation $\sigma_x^2=\mathrm{Var}[F(x,\xi_x)]$.

2. Suppose that the distribution of $F(x,\xi_x)$ is the uniform distribution in the interval $[f(x)-a_x, f(x)+a_x]$, where the values of $f(x)$ and a_x are unknown. In this case, the variance and the sub-Gaussian parameter of $F(x,\xi_x)-f(x)$ are $a_x^2/3$ and a_x, respectively. Therefore, the parameter κ is equal to $1/3$ in this case.

3. Suppose that the distribution of $F(x,\xi_x)$ is the Bernoulli distribution with the parameters (n_x,p_x), where the value of p_x is unknown. In this case, the variance and the sub-Gaussian parameter of $F(x,\xi_x)-f(x)$ are $p_x(1-p_x)$ and $\frac{1-2p_x}{2\log[(1-p_x)/p_x]}$ (Ostrovsky and Sirota 2014), respectively. Therefore, we can show that either the sub-Gaussian parameter is at most $1/2$ or the parameter κ is $(e-1)/(2e^2-4e)$. The analysis of the Bernoulli case can be directly extended to a binomial distribution with the parameters (n_x,p_x), where an upper bound on the parameter n_x is known (for example, we know a priori that $F(x,\xi_x)$ belongs to $\{0,1,\ldots,M\}$ for some integer $M>0$).

Under the above assumption, we propose the adaptive mean estimator.

DEFINITION EC.2. Let $\epsilon > 0$ be the precision and $\delta \in (0, 1]$ be the failing probability. We construct the adaptive mean estimator of $f(x)$ in two steps:

1. Sample $2n$ independent evaluations $F(x, \xi_i)$ for $i \in [2n]$, where $n := \lceil 256\kappa^{-2} \log(2/\delta) \rceil$. Compute the variance estimator

$$\hat{\mathrm{Var}}_x := \frac{1}{n} \sum_{i=1}^{n} [F(x, \xi_{2i-1}) - F(x, \xi_{2i})]^2$$

and the parameter estimator

$$\hat{\sigma}_x^2 := \frac{1}{\kappa} \hat{\mathrm{Var}}_x.$$

2. Let $m := \max\{\lceil 2\epsilon^{-2} \hat{\sigma}_x^2 \log(2/\delta) \rceil, 2n\}$ and sample $m - 2n$ independent evaluations $F(x, \xi_{2n+i})$ for $i \in [m - 2n]$ and compute the empirical mean

$$\hat{F}(x; \delta) := \frac{1}{m} \sum_{i=1}^{m} F(x, \xi_i).$$

The construction of the adaptive mean estimator has two steps. In the first step, we estimate an upper bound for the sub-Gaussian parameter, and in the second step, we use the estimated upper bound to calculate the required number of simulation so that the sub-Gaussian parameter is less than a known constant. The purpose of the choice of $\hat{\mathrm{Var}}$ is to utilize the Bernstein bound on the lower tail of sum of squared sub-Gaussian random variables, i.e., $\mathbb{P}\left(\sum_{i=1}^{n} Z_i^2 \le a\sigma^2\right)$, where $Z_1, \ldots, Z_n$ are independent sub-Gaussian random variables with parameter σ^2 and $a > 0$ is a constant. If we use the common estimator of variance $n^{-1} \sum_i (Z_i - \bar{Z})^2$, where $\bar{Z}$ is the empirical mean, the random variables $Z_1 - \bar{Z}, \ldots, Z_n - \bar{Z}$ are not necessarily independent and the Bernstein bound cannot be applied. We note that the adaptive mean estimator is an online estimator. To be more concrete, if a smaller precision $\epsilon' < \epsilon$ is required, it suffices to add

$$\lceil 2(\epsilon')^{-2} \hat{\sigma}_x^2 \log(2/\delta) \rceil - \lceil 2\epsilon^{-2} \hat{\sigma}_x^2 \log(2/\delta) \rceil$$

more evaluations into the empirical mean in step 2. The following theorem verifies that $\hat{F}(\cdot; \delta)$ is an unbiased mean estimator for $f(x)$ and its tail is sub-Gaussian with a small failing probability.

THEOREM EC.7. *Suppose that Assumption EC.1 holds. Let $\delta \in (0, 1]$ be the failing probability. For all $\epsilon \ge 0$, the adaptive mean estimator satisfies*

$$\mathbb{P}\left[|\hat{F}(x; \delta) - f(x)| \ge \epsilon\right] \le \delta, \tag{EC.27}$$

In addition, the expected simulation cost of the adaptive mean estimator is $O[(\kappa^{-2} + \kappa^{-1}\epsilon^{-2}\sigma_x^2) \log(1/\delta)]$.

