Automatica 134 (2021) 109927

Contents lists available at ScienceDirect

Automatica

journal homepage: www.elsevier.com/locate/automatica

Optimal Computing Budget Allocation for regression with gradient information ☆

Tianxiang Wang [a], Jie Xu [b], Jian-Qiang Hu [a,*], Chun-Hung Chen [b]

[a] Department of Management Science, School of Management, Fudan University, Shanghai, 200433, China
[b] Department of Systems Engineering and Operations Research, George Mason University, Fairfax, VA 22030, United States of America

article info

Article history:
Received 30 March 2020
Received in revised form 13 April 2021
Accepted 10 August 2021
Available online xxxx

Keywords:
Simulation optimization
Stochastic systems
Gradient information
Computing budget allocation
Quadratic model

abstract

We consider the problem of optimizing the performance of a stochastic system, e.g., a discrete-event system, where the system performance is evaluated using stochastic simulations. Our objective is to allocate simulation budget to maximize the probability of correct selection (PCS) of the best design, where both system performance and gradient information can be obtained simultaneously via simulation. The objective function is assumed to be quadratic, or can be approximated by a quadratic regression model. The main contribution of our work is to utilize gradient information to enhance the efficiency of traditional Optimal Computing Budget Allocation (OCBA). We develop near-optimal rules that determine design points where simulations should be run and the number of runs allocated to each point. Our numerical experiments demonstrate that the proposed approach performs much better than other existing ranking and selection methods, even in cases where derivative information is very noisy and its simulation cost is high.

© 2021 Elsevier Ltd. All rights reserved.

1. Introduction

Simulation-based optimization, or simply simulation optimization, is a technique widely used to optimize the performance of stochastic systems, e.g., discrete event systems. Compared to analytical approaches, simulation optimization directly uses simulations to estimate system performance and search for the best design, making it a widely applicable approach for many applications in automation science and engineering (Brantley, Lee, Chen, & Xu, 2014; Peng, Xu, Lee, Hu, & Chen, 2018; Schwartz, Wang, & Rivera, 2006; Teng, Lee, & Chew, 2007; Zhang, Song, Dai, & Xu, 2020). Because simulations can be computationally expensive and simulation output noises can be significant, it is critical to optimally allocate simulation budget among multiple designs such that the best design can be selected with the highest probability possible given a limited computing budget. Formally, the problem we consider in this paper is to select the best design among a finite number of choices, where the performance of each design must be estimated with some uncertainty through stochastic simulation sampling, using a given finite simulation budget. This problem setting has been an active area of research in simulation and automation literature (Peng, Chong, Chen, & Fu, 2018; Xiao, Chen, & Lee, 2019; Xiao, Gao, & Lee, 2017), with a comprehensive review presented in Xu et al. (2016). Among many approaches developed for this problem, the Optimal Computing Budget Allocation (OCBA) approach (Chen, Gao, Chen, & Shi, 2013; Chen, He, Fu, & Lee, 2008; Chen, Lin, Yücesan, & Chick, 2000; Gao, Chen, & Shi, 2017; Gao, Xiao, Zhou, & Chen, 2017; Lee et al., 2010; Lee, Pujowidianto, Li, Chen, & Yap, 2012; Xiao, Gao, & Lee, 2019; Xiao, Lee, & Ng, 2013) is one of the most efficient methods, particularly in the case where the simulation budget is limited and/or the number of alternatives is large (Branke, Chick, & Schmidt, 2007; Pasupathy, Hunter, Pujowidianto, Lee, & Chen, 2014). According to Chen and Ryzhov (2019) and Ryzhov (2016), the expected improvement (EI), which is another class of efficient simulation budget allocation approaches, has the sampling ratios among non-best designs converge to OCBA's sampling ratios for all non-best designs. However, EI would let the sampling ratio for the best design approach 1 asymptotically. In contrast, OCBA's asymptotic allocation to the best design is bounded away from 1.

In the framework of OCBA, the problem is studied from the perspective of allocating a fixed number of simulation replications to the designs that are critical in the process of identifying the best design and thus to maximize the probability of correct selection (PCS). However, previous OCBA works all assume that only the performance information of designs is available.

☆ T. Wang and J.-Q. Hu were supported in part by the National Natural Science Foundation of China (NSFC) under Grants 71720107003, 72033003, and 71571048. J. Xu was supported in part by the National Science Foundation under Grant DMS-1923145. C.-H. Chen was supported in part by the National Science Foundation under Awards FAIN 2123683. The material in this paper was presented at the 2019 INFORMS international conference, June 8–13, 2019, Cancun, Mexico. This paper was recommended for publication in revised form by Associate Editor Rong Su under the direction of Editor Christos G. Cassandras.

* Corresponding author.
E-mail addresses: wangtx16@fudan.edu.cn (T. Wang), jxu13@gmu.edu (J. Xu), hujq@fudan.edu.cn (J.-Q. Hu), cchen9@gmu.edu (C.-H. Chen).

https://doi.org/10.1016/j.automatica.2021.109927
0005-1098/© 2021 Elsevier Ltd. All rights reserved.

In many simulation settings, not only can the performance of each design be estimated, but its sensitivity information can also be obtained, often at the fraction of the simulation budget needed for estimating the performance, for example, by using the technique of perturbation analysis (PA) (Fu & Hu, 1997; Glasserman, 1991; Ho & Cao, 1991) or the likelihood ratio (LR) method (Glynn, 1990). Motivated by this fact, we consider the case where the derivative (gradient) information is also available. In other words, we are interested in the following question: how to optimally allocate a fixed amount of simulation budget if the performance of each design and its corresponding sensitivity (derivative) can be estimated.

In this paper, we use a setting similar to that of Brantley, Lee, Chen, and Chen (2013), Brantley et al. (2014), where the underlying response surface is approximately quadratic or piecewise quadratic. Brantley et al. (2013) proposed an Optimal Simulation Design (OSD) approach that incorporates this quadratic response surface information in simulation budget allocation. Unlike traditional R&S methods, this regression based approach requires simulation of only a subset of the alternative design locations and so the simulation efficiency can be dramatically enhanced. For example, the average system time for a customer in a queueing system may be estimated by simulation output data. The optimization problem then is to determine a service rate within an interval of feasible values that balances customer system time and the cost of servers. If the cost of servers is a linear function of service rates, e.g., determined by how many servers to deploy, and also the queueing system is an M/M/c system, the objective function is then approximately quadratic. So OSD would only require the simulation of a small set of designs and dramatically increase computational efficiency over traditional R&S methods that may need to experiment with many designs. When performing queueing simulations, we may simultaneously use PA or LR to estimate the gradient of average system time with respect to service rate lying in a bounded interval. Our work shows that sensitivity information can further decrease the number of the design locations requiring simulation and thus increase computation efficiency.

Gradient information has been successfully used to improve the efficiency of parameter estimation in simulation metamodels, including linear regression (Fu & Qu, 2014) and stochastic kriging (Chen, Ankenman, & Nelson, 2013; Qu & Fu, 2014). Nevertheless, as to the best of our knowledge, there has not been any work that studies the use of gradient information to help improve the efficiency of R&S methods. In this paper, we make such an attempt. Specifically, to utilize the simulation budget in a most efficient way pursuing the maximization of PCS, we want to determine (i) which designs should be selected for simulation and (ii) the number of simulation runs for those selected designs.

This paper develops a novel Optimal Computing Budget Allocation with Gradient information (OCBAG) method to address these issues. Numerical testing demonstrates that our new method can enhance simulation efficiency, compared with existing efficient R&S methods such as OCBA and OSD. By making full use of gradient information, OCBAG method offers dramatic further improvements. For example, in comparison with OSD method, our novel OCBAG method can potentially save half of the total simulation budget while maintaining the same performance efficiency. The improvement made by OCBAG method is still quite significant even when gradient information is noisy and its simulation cost is high.

The rest of this article is organized as follows. In Section 2, we introduce the simulation optimization problem setting and Bayesian framework. Section 3 provides the development of OCBAG method. Numerical experiments comparing the results using the new OCBAG method and OSD method are provided in Section 4. Finally, Section 6 provides the conclusions and

2. Problem setting

2.1. Problem statement

We consider the problem with the principal goal of selecting the best design among k designs x_i, $i \in I = \{1, 2, \ldots, k\}$, where x_i is a scalar. For notational simplicity, we assume that x_1 is the smallest and x_k is the largest, and let $X = \{x \in \mathrm{R} | x_1 \le x \le x_k\}$. Without loss of generality, we consider the minimization problem shown below where the best design is the one with the smallest expected performance measure

$$\min_{i \in I} \; y(x_i) = E[f(x_i)]. \tag{1}$$

In this paper, we study the case where the performance measure is quadratic or approximately quadratic in X, i.e.,

$$y(x_i) = \beta_0 + \beta_1 x_i + \beta_2 x_i^2. \tag{2}$$

We denote the unknown model parameters by $\boldsymbol{\beta} = [\beta_0, \beta_1, \beta_2]$. We consider the case where $y(x_i)$ can only be estimated via simulation with homogeneous noise. In addition, the corresponding gradient information, denoted as $g(x_i)$, can also be estimated at the same time (possibly based on the same simulation) with homogeneous noise. We assume that the simulation noises of $f(x_i)$ and $g(x_i)$, denoted by ε_j and ε'_j, are independent normal random variables, and they are also independent of each other across replications. Therefore, for replication j at x_i, we have

$$\begin{aligned} f(x_i) &= y(x_i) + \varepsilon_j, && \text{where } \varepsilon_j \sim N(0, \sigma^2); \\ g(x_i) &= \beta_1 + 2\beta_2 x_i + \varepsilon'_j, && \text{where } \varepsilon'_j \sim N(0, \sigma_g^2). \end{aligned}$$

The parameters $\boldsymbol{\beta}$ are unknown, so is $y(x_i)$. However, we can find an estimate for $y(x_i)$, denoted as $\hat{y}(x_i)$, by making full use of both performance and gradient information based on a generalized least square estimate. Let $\hat{\boldsymbol{\beta}} = [\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2]$ be the least square estimates for the corresponding parameters in (2), then we have

$$\hat{y}(x_i) = \hat{\beta}_0 + \hat{\beta}_1 x_i + \hat{\beta}_2 x_i^2. \tag{3}$$

In order to obtain $\hat{\boldsymbol{\beta}}$, suppose we run simulations at a number of x_i's (no simulation at other design points), which we call the support points and denote by $\{x_i : i \in I_n\}$, where $I_n \subset I$ is the set of the indices of the support points and $|I_n| = n \le k$.

