Efficient computing budget allocation for finding simplest good designs

QING-SHAN JIA1,∗, ENLU ZHOU2 and CHUN-HUNG CHEN3,4

1 Center for Intelligent and Networked Systems, Department of Automation, TNLIST, Tsinghua University, Beijing 100084, China
E-mail: jiaqs@tsinghua.edu.cn
2 Department of Industrial & Enterprise Systems Engineering, University of Illinois at Urbana–Champaign, IL 61801, USA
3 Department of Electrical Engineering and Institute of Industrial Engineering, National Taiwan University, Taipei, Taiwan
4 Department of Systems Engineering & Operations Research, George Mason University, Fairfax, VA 22030, USA

Received June 2011 and accepted May 2012

In many applications some designs are easier to implement, require less training data and shorter training time, and consume less storage than others. Such designs are called simple designs and are usually preferred over complex ones when they all have good performance. Despite the abundant existing studies on how to find good designs in simulation-based optimization, there exist few studies on finding simplest good designs. This article considers this important problem and the following contributions are made to the subject. First, lower bounds are provided for the probabilities of correctly selecting the simplest designs with top performance and selecting the best such simplest good designs, respectively. Second, two efficient computing budget allocation methods are developed to find simplest good designs and to find the best such designs, respectively, and their asymptotic optimality has been shown. Third, the performance of the two methods is compared with equal allocations over six academic examples and a smoke detection problem in a wireless sensor network.

Keywords: Simulation-based optimization, complexity preference, optimal computing budget allocation, wireless sensor network

1. Introduction

Many systems nowadays follow not only physical laws but also man-made rules. These systems are called discrete-event dynamic systems and the optimization of their performance enters the realm of Simulation-Based Optimization (SBO). In many applications some designs are easier to implement, require less training data and shorter training time, and consume less storage than the others. Such designs are called simple designs and are usually preferred over complex ones when they all have good performance. For example, a wireless sensor network can be used to detect smoke. While a larger sensing radius allows the smoke to be detected faster, it also increases the power consumption and shortens the lifetime of the network. Thus, when the detection is fast enough, a small sensing radius is preferred over a large one.

There are abundant existing studies on SBO. Ranking and Selection (R&S) procedures are well-known procedures to combine simulation and optimization to improve efficiency levels. Their origin can be traced back to two papers, namely, Bechhofer (1954) for the indifference-zone formulation and Gupta (1956, 1965) for the subset selection formulation. Excellent surveys on R&S can be found in Bechhofer et al. (1995), Kim and Nelson (2003), and Swisher et al. (2003). Chen et al. (2000) and Chen and Yücesan (2005) developed the Optimal Computing Budget Allocation (OCBA) procedure to asymptotically maximize a lower bound of the probability of correctly selecting the best solution candidate. OCBA was later extended to select the best several solution candidates (Chen et al., 2008) and to handle stochastic constraints (Pujowidianto et al., 2009), multiple objective functions (Lee et al., 2004), correlated observation noises (Fu et al., 2007), and opportunity cost (He et al., 2007). A comprehensive introduction to OCBA was recently provided by Chen and Lee (2011). Recent surveys on other methods for SBO can be found in Andradóttir (1998), Fu (2002), Swisher et al. (2004), Tekin and Sabuncuoglu (2004), and Fu et al. (2008).

Despite the abundance of existing studies on finding good designs in SBO, there exist few studies on finding simplest good designs. This problem is challenging due to the following difficulties. First, simulation-based performance evaluation is usually time-consuming and provides only noisy performance estimation. The second difficulty is randomness. Due to the pervasive randomness in the system dynamics, usually multiple runs are required in order to

∗Corresponding author
obtain accurate estimation. The last difficulty is complexity preference for simple designs, which requires one to consider both the performance and the complexity simultaneously. Existing studies on SBO usually handle the first two difficulties well, but very few consider the complexity preference, which is the main contribution of this article.