Proof of Theorem EC.7. The proof is provided in EC.7.1. □

REMARK EC.1. We note that the estimator $\hat{\sigma}_x^2$ in Definition EC.2 is the sum of squared sub-Gaussian random variables and may have a heavy tail. Thus, the simulation cost of $\hat{F}(x;\delta)$ may also have a heavy tail. To deal with this issue, we can utilize the Median-of-Mean (MoM) estimator (Lerasle 2019) of the random variable

$$G_x := \kappa^{-1}[F(x,\xi_1) - F(x,\xi_2)]^2,$$

where ξ_1 and ξ_2 are independent. Choosing $b := \lceil 2^{15}\kappa^{-2} \rceil$ and $K := \lceil 2\log(2/\delta) \rceil$ in the MoM estimator, we define the alternative estimator $(\hat{\sigma}_x^{MoM})^2$ by

$$(\hat{\sigma}_x^{MoM})^2 := \kappa^{-1}\text{median}\left\{ \frac{1}{b}\sum_{j=1}^{b} G_{x,ib+j},\ i \in [K] \right\}$$

where $G_{x,1}, \ldots, G_{x,Kb}$ are independent samples of G_x. The estimator $(\hat{\sigma}_x^{MoM})^2$ also has simulation cost $O[\kappa^{-2}\log(1/\delta)]$. Using Proposition 12 in Lerasle (2019), we know that the estimator $(\hat{\sigma}_x^{MoM})^2$ satisfies

$$\mathbb{P}\left[\sigma_x^2 \le (\hat{\sigma}_x^{MoM})^2 \le (2+\kappa^{-1})\sigma_x^2\right] \ge 1 - \delta/2.$$

Using this MoM estimator, we are able to bound the simulation cost of $\hat{F}(x;\delta)$ in high probability.

If the sub-Gaussian parameter σ_x^2 is known, the Hoeffding bound shows that

$$O[\epsilon^{-2}\sigma_x^2\log(1/\delta)]$$

samples are sufficient to generate an estimator for inequality (EC.27). Therefore, the relative efficiency of the adaptive mean estimator is

$$\frac{\epsilon^{-2}\sigma_x^2}{\kappa^{-2} + \epsilon^{-2}\kappa^{-1}\sigma_x^2} = \frac{1}{\kappa^{-1} + \kappa^{-2}\epsilon^2\sigma_x^{-2}}.$$

If the precision ϵ is small or the parameter σ_x^2 is large, the adaptive mean estimator is only a constant (κ) time less efficient than the known variance case.

Now, we estimate the expected simulation cost of our proposed simulation-optimization algorithms combined with the adaptive estimator. Intuitively, we need to implement the first step in Definition EC.2 once for all simulated choices of decision variable. Suppose that a simulation-optimization algorithm simulates $N(\epsilon,\delta)$ different choices of decision variable in expectation and the expected simulation cost is $T(\epsilon,\delta)$. Then, the expected simulation cost of the adaptive simulation-optimization algorithm is

$$O\left[\kappa^{-1}T(\epsilon,\delta) + \kappa^{-2}N(\epsilon,\delta)\log\left(N(\epsilon,\delta)/\delta\right)\right].$$

For the localization algorithms, we usually have $T(\epsilon,\delta) = O[N(\epsilon,\delta)\log\left(N(\epsilon,\delta)/\delta\right)]$. Therefore, the expected simulation cost of the localization algorithms is $O[T(\epsilon,\delta)]$. More concretely, we have the following corollary.

COROLLARY EC.2. *Suppose that Assumptions 1, 3-EC.1 hold. The following estimates hold:*

- *The expected simulation cost of the adaptive TS algorithm (Algorithm 1) is*

$$O\left[(\kappa^{-2}+\kappa^{-1}\sigma^2\epsilon^{-2})\log(N)\log\left(\frac{\log(N)}{\delta}\right)\right]=\tilde{O}\left[(\kappa^{-2}+\kappa^{-1}\sigma^2\epsilon^{-2})\log(N)\log\left(\frac{1}{\delta}\right)\right].$$

- *The expected simulation cost of the adaptive SUS algorithm (Algorithm 2) is*

$$O\left[(\kappa^{-2}N+\kappa^{-1}\sigma^2\epsilon^{-2})\log\left(\frac{N}{\delta}\right)\right]=\tilde{O}\left[(\kappa^{-2}N+\kappa^{-1}\sigma^2\epsilon^{-2})\log\left(\frac{1}{\delta}\right)\right].$$