For ease of exposition, we use matrix notation for linear regression. For the support points $\{x_i : i \in I_n\}$, we define F as the $2n$-dimensional vector containing $f(x_i)$ and $g(x_i)$, X as the $2n \times 3$ matrix consisting of rows $[1, x_i, x_i^2]$ and $[0, 1, 2x_i]$, and V as the $2n \times 2n$ covariance matrix of the simulation noises of performance and gradient information:

$$F = \begin{pmatrix} (f(x_i))_{i \in I_n} \\ (g(x_i))_{i \in I_n} \end{pmatrix}, \quad X = \begin{pmatrix} (1, x_i, x_i^2)_{i \in I_n} \\ (0, 1, 2x_i)_{i \in I_n} \end{pmatrix},$$

$$V = \begin{bmatrix} \sigma^2 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma^2 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \sigma_g^2 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & \sigma_g^2 \end{bmatrix}.$$

Following standard least square regression analysis, we obtain the generalized least squares estimate for $\boldsymbol{\beta}$ as

$$\hat{\boldsymbol{\beta}} = (X^T V^{-1} X)^{-1} X^T V^{-1} F. \tag{4}$$

suggestions for future work using the concepts introduced in this article.

Since the Gauss–Markov conditions are satisfied, the above estimates for β are unbiased and have the minimum variance among

all unbiased linear estimators (see Draper & Smith, 1998; Fu & Qu, 2014).

We aim to select the design associated with the smallest mean performance measure from the k designs using a total computing budget of T simulation samples (replications). Given the generalized least squares estimates for the parameters, we can use (3) to estimate the expected performance measure at each design point. We designate the design point with the smallest estimated mean performance measure as x_b, i.e., $\hat{y}(x_b) = \min_{i \in I} \hat{y}(x_i)$. Given the uncertainty of the estimated parameters, x_b is a random variable that is dependent upon the size of the computing budget and the number of simulation replications allocated to each design point. We define correct selection as the event where x_b is indeed the best design and N_i as the number of simulation replications conducted at x_i (x_i is a support point if $N_i > 0$). Since simulation is expensive and the computing budget is limited, we seek to develop an allocation policy in order to maximize the probability of correctly selecting the best design (PCS). This Optimal Computing Budget Allocation with Gradient information (OCBAG) problem is described as follows:

$$\begin{array}{ll} \max & PCS = P\{y(x_b) \le y(x_i),\ \forall i = 1, 2, \ldots, k\} \\ \text{s.t.} & N_1 + N_2 + \cdots + N_k = T. \end{array} \tag{5}$$

In the constraint in the above problem we have implicitly assumed that the cost of each simulation run at every design point is one unit.

2.2. A Bayesian regression framework

To solve problem (5), we adopt a Bayesian regression framework as described in Gelman et al. (2013). Using a standard non-informative prior distribution, one would obtain the same estimates of $\boldsymbol{\beta}$ as in classical regression analysis briefly explained previously. The key difference is with a Bayesian perspective, it allows us to calculate PCS using the posterior distribution of $\boldsymbol{\beta}$. In the following, we use $\check{\boldsymbol{\beta}}$ and $\check{y}(x_i)$ to denote the random variables whose probability distributions are the posterior distribution of $\boldsymbol{\beta}$ and $y(x_i)$ conditional on F, respectively. PCS in (5) under the Bayesian regression framework is defined as Chen and Lee (2011), Chen, Yücesan, Dai, and Chen (2009):

$$PCS = P\left\{\check{y}(x_b) \le \check{y}(x_i),\ \forall i = 1, 2, \ldots, k\right\}. \tag{6}$$

Using a non-informative prior distribution and assuming that the conditional distribution of the simulation output vector F is a multi-variate normal distribution with mean $X\boldsymbol{\beta}$ and a covariance matrix V, DeGroot (2005) shows that the posterior distribution of $\boldsymbol{\beta}$ is then given by

$$\check{\boldsymbol{\beta}} | F \sim N\left(\boldsymbol{\beta}, \left(X^T V^{-1} X\right)^{-1}\right). \tag{7}$$

Since $\check{y}(x_i)$ is a linear combination of $\check{\boldsymbol{\beta}}$ elements, the posterior distribution of $\check{y}(x_i)$ is normal:

$$\check{y}(x_i) \sim N\left(X_i \hat{\boldsymbol{\beta}}, X_i^T (X^T V^{-1} X)^{-1} X_i\right), \tag{8}$$

where $X_i^T = [1, x_i, x_i^2]$.

Similar to the approach used in Brantley et al. (2013), we are also interested in how the PCS given by (6) changes if we conduct additional simulation runs so that we can make allocations so as to maximize PCS. See Brantley et al. (2013) for a detailed discussion of the predictive posterior distributions.

2.3. Simplification

In order to simplify problem (5), we aim to reduce the number of design locations at which we need to run simulation without resulting in any information loss. De la Garza (1954) established that for a polynomial of degree m and a discrete domain with more than $m + 1$ support points, the information in $X^T V^{-1} X$ (without any gradient information) obtained from more than $m + 1$ support points will always be attained by using only $m + 1$ of the support points. That is to say, we need just three different support points because of our quadratic model if we only use the performance information. In our problem setting, we obtain performance and gradient information simultaneously after running simulation for each design location. Because gradient information is equivalent to performance information in the sense of determining the unknown parameters $\boldsymbol{\beta}$, two support points provide four observations in the design matrix X ($X_i = [1, x_i, x_i^2]$ and $X_i = [0, 1, 2x_i]$), which is enough to help solve our problem. From now on, we refer to the support points as $\{x_{s_1}, x_{s_2}\}$ where $x_{s_1} < x_{s_2}$.

Furthermore, given the estimated x_b, the assumption that our underlying function is quadratic allows us to reduce the number of required comparisons in our PCS calculation in (6) from $k - 1$ to 2, i.e., we have (see Brantley et al., 2013)

$$\begin{aligned} PCS &= P\left\{\check{y}(x_b) \le \check{y}(x_i),\ \forall i = 1, 2, \ldots, k\right\} \\ &= P\left\{\check{y}(x_A) \ge \check{y}(x_b) \ \cap\ \check{y}(x_Z) \ge \check{y}(x_b)\right\}, \end{aligned} \tag{9}$$

where the identities of x_A and x_Z depend on whether x_b is an interior point or a boundary point, as described in the following three cases:

- Case 1 (Interior Case): $b = 1, k$; $A = b - 1$; $Z = b + 1$,
- Case 2 (Left Boundary Case): $b = 1$; $A = 2$; $Z = k$,
- Case 3 (Right Boundary Case): $b = k$; $A = 1$; $Z = k - 1$.

Let N_{s_1} and N_{s_2} be the number of simulation runs allocated at x_{s_1} and x_{s_2}, respectively, and $\alpha_{s_1} = \frac{N_{s_1}}{T}$ and $\alpha_{s_2} = \frac{N_{s_2}}{T}$. Then (5) can be restated as:

$$\begin{array}{ll} \max & PCS = P\left\{\check{y}(x_A) \ge \check{y}(x_b) \ \cap\ \check{y}(x_Z) \ge \check{y}(x_b)\right\} \\ \text{s.t.} & \alpha_{s_1} + \alpha_{s_2} = 1. \end{array} \tag{10}$$

We can estimate covariance matrix V from the samples generated at the support points and calculate $\check{y}(x_i)$ using (8). In the next section, we present an effective way to approximate PCS and derive approximate solutions to the optimal allocations.

3. Approximate solutions

In the previous section, we show that regardless how the simulation budget is allocated to the support points, PCS can be calculated based on the comparisons of x_b to its two adjacent design points, hence, we just need two support points. Now we are ready to derive our OCBAG allocation policy. In Section 3.1, we derive a lower bound of PCS as an approximation for PCS. We then develop an efficient approximation for the optimal allocation of the total simulation budget in Section 3.2. In Section 3.3, we determine the optimal locations of the two support points. Finally, in Section 3.4, we discuss the algorithmic implementation of the results from Sections 3.1–3.3.

3.1. A lower bound for PCS

First, we have

$$\begin{aligned} PCS &= P\{\check{y}(x_A) \ge \check{y}(x_b) \cap \check{y}(x_Z) \ge \check{y}(x_b)\} \\ &= 1 - P\{\check{y}(x_A) \le \check{y}(x_b) \cup \check{y}(x_Z) \le \check{y}(x_b)\} \\ &= 1 - P\left\{\check{y}(x_A) \le \check{y}(x_b)\right\} - P\left\{\check{y}(x_Z) \le \check{y}(x_b)\right\} \\ &\quad + P\{\check{y}(x_A) \le \check{y}(x_b) \cap \check{y}(x_Z) \le \check{y}(x_b)\}. \end{aligned}$$

Since the underlying function is quadratic, the last term in the above equation is simply the probability that x_b is the "worst" design with the largest expected performance measure. This probability is typically very small so that we can establish an efficient

lower bound, APCS defined as follows, which can be used as an approximation for PCS:

$$PCS \geq APCS = 1 - P\{\tilde{y}(x_A) \leq \tilde{y}(x_b)\} - P\{\tilde{y}(x_Z) \leq \tilde{y}(x_b)\}.$$

Let

$$\tilde{d}(x_i) \equiv \tilde{y}(x_i) - \tilde{y}(x_b) = \tilde{\beta}_2(x_i^2 - x_b^2) + \tilde{\beta}_1(x_i - x_b).$$

Because $\tilde{d}(x_i)$ is a linear combination of the elements of $\tilde{\boldsymbol{\beta}}$, $\tilde{d}(x_i)$ is also normally distributed, i.e., $\tilde{d}(x_i) \sim N(\hat{d}(x_i), \zeta_i)$, where $\hat{d}(x_i) = \hat{y}(x_i) - \hat{y}(x_b)$ and

$$\zeta_i = (0, x_i - x_b, x_i^2 - x_b^2)(\mathbf{X}^T\mathbf{V}^{-1}\mathbf{X})^{-1}(0, x_i - x_b, x_i^2 - x_b^2)^T. \quad (11)$$

Therefor, we have

$$P\{\tilde{y}(x_i) \leq \tilde{y}(x_b)\} = P\{-\tilde{d}(x_i) \geq 0\} = \int_0^\infty \frac{1}{\sqrt{2\pi\zeta_i}} e^{-[v-(-\hat{d}(x_i))]^2/2\zeta_i} dv = \int_{\hat{d}(x_i)/\sqrt{\zeta_i}}^\infty \frac{1}{\sqrt{2\pi}} e^{-r^2/2} dr, \quad (12)$$

where $r = (v + \hat{d}(x_i))/\sqrt{\zeta_i}$. APCS can be calculated based on (12) with i being replaced by A and Z.