The preference for simple designs has been mathematically described by the Kolmogorov complexity (Kolmogorov, 1965), Solomonoff’s universal prior (Solomonoff, 1964, 1978), the Levin complexity (Levin, 1973), and the AIXI model (Hutter, 2005), to name just a few. Most of these formulations assume that the performance of designs can be easily evaluated and thus do not address the unique difficulties of simulation-based performance evaluations. The study of SBO with complexity preference has only recently been considered. Jia and Zhao (2009) studied the relationship between performance and complexity of different policies in Markov Decision Processes (MDPs). Jia (2011b) showed that it is possible to obtain simple policies with good performance in MDPs. Jia (2009, 2011a) considered SBO with descriptive complexity preference and developed an adaptive sampling algorithm to find the simplest design with bounded cardinal performance. Later on, Yan et al. (2010, 2012) developed two algorithms (OCBAmSG and OCBAbSG) to find m simplest designs with bounded cardinal performance and to find the best m such designs, respectively. The above methods suit applications where there are clear bounds on the cardinal performance of designs. However, in many applications it is difficult to estimate the performance of the best design a priori, which makes it difficult to identify “good” designs in a cardinal sense. Ho et al. (2007) showed that the probability of correctly identifying the relative order among two designs converges to unity exponentially fast with respect to (w.r.t.) the number of observations that are taken for each design. Note that the standard deviation of cardinal performance estimation using Monte Carlo simulation only converges in the rate of 1/\( \sqrt{n} \), where n is the number of observations. Thus, in comparison, one finds that the ordinal values converge much faster than the cardinal ones. Since in many applications we want to find simple designs with the best performance, we focus this article on finding simplest good designs in the ordinal sense.

In this article, we consider the important problem of how to allocate the computing budget so that the simplest good designs can be found with high probability and make the following major contributions. First, we mathematically formulate two related problems. One is how to find m simplest designs that have top-g performance. When g > m there could be multiple choices for m such designs. For example, suppose \( \theta_1, \theta_2, \text{and } \theta_3 \) are the best, the second best, and the third best designs, respectively. Also, suppose their complexities are the same. When g = 3 and m = 2, there are three choices for m simplest designs with top-g performance, namely, \( \{\theta_1, \theta_2\}, \{\theta_1, \theta_3\}, \text{and } \{\theta_2, \theta_3\} \). Clearly, the choice of \( \{\theta_1, \theta_2\} \) has the best performance. Thus, another related problem is how to find m simplest top-g designs that have the best performance among all of the choices. We develop lower bounds for the probabilities of correctly selecting the two subsets of m simplest good designs, which are denoted as PCSm and PCSb, respectively. Second, we develop efficient computing budget allocation methods to asymptotically optimize the two PCS functions, respectively. The two methods are called Optimal Computing Budget Allocation for m Simplest Good Designs in the ordinal sense (OCBAmSGO) and Optimal Computing Budget Allocation for the best m Simplest Good Designs in the ordinal sense (OCBAbSGO), respectively. Then we numerically compare their performance with equal allocation on academic examples and a smoke detection problem in Wireless Sensor Networks (WSNs).

The rest of this article is organized as follows. We mathematically formulate the two problems in Section 2, present the main results in Section 3, show the experimental results in Section 4, and draw conclusions in Section 5.

2. Problem formulation

In this section we define the m Simplest Good designs (or mSG for short) and the Best m Simplest Good designs (or bSG for short) using the true performance of the designs in Section 2.1 and define the probabilities of correct selection based on a Bayesian model in Section 2.2.

2.1. Definitions of mSG and bSG

Consider a search space of k competing designs \( \Theta = \{\theta_1, \ldots, \theta_k\} \). Let \( J(\theta) \) be the performance of \( \theta \), which can be accurately evaluated only through an infinite number of replications; that is,

\[
J(\theta) = \lim_{n \to \infty} \frac{1}{n} \sum_{i=1}^{n} J(\theta, \xi_i),
\]

where

\[
\hat{J}(\theta, \xi_i) = J(\theta) + w(\theta, \xi_i)
\]