- *The expected simulation cost of the adaptive stochastic cutting-plane algorithm (Algorithm 4) is*

$$\begin{aligned}&O\left[\left(\kappa^{-2}+\frac{\kappa^{-1}\sigma^2dN^2}{\epsilon^2}\right)\cdot d^2\log\left(\frac{dLN}{\epsilon}\right)\log\left(\frac{1}{\delta}\right)+\kappa^{-2}d^2\log\left(\frac{dLN}{\epsilon}\right)\log\left(d^2\log\left(\frac{dLN}{\epsilon}\right)\right)\right]\\ =&\tilde{O}\left[\left(\kappa^{-2}+\frac{\kappa^{-1}\sigma^2dN^2}{\epsilon^2}\right)\cdot d^2\log\left(\frac{dLN}{\epsilon}\right)\log\left(\frac{1}{\delta}\right)\right].\end{aligned}$$

- *The expected simulation cost of the adaptive dimension reduction algorithm (Algorithm 5) is*

$$\begin{aligned}&O\left[\left(\kappa^{-2}+\frac{\kappa^{-1}\sigma^2dN^2}{\epsilon^2}\right)d^2(d+\log(N))\log\left(\frac{1}{\delta}\right)+\kappa^{-2}d^2(d+\log(N))\log\left(d^2(d+\log(N))\right)\right]\\ =&\tilde{O}\left[\left(\kappa^{-2}+\frac{\kappa^{-1}\sigma^2dN^2}{\epsilon^2}\right)d^2(d+\log(N))\log\left(\frac{1}{\delta}\right)\right].\end{aligned}$$

We can see that the expected simulation cost of the stochastic cutting-plane method and the dimension reduction algorithm is only increased by a factor. Therefore, the adaptive mean estimator is useful in dropping the requirement of known parameter σ for the multi-dimensional case. For the one-dimensional case, the expected simulation cost of the tri-section algorithm is also increased by a constant factor. On the other hand, the cost of the adaptive SUS algorithm is larger than the original version, especially when $\epsilon^{-2}\ll N$. Therefore, in the one-dimensional unknown variance case, we need to consider the size of ϵ to decide whether to use the TS algorithm or the SUS algorithm, More specifically, if $\epsilon=O(\sqrt{\log N/N})$, then the SUS algorithm is preferred; otherwise the TS algorithm is preferred.

Now, we briefly discuss how to apply the dimension reduction algorithm and the adaptive sub-Gaussian parameter estimator to the two examples in Section 5, and we present the effects of the adaptive sub-Gaussian parameter estimator on the optimal allocation problem. We first consider the estimation of the parameter κ. For the synthetic example, the noise is Gaussian and, thus, we have $\kappa=1$. For the queueing example, the dimension reduction algorithm will simulate $\Theta[\hat{\sigma}_x^2dN^2/\epsilon^2\log(1/\delta)]$ independent samples at each iteration point x (more rigorously, the neighbouring points of point x) to estimate the expected value $f(x)$, where $\hat{\sigma}_x^2$ is chosen to be $30\sqrt{N}\gg 1$ in the numerical experiments. Define $n:=dN^2/\epsilon^2\log(1/\delta)$ and

$$G(x,\bar{\xi}_x):=\frac{1}{n}\sum_{i=1}^{n}F(x,\xi_{x,i}),$$

where $\xi_{x,1},\ldots,\xi_{x,n}$ are independently sampled and $\bar{\xi}_x := (\xi_{x,1},\ldots,\xi_{x,n})$. Then, the sub-Gaussian parameter of $F(x,\xi_x)-f(x)$ should be at most n times larger than the sub-Gaussian parameter of $G(x,\bar{\xi}_x)-f(x)$; the same relation also holds for the variances. Hence, we can instead estimate the constant κ of random variable $G(x,\bar{\xi}_x)$. Using the Central Limit Theorem (CLT), the asymptotic behavior of the empirical mean of $F(x,\xi_x)$ is Gaussian. Indeed, with our choice of n, the distribution of $G(x,\bar{\xi}_x)$ is already very close to the Gaussian distribution. To verify this claim, we plot the Quantile-Quantile plot for 200 independent samples of $G(x_0,\bar{\xi}_{x_0})$ and quantiles of the Gaussian distribution, where $x_0 = (\frac{N+1}{2}, N+1, \ldots, \frac{d(N+1)}{2})$. The results of $(d,N) = (4,10),(4,50),(24,10)$ are plotted in Figure EC.1 and we can see that the distributions in all cases are very close to the Gaussian distribution. Therefore, the parameter κ is approximately equal to 1 for $G(x,\bar{\xi}_x)$.