Hereafter, we will use APCS to approximate PCS. In other words, we replace PCS with APCS in (10) and consider instead the following optimization problem:

$$\begin{aligned} \max \quad & APCS \\ \text{s.t.} \quad & \alpha_{s_1} + \alpha_{s_2} = 1. \end{aligned} \quad (13)$$

3.2. The optimal allocation policy

In this subsection, we solve (13). As shown in Appendix B, ζ_i can be simplified as:

$$\zeta_i = \frac{\sigma^2}{T}\left[\frac{D_{i,s_1}^2}{\alpha_{s_1}} + \frac{D_{i,s_2}^2}{\alpha_{s_2}} + \frac{(x_i - x_b)^2(x_{s_1} + x_{s_2} - x_i - x_b)}{K[(x_{s_1} - x_{s_2})^2 + 4K]}\right], \quad (14)$$

where

$$D_{i,s_1}^2 = \frac{(2x_{s_2} - x_i - x_b)^2(x_i - x_b)^2}{K(x_{s_1} - x_{s_2})^2[(x_{s_1} - x_{s_2})^2 + 4K]},$$

$$D_{i,s_2}^2 = \frac{(2x_{s_1} - x_i - x_b)^2(x_i - x_b)^2}{K(x_{s_1} - x_{s_2})^2[(x_{s_1} - x_{s_2})^2 + 4K]},$$

$$K = \sigma^2/\sigma_g^2.$$

(14) shows that ζ_i and thus APCS are dependent on T and the allocations of the two support points. The following theorem provides an optimal allocation policy for (13) for any given two support points.

Theorem 1. *For any given two support points $\{x_{s_1}, x_{s_2}\}$, the optimal solution to (13) is given by*

$$\alpha_{s_1} = \frac{\sqrt{q_{s_1}}}{\sqrt{q_{s_1}} + \sqrt{q_{s_2}}} \quad \text{and} \quad \alpha_{s_2} = \frac{\sqrt{q_{s_2}}}{\sqrt{q_{s_1}} + \sqrt{q_{s_2}}}, \quad (15)$$

where

$$q_j = \phi\left(\frac{\hat{d}(x_A)}{\sqrt{\zeta_A}}\right)\frac{\hat{d}(x_A)D_{A,j}^2}{(\zeta_A)^{3/2}} + \phi\left(\frac{\hat{d}(x_Z)}{\sqrt{\zeta_Z}}\right)\frac{\hat{d}(x_Z)D_{Z,j}^2}{(\zeta_Z)^{3/2}}, \quad j = s_1, s_2,$$

and $\phi(r) = \frac{1}{\sqrt{2\pi}} e^{-r^2/2}$.

Instead of using (15), in what follows we propose an approximate but much simpler way to compute α_{s_1} and α_{s_2}. We note

$$\begin{aligned} APCS &= 1 - P\{\tilde{y}(x_A) \leq \tilde{y}(x_b)\} - P\{\tilde{y}(x_Z) \leq \tilde{y}(x_b)\} \\ &\geq 1 - 2\max\{P\{\tilde{y}(x_A) \leq \tilde{y}(x_b)\}, P\{\tilde{y}(x_Z) \leq \tilde{y}(x_b)\}\} \\ &\geq 1 - 2P\{\tilde{y}(x_M) \leq \tilde{y}(x_b)\}, \end{aligned}$$

where

$$M = \underset{i=A,Z}{\operatorname{argmax}}\{\phi(\hat{d}(x_i)/\sqrt{\zeta_i})\} = \underset{i=A,Z}{\operatorname{argmin}}\{\hat{d}(x_i)/\sqrt{\zeta_i}\}. \quad (16)$$

In fact, x_A and x_Z are usually very close to each other since they are both adjacent to x_b. Therefore, the gap between APCS and $1 - 2P\{\tilde{y}(x_M) \leq \tilde{y}(x_b)\}$ is relatively small, especially when T is large. We propose to use $1 - 2P\{\tilde{y}(x_M) \leq \tilde{y}(x_b)\}$ to replace APCS. With slight abuse of notation, hereafter, we set APCS $= 1 - 2P\{\tilde{y}(x_M) \leq \tilde{y}(x_b)\}$.

With modified APCS, we can re-solve (13) to obtain

$$\alpha_{s_1} = \frac{2\phi\left(\frac{\hat{d}(x_M)}{\sqrt{\zeta_M}}\right)\frac{\hat{d}(x_M)D_{M,s_1}^2}{(\zeta_M)^{3/2}}}{2\phi\left(\frac{\hat{d}(x_M)}{\sqrt{\zeta_M}}\right)\frac{\hat{d}(x_M)D_{M,s_1}^2}{(\zeta_M)^{3/2}} + 2\phi\left(\frac{\hat{d}(x_M)}{\sqrt{\zeta_M}}\right)\frac{\hat{d}(x_M)D_{M,s_2}^2}{(\zeta_M)^{3/2}}} = \frac{|D_{M,s_1}|}{|D_{M,s_1}| + |D_{M,s_2}|} = \frac{|2x_{s_2} - x_M - x_b|}{|2x_{s_1} - x_M - x_b| + |2x_{s_2} - x_M - x_b|}, \quad (17)$$

$$\alpha_{s_2} = \frac{|2x_{s_1} - x_M - x_b|}{|2x_{s_1} - x_M - x_b| + |2x_{s_2} - x_M - x_b|}. \quad (18)$$

The new allocation policy is much easier to compute. Our numerical results in Section 4 demonstrate that it works quite well.

3.3. Optimal location of support points

Given the simulation budget allocation to the two support points $\{x_{s_1}, x_{s_2}\}$ derived in the previous subsection, we now try to determine the optimal location for $\{x_{s_1}, x_{s_2}\}$ among all possible design points. In our derivation, we assume x_{s_1} and x_{s_2} are continuous variables in $[x_1, x_k]$. Once we obtain the values of x_{s_1} and x_{s_2}, if they do not equal to any of given design points, we can simply use the design points closest to them.

According to (12), in order to maximize APCS, it is sufficient to maximize $\frac{\hat{d}(x_M)}{\sqrt{\zeta_M}}$. Given the unbiased properties of our generalized least squares parameter estimators, there would not be large changes in $\hat{d}(x_M)$ as we take additional simulation runs. Therefore, we focus on selecting the best two support points to minimize the variance of $\hat{d}(x_M)$, (ζ_M). In Appendix B, we show that with the simulation budget being allocated to x_{s_1} and x_{s_2} according to (17) and (18), ζ_M is minimized with the following solutions for x_{s_1} and x_{s_2}:

$$\{x_{s_1}, x_{s_2}\} = \begin{cases} \{x_1, x_2\}, \\ \left\{x_1, \frac{x_M + x_b}{2} + \sqrt{\left(\frac{x_M + x_b}{2} - x_i\right)^2 + 4K}\right\}, \\ \quad \text{if } \frac{x_M + x_b}{2} - \frac{x_1 + x_k}{2} \leq -\frac{2K}{x_k - x_1}; \\ \{x_1, x_k\}, \\ \left\{\text{if } -\frac{2K}{x_k - x_1} < \frac{x_M + x_b}{2} - \frac{x_1 + x_k}{2} \leq \frac{2K}{x_k - x_1}; \right. \\ \left. \frac{x_M + x_b}{2} - \sqrt{\left(x_k - \frac{x_M + x_b}{2}\right)^2 + 4K}, x_k\right\}, \\ \left\{\text{if } \frac{x_M + x_b}{2} - \frac{x_1 + x_k}{2} > \frac{2K}{x_k - x_1}.\right. \end{cases} \quad (19)$$

$\sqrt{2\pi}$ $\overline{2}$ $\overline{2}$ $\overline{x_k - x_1}$

(19) implies that when the point $(x_M + x_b)/2$, which can be approximately regarded as the optimal solution to the quadratic objective function over $[x_1, x_k]$, is in the middle part of the interval, we select the two extreme points x_1 and x_k as support points. When the point $(x_M + x_b)/2$ is very close to an extreme point, we select the nearest extreme point as one of the two support points and the other one located such that it is a little farther than the first one from $(x_M + x_b)/2$.