is the observation; \( w(\theta, \xi_i) \) is the observation noise, which has independent Gaussian distribution \( \mathcal{N}(0, \sigma^2(\theta)) \); and \( \xi_i \) represents the randomness in the \( i \)th sample path. Suppose we are considering minimization problems. Sort all of the designs from the smallest to the largest according to \( J(\theta) \). Let \( G \) be the set of the top-g (g < k) designs; i.e., \( G = \{\theta_1, \ldots, \theta_g\} \), where \( \theta_j \) is the top \( j \)th design. Let \( C(\theta) \) be the complexity of design \( \theta \). Without loss of generality, we assume \( M \) different complexities; i.e., \( \{C(\theta), \theta \in \Theta\} = \{1, \ldots, M\} \). Let \( \Theta_i \) denote designs with complexity \( i \); i.e., \( \Theta_i = \{\theta \in \Theta, C(\theta) = i\} \). Then each \( \Theta_i \) can be divided into two subsets, namely, \( G_i = \Theta_i \cap G \), which contains all the good designs with complexity \( i \), and \( D_i = \Theta_i \setminus G_i \), which contains all of the rest of the designs with complexity \( i \), where \( \setminus \) represents set minus. We then
have \( \bigcup_{i=1}^{M} G_i = G \) and \( D = \Theta \setminus G \). It is possible that \( G_i = \emptyset \) or \( D_i = \emptyset \) for some \( i \). Throughout this article, we make the following assumption:

1. The complexities of all the designs are known; i.e., \( C(\theta) \) is known for all \( \theta \in \Theta \).
2. When selecting among good designs, complexity has priority over performance.

A set of designs \( S \) is called the \( m \) \((m < g)\) simplest good designs (or mSG for short) if all of the following conditions are satisfied.

1. \( |S| = m \),
2. \( S \subset G \),
3. \( \max_{\theta \in S} C(\theta) \leq \min_{\theta \in G \setminus S} C(\theta) \).

Furthermore, a set of designs \( S \) is called the best mSG (or bSG for short) if all of the following conditions are satisfied.

1. \( S \) is mSG,
2. if there exists \( \theta \in S \) and \( \theta' \in G \setminus S \) s.t. \( C(\theta) = C(\theta') \), then \( J(\theta) \leq J(\theta') \).

Note that the first condition above clearly shows that a bSG must be an mSG. The second condition shows that a bSG has the best performance among all mSGs.

### 2.2. Definitions of probabilities of correct selection

Note that in the above definitions of \( G \), \( D \), mSG, and bSG the true performance \( J(\theta) \) is used, which can be obtained only when an infinite number of replications is used. In practice only a finite number of replications are affordable; that is,

\[
\hat{J}(\theta) = \frac{1}{n(\theta)} \sum_{i=1}^{n(\theta)} J(\theta, \xi_i). \tag{3}
\]

We follow the Bayesian model as in Chen (1996), Chen et al. (2000), and Chen et al. (2008). The mean of the simulation output for each design, \( J(\theta) \), is assumed unknown and treated as a random variable. After the simulation is performed, a posterior distribution for the unknown mean \( J(\theta) \), \( p(J(\theta) | \hat{J}(\theta, \xi_i), i = 1, \ldots, n(\theta)) \), is constructed based on two pieces of information: (i) prior knowledge of the system’s performance and (ii) current simulation output. As in Chen (1996), we assume that the unknown mean \( J(\theta) \) has a conjugate normal prior distribution and consider noninformative prior distributions, which implies that no prior knowledge is available about the performance of any design before conducting the simulations, in which case the posterior distribution of \( J(\theta) \) is (cf. DeGroot (1970)):

\[
\hat{J}(\theta) \sim N(\bar{J}(\theta), \sigma^2(\theta)/n(\theta)). \tag{4}
\]

Given \( \hat{J} \), then \( G \) and \( D \) are both random sets. Thus, we define the probability of correctly selecting an mSG as

\[
PCS_m \equiv \Pr \{ S \text{ is mSG} \}
= \Pr \left\{ |S| = m, S \subseteq G, \max_{\theta \in S} C(\theta) \leq \min_{\theta \in G \setminus S} C(\theta) \right\}, \tag{5}
\]

and define the probability of correctly selecting a bSG as

\[
PCS_b \equiv \Pr \{ S \text{ is bSG} \}
= \Pr \left\{ |S| = m, S \subseteq G, \max_{\theta \in S} C(\theta) \leq \min_{\theta \in G \setminus S} C(\theta) \right\}.
\tag{6}
\]

Now we can mathematically formulate the following two problems:

- (P1) \( \max_{\theta_1, \ldots, \theta_k} PCS_m \) s.t. \( \sum_{i=1}^{k} n(\theta_i) = T \);
- (P2) \( \max_{\theta_1, \ldots, \theta_k} PCS_b \) s.t. \( \sum_{i=1}^{k} n(\theta_i) = T \);

where \( n(\theta_i) \) is the number of replications allocated to design \( \theta_i \). We will provide methods to address the above two problems in the next section.