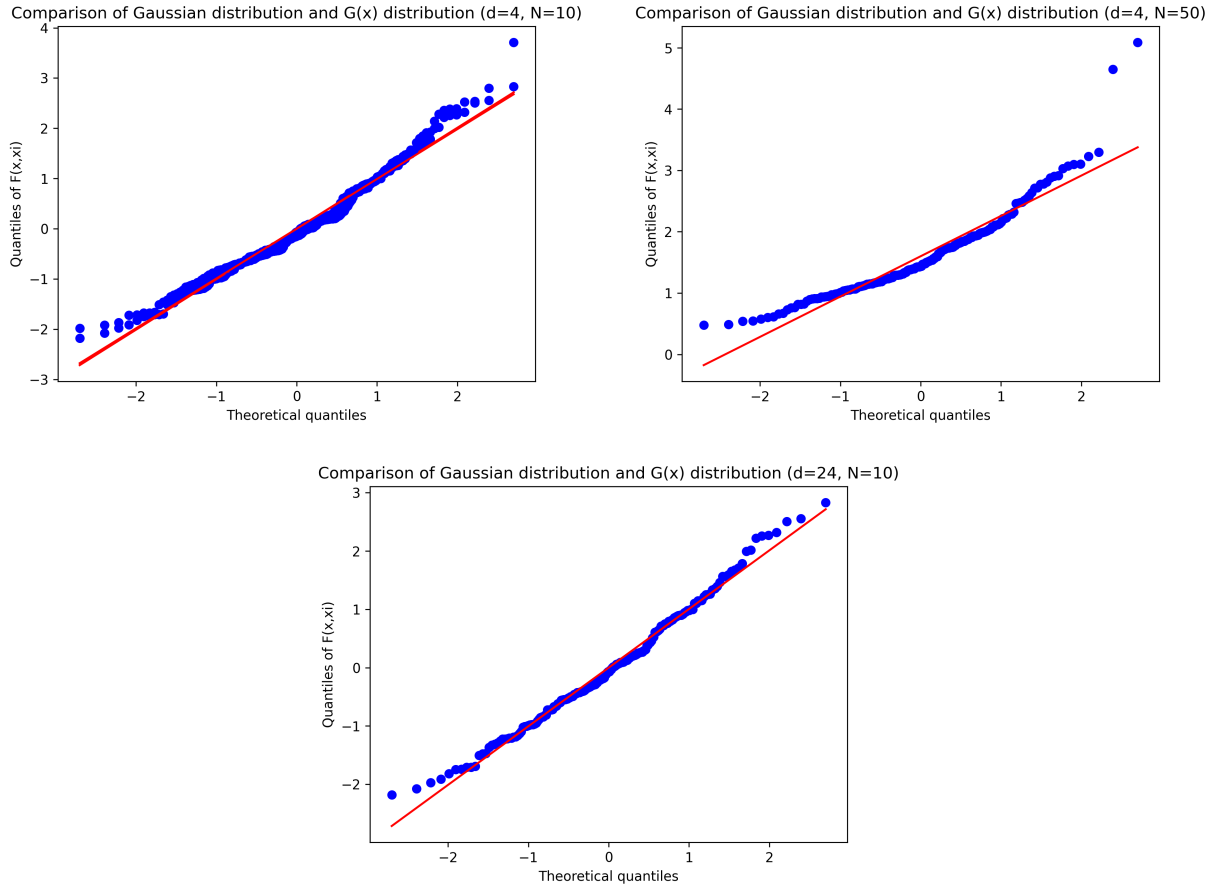


Figure EC.1 The Quantile-Quantile plots of the distribution of $G(x,\bar{\xi}_x)$ in the cases when $(d,N) = (4,10),(4,50),(24,10)$.

Hence, we use the first $50n \le 30\sqrt{N}n$ samples of $F(x,\xi_x)$ to generate 25 pairs of evaluations of

$G(x, \bar{\xi}_x)$. Here, the sample size 25 is derived from the estimate that $\kappa^{-2}\log[d^2(d+\log N)/\delta] = \log[d^2(d+\log N)/\delta] \le 25$. The sub-Gaussian parameter of $F(x,\xi_x) - f(x)$ is then estimated by

$$\hat{\sigma}_x^2 := \frac{n}{25}\sum_{i=1}^{25}\left[G(x,\bar{\xi}_{x,2i-1}) - G(x,\bar{\xi}_{x,2i})\right]^2.$$

Finally, since the failing probability in Theorem EC.7 is bounded by the union bound, we only need to take $\max\{\hat{\sigma}_x^2 - 50, 0\}n$ extra samples of $F(x,\xi_x)$ to generate the mean estimator with the desired confidence level.