To shed more light on our results, we compare them with the results in Brantley et al. (2013) where gradient information is not considered. OSD method requires three support points instead of two in OCBAG because OSD does not use gradient information. The allocation policy of OSD is shown as follows:

$$\alpha_i = \frac{|D_{M,i}|}{|D_{M,1}| + |D_{M,s}| + |D_{M,k}|}, \quad i = 1, s, k, \tag{20}$$

where

$$D_{M,1} = \frac{\{(x_s - x_M)(x_k - x_M) - (x_s - x_b)(x_k - x_b)\}}{(x_1 - x_s)(x_1 - x_k)},$$

$$D_{M,s} = \frac{\{(x_1 - x_M)(x_k - x_M) - (x_1 - x_b)(x_k - x_b)\}}{(x_s - x_1)(x_s - x_k)},$$

$$D_{M,k} = \frac{\{(x_1 - x_M)(x_s - x_M) - (x_1 - x_b)(x_s - x_b)\}}{(x_k - x_1)(x_k - x_s)}.$$

In OSD method, two extreme points $\{x_1, x_k\}$ are always chosen as support points. When $(x_M + x_b)/2$ is in the middle half of the interval, the third support point (x_s) is located such that it has the same distance from $(x_M + x_b)/2$ as the nearest extreme point is from $(x_M + x_b)/2$. This makes the design symmetric around $(x_M + x_b)/2$. When $(x_M + x_b)/2$ is in the outer half of the interval, the third support point (x_s) is located at the center of the interval. Specifically, the location of x_s is given as follows:

$$x_s = \begin{cases} x_{M+b-1}, & \text{if } \frac{3x_1+x_k}{4} \le \frac{x_M+x_b}{2} < \frac{x_1+x_k}{2}; \\ x_{M+b-k}, & \text{if } \frac{x_1+x_k}{2} < \frac{x_M+x_b}{2} \le \frac{x_1+3x_k}{4}; \\ x_{(k-1)/2}, & \text{otherwise.} \end{cases} \tag{21}$$

Since we use additional gradient information while OSD method does not, there are two critical factors which may affect the performance of these two methods. The first is the ratio of performance and gradient noise variances $K = \sigma^2/\sigma_g^2$, and the second is the computational cost of obtaining gradient information. We denote κ as the relative cost of obtaining gradient information with respect to the cost of obtaining performance information. We should point out that the value of κ does not affect the allocation policy although in our derivation process we have implicitly assumed that $\kappa = 0$. For general κ, we can simply replace the total simulation budget T with $\frac{T}{1+\kappa}$ and all our results still hold.

If $K = \sigma^2/\sigma_g^2$ is very small, which means that the variance of the gradient noise is much bigger than that of the performance noise, our results are the same as those in Brantley et al. (2013) because gradient information is not helpful due to its large noise. In this case, the first support point is located at the nearest extreme point and the second one is such that it is the same distance from $(x_M + x_b)/2$ as the nearest extreme point is from $(x_M + x_b)/2$. On the other hand, if K is large, then our results show that we should select the two extreme points as the support points. In fact, if we can only obtain gradient information from simulation, the underlying function becomes a linear function. It is obvious then the two support points should be as far as possible to minimize the variance of the estimated parameters $(\hat{\beta}_1, \hat{\beta}_2)$. For a moderate K, i.e. the variances of the performance and gradient estimates are comparable, our method can optimally utilize performance and gradient information.

3.4. OCBAG algorithm

The following is the algorithm in which we implement the results in the previous subsections and will be used for the numerical experiments in Section 4.

Algorithm 1 (OCBAG)

INPUT: k (the number of design points), T (the simulation budget), x_i ($i = 1, \ldots, k$, the design points), n_0 (the number of initial simulation replications for each support point), θ_j (the number of simulation replications allocated to each iteration j);
INITIALIZE: $j \leftarrow 0$;
Take the two extreme points as the initial support points. Perform n_0 simulation replications for each support point (set $N_1^0 = N_k^0 = n_0$) to get the performance and gradient information.

LOOP: WHILE $\sum_{i=1}^{k} N_i < T$ **DO**
UPDATE:
Estimate the mean, variance, and gradient at each x_i
1: Estimate a quadratic regression model using the performance and gradient information from all prior simulation runs;
2: Estimate the mean and variance each x_i using: $\hat{y}(x_i) = \hat{\beta}_0 + \hat{\beta}_1 x_i + \hat{\beta}_2 x_i^2$.
3: Determine the observed best design $x_b = \text{argmin}_i\, \hat{y}(x_i)$.
4: Based upon the location of x_b, determine x_A, x_Z, and x_M based on the three cases and (16).
5: Determine support points x_{s_1} and x_{s_2} based on (19).
ALLOCATE: Increase the computing budget by θ_{j+1} and calculate its allocation $\alpha_{s_1}^{j+1}$ and $\alpha_{s_2}^{j+1}$ according to (17) and (18).
SIMULATE: Perform $\theta_{j+1}\alpha_{s_1}^{j+1}$ and $\theta_{j+1}\alpha_{s_2}^{j+1}$ simulations (round to integers as needed) for support points x_{s_1} and x_{s_2}. Update N_{s_1} and N_{s_2}.
END OF LOOP

4. Numerical experiments

In this section, we compare our OCBAG method with the Optimal Simulation Design (OSD) method of Brantley et al. (2013). When the underlying function is quadratic or approximately quadratic, Brantley et al. (2013) show that OSD method performs much better than other traditional allocation procedures, including Equal-Allocation (EA), Equal-Allocation and Response Surface combination method (EA-RS), traditional OCBA method (Branke et al., 2007; Chen et al., 2000), D-optimality design (Atkinson, Donev, & Tobias, 2007; Liski, Mandal, Shah, & Sinha, 2002).

We conduct experiments with $K = 0.5$, 1, and 2. Regarding the computational cost of obtaining gradient information,

we consider two scenarios: $\kappa = 0, 1$. These two scenarios are commonly seen in simulation. For example, when the technique of infinitesimal perturbation analysis (IPA) can be used to obtain gradient information, we usually have very small κ, and in the case when the finite difference is used for gradient information we have $\kappa = 1$.

We use the same four test problems in Brantley et al. (2013) to compare OCBAG and OSD. In the first experiment, we include the results from the traditional OCBA method as well for comparison. Since as we should see that both OCBAG and OSD perform much better than OCBA, we focus on the comparison between OCBAG and OSD for the other three experiments. We conduct the first experiment with T up to 1000, the second experiment with T up to 45,000, and the last two experiments with T up to 10,000. We repeat 10,000 times for each instance and then calculate the PCS obtained for each method out of these 10,000 independent macro replications.

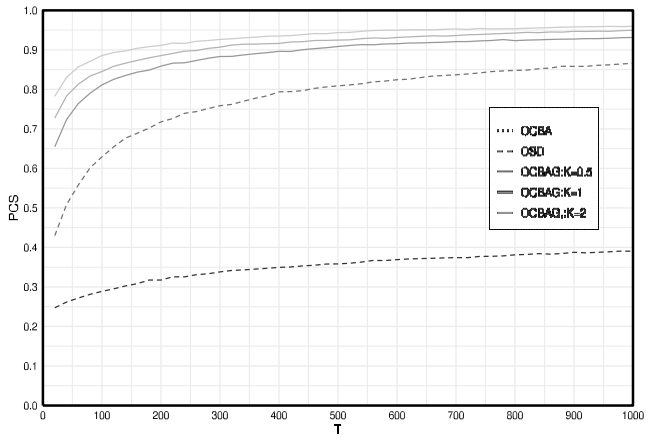


Fig. 1. Experiment 1: $\kappa = 0$ and $K = 0.5, 1, 2$.

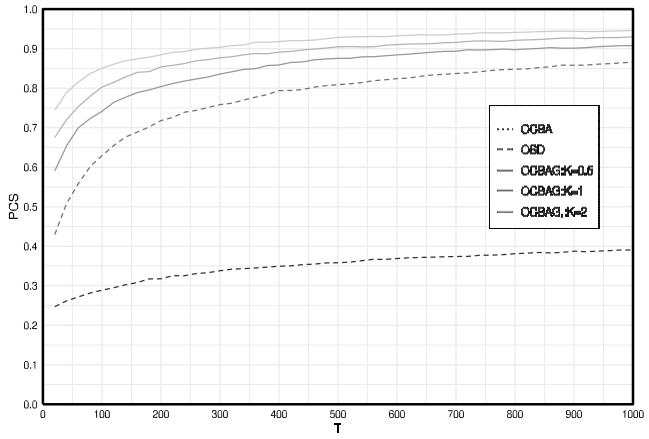


Fig. 2. Experiment 1: $\kappa = 1$ and $K = 0.5, 1, 2$.

4.1. Experiment 1: quadratic function with randomly generated optimal solution, 11 design points

The test function and its gradient are given below:

$$f(x_i) = (x_i - a)^2 + N(0, 1),$$
$$g(x_i) = 2x_i - 2a + N(0, \frac{1}{K}),$$

We use a design space consisting of 11 evenly spaced design points from -1 to 1, i.e. $x_i \in \{-1, -0.8, \ldots 1\}$. In order to test the methods against a diverse set of problems, the stationary point for the underlying quadratic equation was randomly generated from a uniformly distributed random variable a over interval $(-1, 1)$. The optimal solution is then the design point closest to the generated a. For OSD method, we set $N_1^0 = N_{(k+1)/2}^0 = N_k^0 = 2$ and allocate 14 additional runs for each iteration, and for OCBAG method, we set n_0 to 2 and allocate 14 runs for each iteration. For this experiment, we include the results from the traditional OCBA as well.

The results are illustrated in Figs. 1 and 2, in which we plot PCS against the total number of simulation runs (T). In Fig. 1, we compare our OCBAG method with OSD under three different gradient variances (larger than, equal to, smaller than performance variance) with $\kappa = 0$. First, we observe that by using additional gradient information, OCBAG method clearly produces better results than OSD method. Furthermore, even when the gradient noise variance is twice the performance noise variance, OCBAG still performs much better than OSD. Second, we observe that the smaller the gradient noise variance is, the higher the PCS would be. As shown in Fig. 2, even when $\kappa = 1$, OCBAG method still performs better than OSD method though the improvement is not as large as for $\kappa = 0$. Finally, we should point out both OCBAG and OSD perform much better than OCBA as shown in the two figures.