### 3. Main results

In this section we address problems P1 and P2 in subsections 3.1 and 3.2, respectively.

#### 3.1. Selecting an mSG

Given the observed performance of all of the designs, we can divide the entire design space \( \Theta \) into at most \( 2M \) subsets, namely, \( \bar{G}_1, \ldots, \bar{G}_M \) and \( \bar{D}_1, \ldots, \bar{D}_M \), where \( \bar{G} \) and \( \bar{D} \) represent the observed top-\( g \) designs and the rest of the designs, respectively, \( \bar{G}_i = \Theta \setminus \bar{G} \) and \( \bar{D}_i = \Theta \setminus \bar{G}_i \). Though there are multiple choices of an mSG, we are specifically interested in the following procedure to select an mSG. Initially, set \( S = \emptyset \). We start from \( \bar{G}_1 \) and add designs from \( \bar{G}_1 \) to \( S \) from the smallest to the largest according to their observed performance \( \bar{J}(\theta) \). When all of the designs in \( \bar{G}_i \) have been added to \( S \) and \( |S| < m \), we move on to \( \bar{G}_{i+1} \) and continue the above procedure until \( |S| = m \). Suppose \( t \) satisfies \( \sum_{i=1}^{t} |\bar{G}_i| < m \leq \sum_{i=1}^{t+1} |\bar{G}_i| \). Let \( \theta_{i,j} \) be the observed \( j \)th best design in \( \theta_i \). Then we have

\[
S = \left( \bigcup_{i=1}^{t} \bar{G}_i \right) \cup \left\{ \theta_{t+1,1}, \ldots, \theta_{t+1,m-\sum_{l=1}^{t} |\bar{G}_l|} \right\}. \tag{7}
\]

This specific choice of \( S \) contains the best simplest \( m \) designs in \( G \). It is not only an estimate of mSG under the given \( \bar{J}(\theta) \) values but also is an estimate of bSG. This latter point will be explored later in Section 3.2.
Note that complexity has been considered in the choice of $S$ in Equation (7). We have $|S| = m$, $S \subseteq \bar{G}$, and $\max_{\theta \in \hat{G}} C(\theta) \leq \min_{\theta' \in \hat{G}} C(\theta')$. Then following equation (5) we have

$$\begin{align*}
PCSm &= \Pr\{S \text{ is mSG}\} \\
&= \Pr\{|S| = m, S \subseteq \bar{G}, \max_{\theta \in S} C(\theta) \leq \min_{\theta' \in \hat{G}} C(\theta')\} \\
&= \Pr\{S \subseteq \bar{G}, \max_{\theta \in S} C(\theta) \leq \min_{\theta' \in \hat{G}} C(\theta'), G = \bar{G}\} \\
&\quad + \Pr\{S \subseteq \bar{G}, \max_{\theta \in S} C(\theta) \leq \min_{\theta' \in \hat{G}} C(\theta'), G \neq \bar{G}\} \\
&\geq \Pr\{S \subseteq \bar{G} \subseteq \bar{G}, G = \bar{G}\} \\
&\quad \times \Pr\{G = \bar{G}\} \\
&= \Pr\{\bar{G} \subseteq \bar{G} \subseteq \bar{G}, G = \bar{G}\} \\
&\quad \times \Pr\{\bar{G} \subseteq \bar{G} \subseteq \bar{G}, G = \bar{G}\} \\
&\quad \geq \Pr\{\bar{G} \subseteq \bar{G} \subseteq \bar{G}, G = \bar{G}\} \\
&\quad \times \Pr\{\bar{G} \subseteq \bar{G} \subseteq \bar{G}, G = \bar{G}\}.
\end{align*}$$

where $\mu_1$ is a given constant. Due to the independence between the $\bar{G}$. values, we have

$$\begin{align*}
\Pr\{\bar{G} \subseteq \bar{G} \subseteq \bar{G}, G = \bar{G}\} &= \Pr\{\bar{G} \subseteq \bar{G} \subseteq \bar{G}, G = \bar{G}\} \\
&= \Pr\{\bar{G} \subseteq \bar{G} \subseteq \bar{G}, G = \bar{G}\} \\
&= \Pr\{\bar{G} \subseteq \bar{G} \subseteq \bar{G}, G = \bar{G}\} \\
&= \Pr\{\bar{G} \subseteq \bar{G} \subseteq \bar{G}, G = \bar{G}\} \\
&= \Pr\{\bar{G} \subseteq \bar{G} \subseteq \bar{G}, G = \bar{G}\}.
\end{align*}$$