We test the dimension reduction algorithm with the adaptive sub-Gaussian parameter estimator on the optimal allocation example under the same setting as in Section 5 and the results are summarized in Table EC.1. From the numerical results, we can see that the adaptive sub-Gaussian parameter estimator is able to reduce the expected simulation cost in most cases. This is because the maximum variance is usually attained by decisions around the global minimum. Thus, during the early stage of the optimization process, the true sub-Gaussian parameter is relatively small and is overestimated by our estimation $\hat{\sigma}^2 = 30\sqrt{N}$. Using the adaptive estimator, we are able to estimate the sub-Gaussian parameter and reduce the expected simulation cost.

Table EC.1 Simulation cost of the dimension reduction algorithm with and without the adaptive variance estimator on the resource allocation problem.

Params.		No adaptive estimator		With adaptive estimator	
d	N	Cost	Obj.	Cost	Obj.
4	10	2.42e4	2.40e1	**1.22e4**	2.42e1
4	20	1.40e4	3.44e1	**1.06e4**	3.42e1
4	30	9.21e3	4.59e1	**8.21e3**	4.15e1
4	40	6.31e3	5.75e1	**4.31e3**	5.75e1
4	50	**4.03e3**	6.67e1	6.03e3	6.70e1
8	10	1.48e5	2.12e1	**1.55e4**	2.21e1
12	10	6.10e5	2.01e1	**4.72e4**	2.11e1
16	10	1.59e6	1.91e1	**3.65e5**	1.92e1
20	10	3.21e6	1.81e1	**7.83e5**	1.88e1
24	10	8.54e6	1.76e1	**1.81e6**	1.81e1

EC.7.1. Proof of Theorem EC.7

We first prove that $\hat{\sigma}$ serves as an upper bound on the sub-Gaussian parameter. The proof is based on the property that the lower tail of a squared sub-Gaussian random variable is sub-Gaussian.

LEMMA EC.11. *Let $\delta \in (0,1]$ be the failing probability. The parameter estimator $\hat{\sigma}^2$ in Definition EC.2 satisfies*

$$\mathbb{P}\left(\hat{\sigma}^2 \le \sigma_x^2\right) \le \delta/2.$$

Proof of Lemma EC.11. By the definitions of $\hat{\text{Var}}$ and $\hat{\sigma}^2$, we only need to prove that

$$\mathbb{P}\left(\frac{1}{n}\sum_{i=1}^{n}\left[F(x,\xi_{2i-1})-F(x,\xi_{2i})\right]^2 \le \kappa\sigma_x^2\right) \le \delta/2. \tag{EC.28}$$

By the independence between ξ_{2i-1} and ξ_{2i}, the random variable $F_i := F(x,\xi_{2i-1}) - F(x,\xi_{2i})$ is zero-mean and sub-Gaussian with parameter $2\sigma_x^2$ for all $i \in [n]$. Using the fact that $-F_i^2 \le 0$ almost surely and the one-sided Bernstein's inequality, we have

$$\mathbb{P}\left[\frac{1}{n}\sum_{i=1}^{n}\left(-F_i^2+\mathbb{E}(F_i^2)\right) \ge \kappa\sigma_x^2\right] \le \exp\left[-\frac{n\kappa^2\sigma_x^4}{2/n\sum_{i=1}^{n}\mathbb{E}(F_i^4)}\right].$$

Since $\{F_i, i\in[n]\}$ are i.i.d. and zero-mean random variables, the above inequality is equivalent to

$$\mathbb{P}\left[\frac{1}{n}\sum_{i=1}^{n}F_i^2 - \text{Var}(F_1) \le -\kappa\sigma_x^2\right] \le \exp\left[-\frac{n\kappa^2\sigma_x^4}{2\mathbb{E}(F_1^4)}\right].$$

Now, recalling the assumption in (EC.26) and $\text{Var}(F_1) = 2\text{Var}(F(x,\xi_1))$, we get

$$\mathbb{P}\left[\frac{1}{n}\sum_{i=1}^{n}F_i^2 - 2\kappa\sigma_x^2 \le -\kappa\sigma_x^2\right] \le \mathbb{P}\left[\frac{1}{n}\sum_{i=1}^{n}F_i^2 - \text{Var}(F_1) \le -\kappa\sigma_x^2\right] \le \exp\left[-\frac{n\kappa^2\sigma_x^4}{2\mathbb{E}(F_1^4)}\right]. \tag{EC.29}$$