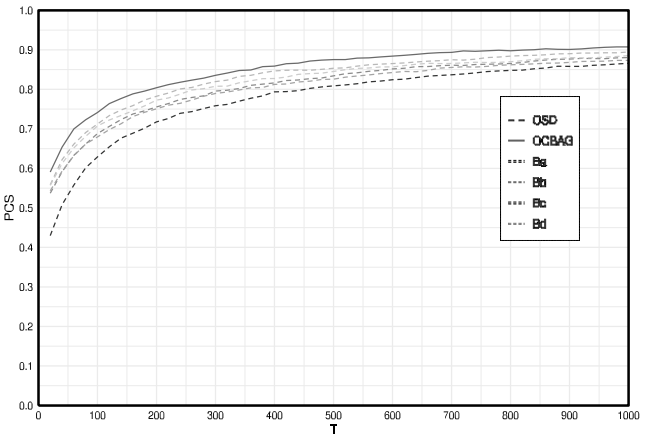


Fig. 3. Experiment 1: $\kappa = 1$ and $K = 0.5$.

To show that OCBAG not only uses gradient information, but does so in an efficient way, we compare OCBAG with the following four benchmark allocation schemes that also use gradient information:

- (Ba) There are three support points, with two being the extremes and one being the middle point. Simulation budget is allocated equally to these three points.
- (Bb) There are three support points, with two being the extremes and one being the middle point. Simulation budget allocation is proportional to simulation variances.
- (Bc) There are two support points, and the locations of the two points are determined by Eq. (19) in this paper. Simulation budget is allocated equally.
- (Bd) There are two support points, and the locations of the two points are determined by Eq. (19) in this paper. Simulation budget allocation is proportional to simulation variances.

We consider the case with $\kappa = 1$ and $K = 0.5$. Results are shown in Fig. 3. We first observe that all four benchmark allocation schemes outperform OSD. This clearly demonstrates the benefit of using gradient information in R&S. Furthermore, OCBAG clearly outperforms that these four benchmark allocation schemes, demonstrating its efficiency.

4.2. Experiment 2: quadratic function with randomly generated optimal solution, 101 design points

This experiment has the same setup as in Experiment 1 except that we now have 101 evenly spaced design points $\{-1, -0.98, \ldots, 1\}$. For OSD, we set $N_1^0 = N_{(k+1)/2}^0 = N_k^0 = 20$ and allocate 99 runs at each iteration as described in Brantley et al. (2013). For OCBAG, we set $n_0 = 2$ and allocate 99 runs at each iteration.

The results are shown in Figs. 4 and 5. With more designs points within the same interval, the performance differences between the best design and its nearest competitors are smaller relative to the simulation noise. Therefore, PCS for each method is smaller than in Experiment 1. However, it can be seen from both Figs. 4 and 5 that OCBAG still significantly outperforms OSD. Specifically, as Fig. 5 shows, after exhausting the total computing

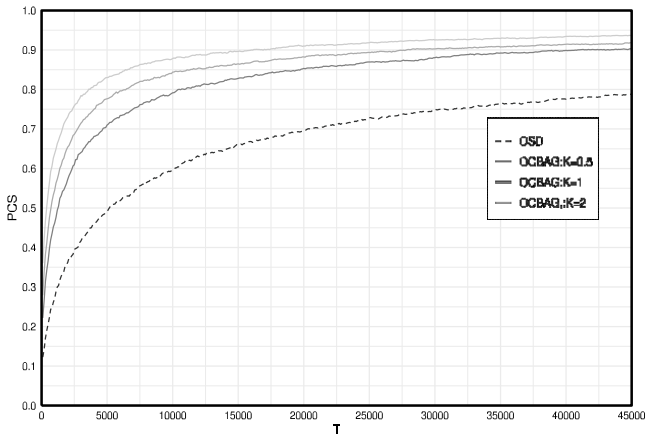


Fig. 4. Experiment 2: $\kappa = 0$ and $K = 0.5, 1, 2$.

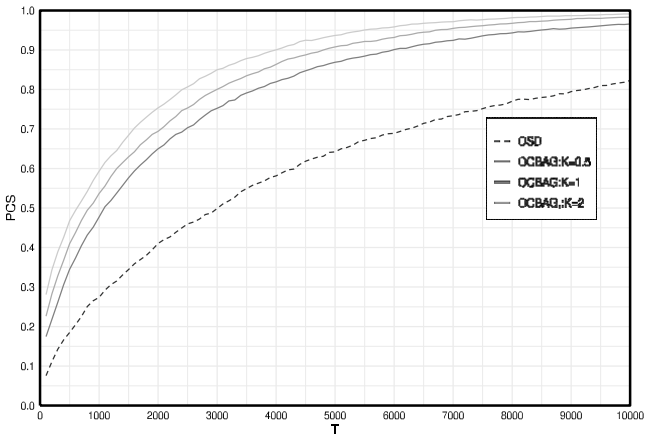


Fig. 6. Experiment 3: $\kappa = 0$ and $K = 0.5, 1, 2$.

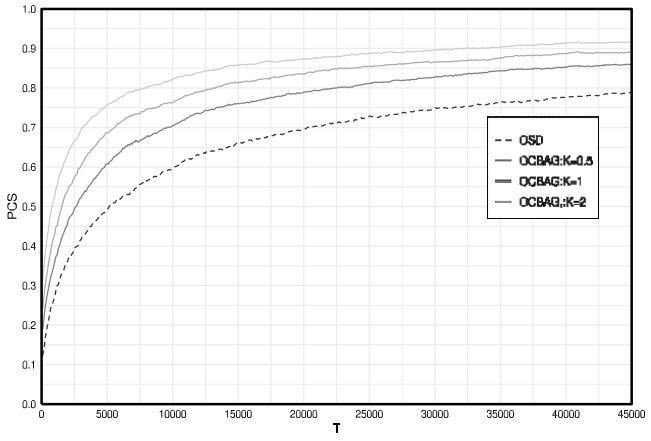


Fig. 5. Experiment 2: $\kappa = 1$ and $K = 0.5, 1, 2$.

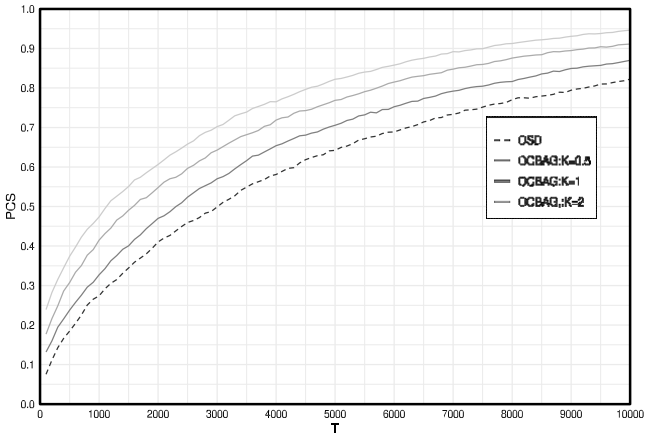


Fig. 7. Experiment 3: $\kappa = 1$ and $K = 0.5, 1, 2$.

budget, OSD just achieves a 78.7% PCS. In comparison, even when gradient information has large variance ($K = 0.5$) and the cost is high ($\kappa = 1$), OCBAG achieves the same PCS after only about 19,700 runs or about 43.78% of those required by OSD. This suggests the benefit of OCBAG increases as the number of design points increases.

4.3. Experiment 3: three local minima, 60 design points

This test function is not quadratic and is a test function from a classical optimization literature (Törn & Žilinskas, 1989):

$$f(x_i) = \sin(x_i) + \sin(10x_i/3) + \ln(x_i) - 0.84x_i + 3 + N(0, 10),$$

$$g(x_i) = \cos(x_i) + \frac{10}{3}\cos(10x_i/3) + \frac{1}{x_i} - 0.84 + N(0, \frac{10}{K}).$$

The performance noise variance is set to 10. There are 60 design points are evenly spaced between 3 and 8, with the global minimum $x_{27} \approx 5.20$ with $y(x_{27}) \approx -1.60$. This function also has two local minima at $x_6 \approx 3.42$ with $y(x_6) \approx 0.16$ and $x_{47} \approx 7.07$ with $y(x_{47}) \approx -1.27$. Because this function is not quadratic, we cannot directly apply OSD or OCBAG. Following Brantley et al. (2013), we partition the design space into six smaller intervals and each of the local minimums was in a separate interval. We then equally allocate the total computing budget to these smaller intervals. For OSD, we set $N_1 = N_{k/2} = N_k = 20$ and then allocate 99 runs for each iteration as described in Brantley et al. (2013). For OCBAG, we set $n_0 = 20$ and then allocate 99 runs for each iteration.

The results shown in Figs. 6 and 7 are consistent with the first two experiments where the underlying functions are quadratic. OCBAG still outperforms OSD. To illustrate the efficiency improvement achieved by OCBAG, as Fig. 7 shows, after exhausting the total computing budget, OSD just achieves a 82.1% PCS while OCBAG achieves the same PCS after only about 8000 runs or about 80% of those required by OSD even though the gradient noise variance is twice the performance noise variance and the costs of performance and gradient information are equal.

4.4. Experiment 4: one global minimum, asymmetric function, 60 design points

This experiment also uses a non-quadratic test function:

$$f(x_i) = 10x_i + \frac{10}{x_i} + N(0, 1),$$

$$g(x_i) = 10 - \frac{10}{x_i^2} + N(0, \frac{1}{K}).$$

We again use a design space consisting of 60 evenly spaced design points in $[0.5, 2.5]$. The global minimum is $x_{16} \approx 1.01$ with $y(x_{16}) \approx 20$. As with the third experiment, we partition the design space into six partitions. For OSD, we set $N_1 = N_{k/2} = N_k = 20$ and allocate 99 runs for each iteration as described in Brantley

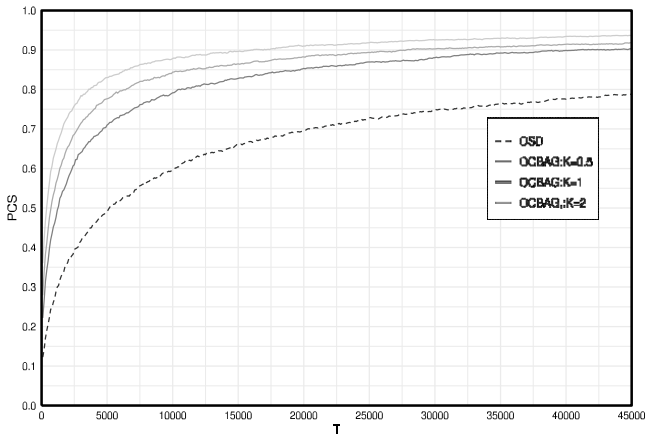


Fig. 8. Experiment 4: $\kappa = 0$ and $K = 0.5, 1, 2$.