Following Equations (8) and (9), we have the following Approximate PCSm (APCSm),

$$\begin{align*}
APCSm &\equiv \Pr\{\bar{G} \subseteq \bar{G} \subseteq \bar{G}, G = \bar{G}\} \\
&= \Pr\{\bar{G} \subseteq \bar{G} \subseteq \bar{G}, G = \bar{G}\} \\
&= \Pr\{\bar{G} \subseteq \bar{G} \subseteq \bar{G}, G = \bar{G}\} \\
&= \Pr\{\bar{G} \subseteq \bar{G} \subseteq \bar{G}, G = \bar{G}\}.
\end{align*}$$

Then an approximate version of PI is

- (AP1) max$_{n(\theta_1), \ldots, n(\theta_k)}$ APCSms s.t. $\sum_{i=1}^k n(\theta_i) = T$.

Note that the idea of APCSms is that PCSm is lower bounded by the probability that all observed good enough designs are truly good enough, which can be asymptotically maximized if we follow the allocation procedure in Chen et al. (2008). This explains why the following Theorem 1 leads to a similar allocation procedure as in Chen et al. (2008). We briefly provide the analysis as follows.

Let $L_m$ be the Lagrangian relaxation of AP1:

$$\begin{align*}
L_m &= APCSms + \lambda_m \left(\sum_{i=1}^k n(\theta_i) - T\right) \\
&= \Pr\{\bar{G} \subseteq \bar{G} \subseteq \bar{G}, G = \bar{G}\} \\
&\quad \times \Pr\{\bar{G} \subseteq \bar{G} \subseteq \bar{G}, G = \bar{G}\} \\
&\quad \times \Pr\{\bar{G} \subseteq \bar{G} \subseteq \bar{G}, G = \bar{G}\} \\
&\quad \geq \Pr\{\bar{G} \subseteq \bar{G} \subseteq \bar{G}, G = \bar{G}\} \\
&\quad \times \Pr\{\bar{G} \subseteq \bar{G} \subseteq \bar{G}, G = \bar{G}\} \\
&\quad \geq \Pr\{\bar{G} \subseteq \bar{G} \subseteq \bar{G}, G = \bar{G}\}.
\end{align*}$$

The Karush–Kuhn–Tucker (KKT) conditions (cf. Walker (1999)) of AP1 are as follows. For $\theta \in \hat{G}$:

$$\begin{align*}
\frac{\partial L_m}{\partial n(\theta)} &= (\Pi_{\theta' \in G, \theta' \neq \theta} \Pr\{\bar{G}(\theta') \leq \mu_1\}) \\
&\quad \times (\Pi_{\theta' \in D, \theta' \neq \theta} \Pr\{\bar{G}(\theta') > \mu_1\}) \\
&\quad \times \phi\left(\frac{\bar{G}(\theta) - \mu_1}{\sigma(\theta)/\sqrt{n(\theta)}}\right) + \lambda_m = 0.
\end{align*}$$

for $\theta \in \hat{D}$:

$$\begin{align*}
\frac{\partial L_m}{\partial \lambda_m} &= \sum_{i=1}^k n(\theta_i) - T = 0,
\end{align*}$$

where $\phi(\cdot)$ is the probability density function of the standard Gaussian distribution. We now analyze the relationship between $n(\theta)$ and $n(\theta')$. First, we consider the case that $\theta$ and $\theta' \in \hat{G}$ and $\theta' \neq \theta$. Following Equation (12) we have