To estimate the fourth moment of F_1, we calculate that

$$\begin{aligned}\mathbb{E}(F_1^4) &= \int_0^\infty t\mathbb{P}(F_1^4 \ge t)\ dt = \int_0^\infty 4s^3\mathbb{P}(F_1^4 \ge s^4)\ ds = \int_0^\infty 4s^3\mathbb{P}(|F_1| \ge s)\ ds \\ &\le \int_0^\infty 4s^3\cdot 2\exp[-s^2/(8\sigma_x^2)]\ ds = 128\sigma_x^4,\end{aligned}$$

where the second equality is from the substitution $t = s^4$ and the last inequality is from the fact that F_1 is $2\sigma_x^2$-sub-Gaussian. Substituting into inequality (EC.29), we get

$$\mathbb{P}\left[\frac{1}{n}\sum_{i=1}^{n}F_i^2 \le \kappa\sigma_x^2\right] \le \exp\left(-\frac{n\kappa^2\sigma_x^4}{256\sigma_x^4}\right) = \exp\left(-\frac{n\kappa^2}{256}\right) \le \frac{\delta}{2},$$

where the last inequality is from the choice of n. The above inequality is equivalent to inequality (EC.28) and the proof is done. □

With the help of Lemma EC.11, we now prove the theorem.

Proof of Theorem EC.7. By Lemma EC.11, we know that the event $\mathcal{E} := \{\sigma_x^2 \le \hat{\sigma}^2\}$ happens with probability at least $1-\delta/2$. By the Hoeffding's inequality and the definitions of n and m, we have

$$\begin{aligned}\mathbb{P}\left[|\hat{F}(x;\delta) - f(x)| \ge \epsilon, \mathcal{E} \mid \hat{\sigma}^2\right] &\le \exp\left[-\frac{m\epsilon^2}{2\sigma_x^2}\right] \le \exp\left[-\frac{2\epsilon^{-2}\hat{\sigma}^2\cdot\log(2/\delta)\epsilon^2}{2\sigma_x^2}\right] \\ &\le \exp\left[-\log(2/\delta)\epsilon^2\right] = \frac{\delta}{2}.\end{aligned}$$

Taking expectation over $\hat{\sigma}^2$ leads to

$$\mathbb{P}\left[|\hat{F}(x;\delta) - f(x)| \geq \epsilon, \mathcal{E}\right] \leq \frac{\delta}{2}.$$

Therefore, we get

$$\begin{aligned}\mathbb{P}\left[|\hat{F}(x;\delta) - f(x)| \geq \epsilon\right] &= \mathbb{P}\left[|\hat{F}(x;\delta) - f(x)| \geq \epsilon, \mathcal{E}\right] + \mathbb{P}\left[|\hat{F}(x;\delta) - f(x)| \geq \epsilon, \mathcal{E}^c\right] \\ &\leq \mathbb{P}\left[|\hat{F}(x;\delta) - f(x)| \geq \epsilon, \mathcal{E}\right] + \mathbb{P}\left[\mathcal{E}^c\right] \leq \delta,\end{aligned}$$

where $\mathcal{E}^c$ is the complementary set of $\mathcal{E}$.

The estimation of the expected simulation cost is from the fact that $\hat{\text{Var}}$ is an unbiased estimator of $\text{Var}[f(x,\xi_x)]$. Therefore, we get the bound

$$\mathbb{E}[m] \leq \max\{\lceil 2\epsilon^{-2}\mathbb{E}(\hat{\sigma}^2)\log(2/\delta)\rceil, \lceil 512\kappa^{-2}\log(2/\delta)\rceil\} \leq \lceil (512\kappa^{-2} + 2\kappa^{-1}\epsilon^{-2}\sigma_x^2)\log(2/\delta)\rceil.$$

□

EC.8. Additional Numerical Experiments

EC.8.1. Comparison to Industrial Strength COMPASS

In this subsection, we compare the dimension reduction algorithm (Algorithm 5), the subgradient descent algorithm (Zhang et al. 2020), and the Industrial Strength COMPASS (ISC) algorithm (Xu et al. 2010) on multi-dimensional optimization via simulation problems. For the dimension reduction algorithm and the subgradient descent algorithm, we use the results in Section 5. For the ISC algorithm, our experiments are based on the source codes provided by the authors of Xu et al. (2010). We implemented the ISC codes on a C++17 compiler and used the *LpSolve* package with version 5.5.0.3. Since the ISC algorithm does not support the PGS criterion, we empirically set the value of the CLEANUP_DELTA parameter so that the ISC algorithm finds a comparable solution to other algorithms. To be more concrete, we choose CLEANUP_DELTA to be $d/10$ in the separable convex function minimization problem and $N/100$ in the optimal allocation problem. Other parameters are set to their default values. We test the performance of the ISC algorithm on 100 independent experiments for the separable convex function minimization and 5 independent experiments for the optimal allocation problem.

The results are summarized in Tables EC.2 and EC.3. For the separable convex function minimization problem, the coverage rate in all settings is equal to 100%, and the proposed dimension reduction methods is always better than the ISC algorithm. For the optimal allocation problem, we can see that the proposed dimension reduction methods is better when the scale is large (e.g., when $N \geq 20$) or when the dimension is large (e.g., when $d \geq 20$). This is consistent with the main

Table EC.2 Simulation cost of different algorithms on separable convex functions.