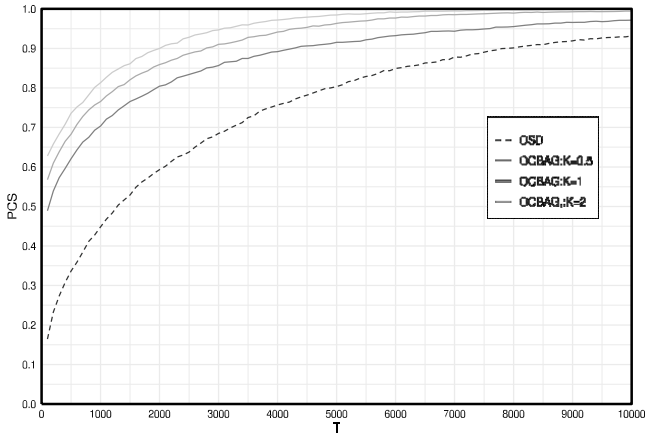


Fig. 9. Experiment 4: $\kappa = 1$ and $K = 0.5, 1, 2$.

et al. (2013). For OCBAG, we set $n_0 = 20$ and allocate 99 runs for each iteration.

The results, shown in Figs. 8 and 9, are again consistent with the previous experiments. OCBAG achieves a PCS close 1. From Fig. 9, OSD achieves a 93.1% PCS after about 10,000 runs while OCBAG achieves the same PCS after only 5900 runs or about 59% of those required by OSD even though the gradient noise variance is twice the performance noise variance and the costs of performance and gradient information are equal.

5. Queueing simulation example

Queueing systems are one of the main application areas for stochastic simulation, and the earliest application of direct gradient estimation in simulation was queueing. Therefore, we chose a simple queueing model to investigate the performance of our OCBAG method in a setting where direct gradient estimates are available but where one or more of the assumptions of the theoretical results are generally not satisfied. For example, the objective function is not strictly quadratic, the system time performance and its gradient estimate are clearly highly correlated, and the variance of both the performance and its gradient are not homogeneous across the range of design space, etc. Specifically, we consider the first-come, first-served, single-server queue. The customers arrive according to a Poisson process with constant rate λ and the server has i.i.d. exponential random service times with rate x. Similar to many simulation optimization settings, we

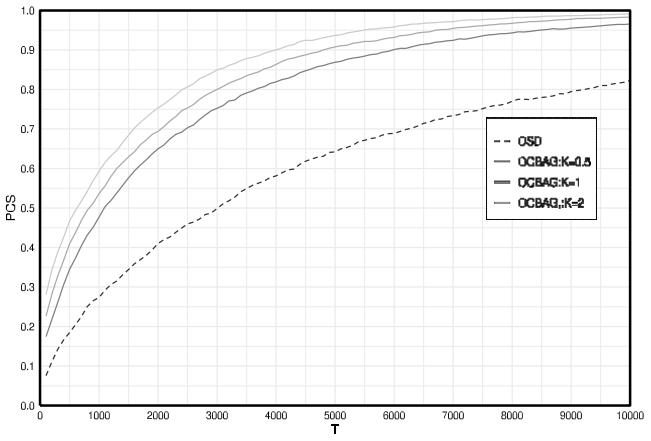


Fig. 10. Queueing simulation example.

consider the following objective function:

$$y(x) = E[S(x)] + cx = \frac{1}{x - \lambda} + cx$$

where $E[S(x)]$ is the average system time for a customer (queueing or delay time plus service time) and the additional term cx can be viewed as a cost on server speed (c is a given constant). Then, we let $c = 4$ and $\lambda = 0.5$ and use a design space consisting of 11 evenly spaced design points in $[0.55, 1.1]$. The global optimal solution is $x_9 \approx 0.99$ with $y(x_9) \approx 6$. For each design x_i, we run simulation with 5000 customers to get the performance estimate. At the same time, we use IPA method to obtain the gradient estimate without the need to run additional simulations. That is to say, the additional simulation cost of obtaining the gradient information is approximately equal to 0, i.e. $\kappa = 0$. We conduct the queueing experiment with T up to 800. Then, we repeat 1000 times and calculate the PCS obtained for three method (OCBAG, OSD and OCBA) out of these 1000 independent macro replications. For OSD method, we set $N_1^0 = N_{(k+1)/2}^0 = N_k^0 = 2$ and allocate 20 additional runs for each iteration, and for OCBAG method, we set n_0 to 2 and allocate 20 runs for each iteration. For this experiment, we include the results from the traditional OCBA as well. The results are illustrated in Fig. 10.

In spite of the correlated performance and gradient information and heterogeneous noise for all designs, OCBAG still performs better than OSD and traditional OCBA.

6. Conclusion

In this paper, we explore the potential of enhancing R&S efficiency by incorporating simulation performance and gradient information into a regression metamodel. Compared to the OSD method that motivated this work, we show that the inclusion of gradient information increases the accuracy of the regression metamodel estimated from noisy simulation output. Numerical experiments demonstrate that the inclusion of gradient information enables OCBAG to consistently outperform OSD and achieves much higher PCS than OSD, as well as a state-of-the-art R&S procedure OCBA. As to the best of our knowledge, this is the first work to demonstrate that gradient information can be used to improve the efficiency of an R&S procedure.

Though the use of gradient information and regression metamodels can dramatically enhance simulation efficiency, they are also constrained with some typical assumptions such as an underlining quadratic function for performance measures, homogeneous simulation noise and independence between performance

and gradient information. As shown in our numerical experiments, many of these assumptions can be alleviated if we can efficiently partition the design space so that we focus only on a local area of the design space where the assumptions hold. Specifically, on one hand, we should have more fine partitioning near the optimal point in order to have good fitting of the response surface with a quadratic model. On the other hand, for areas that have a small probability of containing the optimal design, we could use just a few rough partitions. Ideally, we want to have an 'smart partitioning' scheme which can maximize the overall efficiency, e.g., the probability of correct selection. The integration of OCBAG method with a smart partitioning mechanism for general simulation optimization problems is a direction for future research. Alternatively, another research topic worthwhile to investigate is the optimal design and budget allocation of other forms of regression functions, such as a higher order polynomial, that may provide better fit of the response surface. The dimensional curse is inherently a big challenge faced by any regression like procedures, including our approach. One promising approach we are taking as another ongoing research is to integrate our OCBAG method with some multi-dimensional search methods such as the stochastic trust region gradient-free method (Chang, Hong, & Wan, 2013). Another future research topic is to derive (near) optimal allocation policies without assuming the independence between performance and gradient estimates.

Appendix A. Proof of Theorem 1

Proof. Our proof here is similar to that of Theorem 3 in Brantley et al. (2013). To find the optimal solution to (13) for given support points $\{x_{s_1}, x_{s_2}\}$, we introduce its Lagrangian function $Q = APCS - \lambda\left(\alpha_{s_1} + \alpha_{s_2} - 1\right)$, where λ is the Lagrangian multiplier. Combining it with (12), we have

$$Q = 1 - \int_{-\infty}^{\hat{d}(x_A)/\sqrt{\zeta_A}} \phi(r)\mathrm{d}r - \int_{-\infty}^{\hat{d}(x_Z)/\sqrt{\zeta_Z}} \phi(r)\mathrm{d}r - \lambda\left(\alpha_{s_1} + \alpha_{s_2} - 1\right). \tag{A.1}$$

Using the chain rule, we can establish

$$\frac{\partial Q}{\partial \alpha_j} = -\frac{\partial Q}{\partial \zeta_A}\frac{\partial \zeta_A}{\partial \alpha_j} - \frac{\partial Q}{\partial \zeta_Z}\frac{\partial \zeta_Z}{\partial \alpha_j} - \lambda. \tag{A.2}$$

From (14) and (A.1), we obtain $\frac{\partial Q}{\partial \zeta_i} = \phi\left(\frac{\hat{d}(x_i)}{\sqrt{\zeta_i}}\right)\frac{\hat{d}(x_i)}{2(\zeta_i)^{3/2}}$ $(i = A, Z)$ and $\frac{\partial \zeta_i}{\partial \alpha_j} = \frac{\sigma^2}{T}\frac{D_{i,j}^2}{\alpha_j^2}$ $(j = s_1, s_2)$. We thus have

$$\begin{aligned}\frac{\partial Q}{\partial \alpha_j} &= \phi\left(\frac{\hat{d}(x_A)}{\sqrt{\zeta_A}}\right)\frac{\hat{d}(x_A)}{2(\zeta_A)^{3/2}}\frac{\sigma^2 D_{A,j}^2}{T\alpha_j^2} + \phi\left(\frac{\hat{d}(x_Z)}{\sqrt{\zeta_Z}}\right)\frac{\hat{d}(x_Z)}{2(\zeta_Z)^{3/2}}\frac{\sigma^2 D_{Z,j}^2}{T\alpha_j^2} - \lambda \\ &= 0,\end{aligned}$$

which is equivalent to

$$\frac{1}{\alpha_j^2}\left[\phi\left(\frac{\hat{d}(x_A)}{\sqrt{\zeta_A}}\right)\frac{\hat{d}(x_A)}{2(\zeta_A)^{3/2}}\frac{D_{A,j}^2}{\alpha_j^2}\right] + \phi\left(\frac{\hat{d}(x_Z)}{\sqrt{\zeta_Z}}\right)\frac{\hat{d}(x_Z)}{2(\zeta_Z)^{3/2}}\frac{D_{Z,j}^2}{\alpha_j^2} = \frac{2\lambda T}{\sigma^2}.$$

Taking $\alpha_{s_1} + \alpha_{s_2} = 1$ into consideration, we then have (15). Similar to Brantley et al. (2013), we can further show that $D_{i,j}^2$ is a concave function of α_{s_1} and α_{s_2}, hence APCS is also a concave function of α_{s_1} and α_{s_2} (see Brantley et al., 2013 for more details). This completes our proof.

Appendix B. Calculation of ζ_i

Given the fact that we only have simulation runs at two support points x_{s_1} and x_{s_1}, matrix $X^T V^{-1} X$ can be rewritten in terms of these two support points as:

$$\begin{aligned}X^T V^{-1} X &= \begin{pmatrix} 1 & 1 & 0 & 0 \\ x_{s_1} & x_{s_1} & 1 & 1 \\ x_{s_1}^2 & x_{s_2}^2 & 2x_{s_1} & 2x_{s_2} \end{pmatrix} \begin{pmatrix} \frac{T\alpha_{s_1}}{\sigma^2} & & & \\ & \frac{T\alpha_{s_1}}{\sigma^2} & & \\ & & \frac{T\alpha_{s_2}}{\sigma_g^2} & \\ & & & \frac{T\alpha_{s_2}}{\sigma_g^2} \end{pmatrix} \begin{pmatrix} 1 & x_{s_1} & x_{s_1}^2 \\ 1 & x_{s_2} & x_{s_2}^2 \\ 0 & 1 & 2x_{s_1} \\ 0 & 1 & 2x_{s_2} \end{pmatrix} \\ &= \frac{T}{\sigma^2} B,\end{aligned}$$

where

$$B = \begin{pmatrix} \gamma_{10} & \gamma_{11} & \gamma_{12} \\ \gamma_{11} & K\gamma_{10} + \gamma_{12} & 2K\gamma_{11} + \gamma_{13} \\ \gamma_{12} & 2K\gamma_{11} + \gamma_{13} & 4K2K\gamma_{12} + \gamma_{14} \end{pmatrix},$$

$$\gamma_{ij} = \alpha_{s_1}^i x_{s_1}^j + \alpha_{s_2}^i x_{s_2}^j.$$

The determinant and some of the algebraic complements of matrix B can be calculated as follows:

$$\begin{aligned} B^*_{22} &= 4K\gamma_{22} + \alpha_{s_1}\alpha_{s_2}\left[(x_{s_1}^2 - x_{s_2}^2)^2 + 4K(x_{s_1} - x_{s_2})^2\right], \\ B^*_{23} &= -2K\gamma_{21} - \alpha_{s_1}\alpha_{s_2}\left[x_{s_1} + x_{s_2}\right)\left(x_{s_1} - x_{s_2}\right)^2 + 2K\right], \\ B^*_{33} &= K\gamma_{20} + \alpha_{s_1}\alpha_{s_2}\left[x_{s_1} - x_{s_2}\right)^2 + 2K\right], \\ |B| &= K\alpha_{s_1}\alpha_{s_2}(x_{s_1} - x_{s_2})^2[(x_{s_1} - x_{s_2})^2 + 4K]. \end{aligned}$$

Combining them with (11), we have

$$\begin{aligned}\zeta_i &= \frac{\sigma_2}{T}\left[\frac{B^*_{22}}{|B|}(x_i - x_b)^2 + 2\frac{B^*_{23}}{|B|}(x_i - x_b)^2(x_i + x_b) + \frac{B^*_{33}}{|B|}(x_i - x_b)^2(x_i + x_b)^2\right] \\ &= \frac{\sigma^2(x_i - x_b)^2\alpha_{s_1}^2(2x_{s_1} - x_i - x_b)^2 + \alpha_{s_2}^2(2x_{s_2} - x_i - x_b)^2}{T\alpha_{s_1}\alpha_{s_2}(x_{s_1} - x_{s_2})^2[(x_{s_1} - x_{s_2})^2 + 4K]} \\ &+ \frac{\sigma^2(x_i - x_b)^2\alpha_{s_1}\alpha_{s_2}(x_{s_1} - x_{s_2})^2(x_{s_1} + x_{s_2} - x_i - x_b)^2}{TK\alpha_{s_1}\alpha_{s_2}(x_{s_1} - x_{s_2})^2[(x_{s_1} - x_{s_2})^2 + 4K]} \\ &+ \frac{2K\sigma^2(x_i - x_b)^2\alpha_{s_1}\alpha_{s_2}(x_{s_1} + x_{s_2} - x_i - x_b)^2}{TK\alpha_{s_1}\alpha_{s_2}(x_{s_1} - x_{s_2})^2[(x_{s_1} - x_{s_2})^2 + 4K]} \\ &+ \frac{2K\sigma^2(x_i - x_b)^2\alpha_{s_1}\alpha_{s_2}(x_{s_1} - x_{s_2})^2}{TK\alpha_{s_1}\alpha_{s_2}(x_{s_1} - x_{s_2})^2[(x_{s_1} - x_{s_2})^2 + 4K]}\end{aligned} \tag{B.1}$$

Since $\alpha_{s_1} + \alpha_{s_2} = 1$, we can rewrite $\alpha_{s_1}^2$ and $\alpha_{s_2}^2$ as $\alpha_{s_1}(1 - \alpha_{s_2})$ and $\alpha_{s_2}(1 - \alpha_{s_1})$. Substituting them into (B.1) and after some simplification, we can obtain (14).

Appendix C. Minimizing ζ_M

This yields

$$\frac{\sqrt{q_{s_1}}}{\alpha_{s_1}} = \frac{\sqrt{q_{s_2}}}{\alpha_{s_2}}.$$

We denote $x_c = (x_b + x_M)/2$. For notational simplicity, let

$$\dot{\zeta}_M = \frac{T}{\sigma^2(x_M - x_b)^2}\zeta_M, \tag{A.3}$$

$$d_1 = \min\{x_c - x_1, x_k - x_c\},$$
$$d_2 = \max\{x_c - x_1, x_k - x_c\}.$$

Combining (14) with (17) and (18), we have

$$\hat{\zeta}_M = \frac{(x_{s_1} + x_{s_2} - 2x_c)^2}{K[(x_{s_1} - x_{s_2})^2 + 4K]} + \frac{4(x_{s_1} - x_c)^2 + 8|x_{s_1} - x_c||x_{s_2} - x_c| + 4(x_{s_2} - x_c)^2}{(x_{s_1} - x_{s_2})^2[(x_{s_1} - x_{s_2})^2 + 4K]}.$$

To minimize ζ_M is the same as to minimize $\hat{\zeta}_M$. We need to consider two cases in which $(x_{s_1} - x_c)(x_{s_2} - x_c)$ is either positive or negative.

Case 1: $(x_{s_1} - x_c)(x_{s_2} - x_c) \geq 0$. In this case, we have $x_{s_2} > x_{s_1} \geq x_c$ or $x_{s_1} < x_{s_2} \leq x_c$ (which means that x_{s_1} and x_{s_2} are on the same side of point x_c), hence

$$\begin{aligned}\hat{\zeta}_M &= \frac{4(x_{s_1} + x_{s_2} - 2x_c)^2}{(x_{s_1} - x_{s_2})^2[(x_{s_1} - x_{s_2})^2 + 4K]} + \frac{(x_{s_1} + x_{s_2} - 2x_c)^2}{K[(x_{s_1} - x_{s_2})^2 + 4K]} \\ &= \frac{1}{K}\frac{(x_{s_1} + x_{s_2} - 2x_c)^2}{(x_{s_1} - x_{s_2})^2} \\ &= \frac{1}{K}\frac{(x_{s_1} - x_{s_2})^2 + 4(x_{s_1} - x_c)(x_{s_2} - x_c)}{(x_{s_1} - x_{s_2})^2} \\ &\geq \frac{1}{K}.\end{aligned}$$

Case 2: $(x_{s_1} - x_c)(x_{s_2} - x_c) \leq 0$. In this case, we have $x_{s_2} \geq x_c > x_{s_1}$ (which means that x_{s_1} and x_{s_2} are on the different sides of point x_c), therefore

$$\hat{\zeta}_M = \frac{4}{(x_{s_1} - x_{s_2})^2 + 4K} + \frac{(x_{s_1} + x_{s_2} - 2x_c)^2}{K[(x_{s_1} - x_{s_2})^2 + 4K]}.$$

If we choose the two support points such that $x_{s_1} + x_{s_2} - 2x_c = 0$, then

$$\hat{\zeta}_M = \frac{4}{(x_{s_1} - x_{s_2})^2 + 4K} < \frac{1}{K}.$$

Combining the results from Cases 1 and 2, it is clear that in order to minimize $\hat{\zeta}_M$ we should choose the support points x_{s_1} and x_{s_2} on different sides of x_c, i.e., we only need to consider Case 2. Hence, we rewrite $\hat{\zeta}_M$ as:

$$\begin{aligned}\hat{\zeta}_M &= \frac{4}{(x_{s_1} - x_{s_2})^2 + 4K} + \frac{(x_{s_1} + x_{s_2} - 2x_c)^2}{K[(x_{s_1} - x_{s_2})^2 + 4K]} \\ &= \frac{4K + (x_{s_1} + x_{s_2} - 2x_c)^2}{K[(x_{s_1} - x_{s_2})^2 + 4K]}.\end{aligned}$$

We now further consider two cases:

Case 2.1: $x_{s_2} - x_{s_1} \leq 2d_1$. Without loss of generality, we assume $x_c - x_1 < x_k - x_c$. It is clear that to minimize $\hat{\zeta}_M$ we want to maximize $x_{s_2} - x_{s_1}$, i.e., to keep the two support points as far apart as possible, and on the other hand to minimize $|x_{s_1} + x_{s_2} - 2x_c|$. Hence, $\{x_{s_1}, x_{s_2}\} = \{x_1, x_1 + 2d_1\}$ is the optimal solution and the minimum of $\hat{\zeta}_M$ is $\frac{1}{K + d_1^2}$.