Params.		SubGD	Dim Reduction	ISC	R-SPLINE
d	N	Cost	Cost	Cost	Cost
2	50	1.08e3	1.56e2	8.73e2	**4.50e1**
2	500	2.54e4	2.08e2	9.00e2	**9.00e1**
2	5000	3.97e5	4.66e2	1.05e3	**1.84e2**
6	50	5.00e3	4.05e2	4.06e3	**1.60e2**
6	500	4.75e4	6.45e2	8.86e3	**3.20e2**
6	5000	2.72e6	8.25e2	1.20e4	**8.25e2**
10	50	8.46e3	8.34e2	3.69e4	**3.32e2**
10	500	6.32e4	1.48e3	6.71e4	**8.88e2**
10	5000	7.76e6	2.02e3	1.08e5	**1.62e3**
15	50	1.23e4	2.18e3	2.19e5	**8.72e2**
15	500	2.83e5	3.19e3	4.26e5	**1.94e3**
15	5000	1.85e7	4.85e3	1.16e6	**3.40e3**

Table EC.3 Simulation cost and objective value of different algorithms on the resource allocation problem.

Params.		SubGD		Dim Reduction		ISC		R-SPLINE	
d	N	Cost	Obj.	Cost	Obj.	Cost	Obj.	Cost	Obj.
4	10	3.06e5	2.13e1	2.42e4	2.40e1	1.01e5	2.18e1	**1.21e4**	2.23e1
4	20	1.08e5	3.41e1	**1.40e4**	3.44e1	6.47e4	3.41e1	2.80e4	3.43e1
4	30	7.79e4	4.59e1	**9.21e3**	4.59e1	7.27e4	4.79e1	1.66e4	4.71e1
4	40	5.06e4	5.73e1	**6.31e3**	5.75e1	8.95e4	5.63e1	1.07e4	5.73e1
4	50	4.50e4	6.91e1	**4.03e3**	6.67e1	9.79e4	6.66e1	6.86e3	6.75e1
8	10	1.20e6	2.01e1	**1.48e5**	2.12e1	1.54e5	2.13e1	5.33e5	2.17e1
12	10	2.69e6	1.90e1	6.10e5	2.01e1	**3.02e5**	2.09e1	1.71e6	5.84e1
16	10	4.78e6	1.83e1	1.59e6	1.91e1	**1.16e6**	1.88e1	4.29e6	7.13e1
20	10	7.45e6	1.78e1	**3.21e6**	1.81e1	6.51e6	1.78e1	9.95e6	8.05e1
24	10	1.43e7	1.71e1	**8.54e6**	1.76e1	2.42e7	2.67e1	2.48e7	8.42e1

contribution of this work, i.e., efficient simulation-optimization algorithms for large-scale problems. However, the ISC algorithm achieves better simulation costs for high-dimensional problems with a relatively small scale or dimension. There are two possible reasons for this phenomenon.

First, the ISC algorithm only finds "locally optimal solutions". Specifically, in Xu et al. (2010), a solution $x \in \mathcal{X}$ is called a locally optimal solution if $f(y) \geq f(x)$ for all $y \in \mathcal{X}$ such that $\|y-x\|_1 \leq 1$. For $L^\natural$-convex functions, a locally optimal solution is not necessarily a globally optimal solution. Instead, a solution $x \in \mathcal{X}$ is a global optimum of a $L^\natural$-convex function if and only if $f(y) \geq f(x)$ for all $y \in \mathcal{X}$ such that $\|y-x\|_\infty \leq 1$. Therefore, to check the global optimality of a solution, the algorithm needs to estimate the objective values of 3^d-1 neighbouring points. This requires considerably more computational efforts compared to the ISC algorithm that only checks $2d$ neighbouring points to verify the local optimality. Even in the case when $d=8$, this will lead to 400 times more simulations. For the optimal allocation example, the ISC algorithm simulates the neighbouring points of each potential solution at least 20 times to guarantee the targeted confidence level of statistical tests, which will lead to 1.31×10^5 extra simulations for each potential solution. On the other hand, our

proposed algorithms are able to find globally optimal solutions for $L^\natural$-convex functions and, thus, our proposed algorithms provides a stronger theoretical guarantee.

Second, the implementation of the ISC algorithm is highly optimized to reduce the simulation costs in large-scale industrial applications. As a comparison, our implementation of the algorithms proposed in this paper are not optimized to achieve the best performance, since the purpose of our codes is to compare the performance of our algorithms and verify our theoretical analysis. We believe that the simulation costs of our proposed algorithms can be further reduced by using an improved implementation. For example, we can use different estimated variances at different solutions to reduce the simulation costs (since the simulation costs can be lower at solutions whose simulation output has a lower variance).