Case 2.2: $x_{s_2} - x_{s_1} \geq 2d_1$. We still assume $x_c - x_1 < x_k - x_c$. First, in order to increase $x_{s_2} - x_{s_1}$ and decrease $|x_{s_1} + x_{s_2} - 2x_c|$, we should select $s_1 = 1$. Next, let $x_{s_2} = x_1 + 2d_1 + \delta$ ($\delta \geq 0$), we then have

$$\hat{\zeta}_M = \frac{4K + \delta^2}{K[(2d_1 + \delta)^2 + 4K]},$$

which is minimized when

$$\delta^* = \begin{cases} \sqrt{d_1^2 + 4K}, & \text{if } d_2 - d_1 \geq \sqrt{d_1^2 + 4K} - d_1; \\ d_2 - d_1, & \text{if } d_2 - d_1 \leq \sqrt{d_1^2 + 4K} - d_1. \end{cases}$$

Putting the above results together, we have (19). It is also worthwhile pointing out when x_{s_1} or x_{s_2} provided by (19) does not coincide with any of the design points, our derivations here in fact provide an easy way to choose a design point. Usually, we can simply choose the one closest to x_{s_1} or x_{s_2}.

References

Atkinson, A., Donev, A., & Tobias, R. (2007). *Optimum experimental designs, with SAS, volume 34*. Oxford University Press.

Branke, J., Chick, S. E., & Schmidt, C. (2007). Selecting a selection procedure. *Management Science*, *53*(12), 1916–1932.

Brantley, M. W., Lee, L. H., Chen, C. H., & Chen, A. (2013). Efficient simulation budget allocation with regression. *IIE Transactions*, *45*(3), 291–308.

Brantley, M. W., Lee, L. H., Chen, C. H., & Xu, J. (2014). An efficient simulation budget allocation method incorporating regression for partitioned domains. *Automatica*, *50*(5), 1391–1400.

Chang, K. H., Hong, L. J., & Wan, H. (2013). Stochastic trust-region response-surface method (STRONG)—A new response-surface framework for simulation optimization. *INFORMS Journal on Computing*, *25*(2), 230–243.

Chen, X., Ankenman, B. E., & Nelson, B. L. (2013). Enhancing stochastic kriging metamodels with gradient estimators. *Operations Research*, *61*(2), 512–528.

Chen, W., Gao, S., Chen, C. H., & Shi, L. (2013). An optimal sample allocation strategy for partition-based random search. *IEEE Transactions on Automation Science and Engineering*, *11*(1), 177–186.

Chen, C. H., He, D., Fu, M., & Lee, L. H. (2008). Efficient simulation budget allocation for selecting an optimal subset. *INFORMS Journal on Computing*, *20*(4), 579–595.

Chen, C. H., & Lee, L. H. (2011). *Stochastic simulation optimization: An optimal computing budget allocation, volume 1*. World scientific.

Chen, C. H., Lin, J., Yücesan, E., & Chick, S. E. (2000). Simulation budget allocation for further enhancing the efficiency of ordinal optimization. *Discrete Event Dynamic Systems*, *10*(3), 251–270.

Chen, Y., & Ryzhov, I. O. (2019). Complete expected improvement converges to an optimal budget allocation. *Advances in Applied Probability*, *51*(1), 209–235.

Chen, C. H., Yücesan, E., Dai, L., & Chen, H. C. (2009). Optimal budget allocation for discrete-event simulation experiments. *IIE Transactions*, *42*(1), 60–70.

DeGroot, M. H. (2005). *Optimal statistical decisions, volume 82*. John Wiley & Sons.

Draper, N. R., & Smith, H. (1998). *Applied regression analysis, volume 326*. John Wiley & Sons.

Fu, M. C., & Hu, J. Q. (1997). *Conditional Monte Carlo: Gradient estimation and optimization applications*. Kluwer Academic.

Fu, M. C., & Qu, H. (2014). Regression models augmented with direct stochastic gradient estimators. *INFORMS Journal on Computing*, *26*(3), 484–499.

Gao, S., Chen, W., & Shi, L. (2017). A new budget allocation framework for the expected opportunity cost. *Operations Research*, *65*(3), 787–803.

Gao, S., Xiao, H., Zhou, E., & Chen, W. (2017). Robust ranking and selection with optimal computing budget allocation. *Automatica*, *81*, 30–36.

De la Garza, A. (1954). Spacing of information in polynomial regression. *The Annals of Mathematical Statistics*, *25*(1), 123–130.

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2013). *Bayesian data analysis* (pp. 353–357). CRC Press.

Glasserman, P. (1991). *Gradient estimation via perturbation analysis*. Kluwer Academic.

Glynn, P. W. (1990). Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM*, *33*(10), 75–84.

Ho, Y. C., & Cao, X. R. (1991). *Perturbation analysis of discrete event dynamic systems*. Kluwer Academic.

Lee, L. H., Chen, C. H., Chew, E. P., Li, J., Pujowidianto, N. A., & Zhang, S. (2010). A review of optimal computing budget allocation algorithms for simulation optimization problem. *International Journal of Operations Research*, *7*(2), 19–31.

Lee, L. H., Pujowidianto, N. A., Li, L. W., Chen, C. H., & Yap, C. M. (2012). Approximate simulation budget allocation for selecting the best design in the presence of stochastic constraints. *IEEE Transactions on Automatic Control*, *57*(11), 2940–2945.

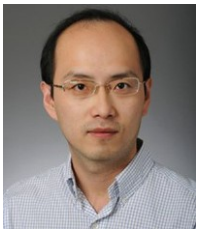
Liski, E. P., Mandal, N. K., Shah, K. R., & Sinha, B. (2002). *Topics in optimal design, volume 163*. Springer Science & Business Media.

Pasupathy, R., Hunter, S. R., Pujowidianto, N. A., Lee, L. H., & Chen, C. H. (2014). Stochastically constrained ranking and selection via SCORE. *ACM Transactions on Modeling and Computer Simulation*, *25*(1), 1–26.

Peng, Y., Chong, E. K., Chen, C. H., & Fu, M. C. (2018). Ranking and selection as stochastic control. *IEEE Transactions on Automatic Control*, *63*(8), 2359–2373.

Peng, Y., Xu, J., Lee, L. H., Hu, J., & Chen, C. H. (2018). Efficient simulation sampling allocation using multifidelity models. *IEEE Transactions on Automatic Control*, *64*(8), 3156–3169.
Qu, H., & Fu, M. C. (2014). Gradient extrapolated stochastic kriging. *ACM Transactions on Modeling and Computer Simulation*, *24*(4), 1–25.
Ryzhov, I. O. (2016). On the convergence rates of expected improvement methods. *Operations Research*, *64*(6), 1515–1528.
Schwartz, J. D., Wang, W., & Rivera, D. E. (2006). Simulation-based optimization of process control policies for inventory management in supply chains. *Automatica*, *42*(8), 1311–1320.
Teng, S., Lee, L. H., & Chew, E. P. (2007). Multi-objective ordinal optimization for simulation optimization problems. *Automatica*, *43*(11), 1884–1895.
Törn, A., & Žilinskas, A. (1989). *Global optimization, volume 350*. Springer.
Xiao, H., Chen, H., & Lee, L. H. (2019). An efficient simulation procedure for ranking the top simulated designs in the presence of stochastic constraints. *Automatica*, *103*, 106–115.
Xiao, H., Gao, S., & Lee, L. H. (2017). Simulation budget allocation for simultaneously selecting the best and worst subsets. *Automatica*, *84*, 117–127.
Xiao, H., Gao, F., & Lee, L. H. (2019). Optimal computing budget allocation for complete ranking with input uncertainty. *IISE Transactions*, 1–11.
Xiao, H., Lee, L. H., & Ng, K. M. (2013). Optimal computing budget allocation for complete ranking. *IEEE Transactions on Automation Science and Engineering*, *11*(2), 516–524.
Xu, J., Huang, E., Hsieh, L., Lee, L. H., Jia, Q. S., & Chen, C. H. (2016). Simulation optimization in the era of Industrial 4.0 and the Industrial Internet. *Journal of Simulation*, *10*(4), 310–320.
Zhang, F., Song, J., Dai, Y., & Xu, J. (2020). Semiconductor wafer fabrication production planning using multi-fidelity simulation optimisation. *International Journal of Productions Research*, 1–16.

Tianxiang Wang is a Postdoctoral Researcher in School of Management, Fudan University. His research interests include financial engineering, stochastic optimization via simulation, and operations management. He received his BS degree in Mathematics Sciences from Fudan University and his Ph.D. degree in Management Science and Engineering from Fudan University.



Jie Xu received the Ph.D. degree in industrial engineering and management sciences from Northwestern University, the M.S. degree in computer science from The State University of New York, Buffalo, the M.E. degree in electrical engineering from Shanghai Jiaotong University, and the B.S. degree in electrical engineering from Nanjing University. He is currently an Associate Professor of Systems Engineering and Operations Research at George Mason University. His research interests are data analytics, stochastic simulation and optimization, with applications in cloud computing, manufacturing, and power systems.

Jian-Qiang Hu is the Distinguished Professor of Fudan University and the Hongyi Professor of Management Science in School of Management, Fudan University. He received his B.S. degree in applied mathematics from Fudan University, China, and M.S. and Ph.D. degrees in applied mathematics from Harvard University. His research interests include discrete-event stochastic systems, simulation, stochastic optimization, with applications in supply chain management, financial engineering, and healthcare. He won the Outstanding Simulation Publication Award from INFORMS Simulation Society twice (1998, 2019) and the Outstanding Research Award from Operations Research Society of China in 2020. He has been on editorial board of *Automatica*, *Operation Research*, *IIE Transaction on Design and Manufacturing*, and *Journal of the Operations Research Society of China*.

Chun-Hung Chen received his Ph.D. from Harvard University. He is a Professor at George Mason University and had been faculty with University of Pennsylvania and National Taiwan University. He has served as a Department Editor for *IIE Transactions*, Department Editor for *Asia-Pacific Journal of Operational Research*, Associate Editor for *IEEE Transactions on Automation Science and Engineering*, Associate Editor for *IEEE Transactions on Automatic Control*, Area Editor for *Journal of Simulation Modeling Practice and Theory*. Dr. Chen is the author of two books, including a best seller: ''*Stochastic Simulation Optimization: An Optimal Computing Budget Allocation*''. He is an IEEE Fellow.