In summary, the ISC algorithm achieves a better empirical performance on some experiments but provides a weaker theoretical guarantee. Our proposed algorithms, on the other hand, may be inferior in certain cases but have a better performance in the large-scale case and provide a stronger theoretical guarantee.

EC.8.2. Comparison to R-SPLINE

In this subsection, we compare the dimension reduction algorithm (Algorithm 5), the subgradient descent algorithm (Zhang et al. 2020), and the R-SPLINE algorithm (Wang et al. 2013). For the R-SPLINE algorithm, we choose the maximal number of retrospective iterations to be the maximal budget, the initial sample size to be 10 and the initial spline budget to be 10. Other parameters are chosen to be their default values. For the separable convex function optimization problem, we require that the returned solutions of all experiments have objective function values at most d; for the optimal allocation problem, we require that the average of the estimated objective function value of the returned solution not be larger than that of the dimension reduction algorithm. Since the R-SPLINE algorithm only supports the fixed-budget optimization via simulation, we set the budget of the R-SPLINE algorithm to be the minimum multiple of $\lfloor 0.1B \rfloor$ such that the aforementioned condition is satisfied, where B is the average simulation cost of the dimension reduction algorithm in this setting. In addition, we cap the maximum budget at $5B$. We implement 100 independent experiments for the separable convex function minimization and 20 independent experiments for the optimal allocation problem. The R-SPLINE codes are implemented using MATLAB 2020a.

The results are summarized in Tables EC.2 and EC.3. For the separable convex function minimization problem, the R-SPLINE algorithm achieves the best performance. Similar to the ISC algorithm, this is because the R-SPLINE algorithm only checks the local optimality by estimating

the objective function values of $2d$ neighbouring points. In the separable convex function minimization problem, the locally optimal solution happens to be the globally optimal solution and, thus, the R-SPLINE algorithm can achieve the best performance. We note that the growth of the simulation cost of the R-SPLINE algorithm is faster than that of the dimension reduction algorithm when the scale N becomes larger. Hence, we expect that the dimension reduction algorithm will be better when the problem scale is very large.

For the optimal allocation problem, the R-SPLINE algorithm has a difficulty in reaching a global solution when the dimension is larger than 4. This is also because the R-SPLINE only checks the local optimality condition and may get stuck at locally optimal solutions that are not global optimal. By comparing with the performance of the dimension reduction algorithm, we can see that our proposed algorithms can provide a stronger theoretical guarantee for $L^\natural$-convex functions.

EC.8.3. Numerical Results with Small Precision Parameter

In this subsection, we consider the optimal allocation problem and compare the simulation cost of different algorithms with a smaller precision parameter ϵ. In Section 5, we choose ϵ to be $N/2$, which is at least 20% of the optimal objective function value. To show the performance of algorithms when the precision parameter is small, we consider the case when $\epsilon = N/10 + 1$. This choice of the precision parameter is approximately 10% of the optimal objective function value. With this smaller choice of ϵ and dimension $d \geq 8$, the simulation cost may be prohibitively large (longer than 24 hours) on a personal computer. Therefore, we focus on the case when $d = 4$ and $N = 10, \ldots, 50$. The results are summarized in Table EC.4. Compared with the results of large precision parameter in Table 2, we can see that the algorithms perform similarly and the dimension reduction algorithm also achieves the best performance. With a smaller precision parameter, the objective function value of the solution returned by different algorithms is closer to each other compared with the large precision parameter case. This indicates that our algorithms may have achieved a much better optimality gap than ϵ.

Table EC.4 Simulation cost and objective value on the allocation problem with smaller precision parameter.

Params.		**Search Methods** SubGD		**Localization Methods (this work)** Vaidya's		Random Walk		Dim Reduction	
d	N	Cost	Obj.	Cost	Obj.	Cost	Obj.	Cost	Obj.
4	10	5.61e6	2.13e1	7.41e5	2.18e1	9.88e5	2.13e1	**1.93e5**	2.25e1
4	20	3.53e6	3.42e1	4.67e5	3.41e1	5.27e5	3.41e1	**1.53e5**	3.42e1
4	30	2.43e6	4.52e1	3.48e5	4.59e1	3.81e5	4.53e1	**1.24e5**	4.51e1
4	40	1.80e6	5.62e1	2.17e5	5.68e1	2.76e5	5.63e1	**1.06e5**	5.65e1
4	50	1.66e6	6.67e1	1.53e5	6.74e1	1.68e5	6.81e1	**9.81e4**	6.66e1