$$\begin{align*}
\Pi_{\theta' \in G, \theta' \neq \theta} \Pr\{\bar{G}(\theta') \leq \mu_1\} \Pi_{\theta' \in D, \theta' \neq \theta} \Pr\{\bar{G}(\theta') > \mu_1\} \\
&\quad \times \phi\left(\frac{\bar{G}(\theta) - \mu_1}{\sigma(\theta)/\sqrt{n(\theta)}}\right) - \frac{\delta_1(\theta)}{\sigma(\theta)/\sqrt{n(\theta)}} = 0
\end{align*}$$

where $\delta_1(\theta) \equiv \bar{G}(\theta) - \mu_1$ for all $\theta \in \Theta$. Simplifying Equation (15), we have

$$\begin{align*}
\Pr\{\bar{G}(\theta) \leq \mu_1\} \phi\left(\frac{\delta_1(\theta)}{\sigma(\theta)/\sqrt{n(\theta)}}\right) - \frac{\delta_1(\theta)}{\sigma(\theta)/\sqrt{n(\theta)}} = 0,
\end{align*}$$

11

Note that the idea of APCSms is that PCSm is lower bounded by the probability that all observed good enough designs are truly good enough, which can be asymptotically maximized if we follow the allocation procedure in Chen et al. (2008). This explains why the following Theorem 1 leads to a similar allocation procedure as in Chen et al. (2008). We briefly provide the analysis as follows.

Let $L_m$ be the Lagrangian relaxation of AP1:

$$\begin{align*}
L_m \equiv APCSms + \lambda_m \left(\sum_{i=1}^k n(\theta_i) - T\right) \\
&= \Pr\{\bar{G} \subseteq \bar{G} \subseteq \bar{G}, G = \bar{G}\} \\
&\quad \times \Pr\{\bar{G} \subseteq \bar{G} \subseteq \bar{G}, G = \bar{G}\} \\
&\quad \times \Pr\{\bar{G} \subseteq \bar{G} \subseteq \bar{G}, G = \bar{G}\} \\
&\quad \geq \Pr\{\bar{G} \subseteq \bar{G} \subseteq \bar{G}, G = \bar{G}\}.
\end{align*}$$

Note that the idea of APCSms is that PCSm is lower bounded by the probability that all observed good enough designs are truly good enough, which can be asymptotically maximized if we follow the allocation procedure in Chen et al. (2008). This explains why the following Theorem 1 leads to a similar allocation procedure as in Chen et al. (2008). We briefly provide the analysis as follows.

Let $L_m$ be the Lagrangian relaxation of AP1:

$$\begin{align*}
L_m \equiv APCSms + \lambda_m \left(\sum_{i=1}^k n(\theta_i) - T\right) \\
&= \Pr\{\bar{G} \subseteq \bar{G} \subseteq \bar{G}, G = \bar{G}\} \\
&\quad \times \Pr\{\bar{G} \subseteq \bar{G} \subseteq \bar{G}, G = \bar{G}\} \\
&\quad \times \Pr\{\bar{G} \subseteq \bar{G} \subseteq \bar{G}, G = \bar{G}\} \\
&\quad \geq \Pr\{\bar{G} \subseteq \bar{G} \subseteq \bar{G}, G = \bar{G}\}.
\end{align*}$$

11
Taking the natural log of both sides, we have
\[
\log \Pr \{ \tilde{J}(\theta') \leq \mu_1 \} + \log \frac{1}{\sqrt{2\pi}} - \frac{\delta_1^2(\theta)}{2\sigma^2(\theta)} n(\theta) + \log \frac{\delta_1(\theta)}{\sigma(\theta)} - \frac{1}{2} \log n(\theta)
\]
\[
= \log \Pr \{ \tilde{J}(\theta) \leq \mu_1 \} + \log \frac{1}{\sqrt{2\pi}} - \frac{\delta_1^2(\theta')}{2\sigma^2(\theta')n(\theta')} + \log \frac{\delta_1(\theta')}{\sigma(\theta')} - \frac{1}{2} \log n(\theta'). \quad (17)
\]

Letting \( n(\theta) = \alpha(\theta) T \) and \( n(\theta') = \alpha(\theta') T \) and dividing both sides by \( T \), we have
\[
\frac{1}{T} \log \Pr \{ \tilde{J}(\theta') \leq \mu_1 \} + \frac{1}{T} \log \frac{1}{\sqrt{2\pi}} - \frac{\delta_1^2(\theta)}{2\sigma^2(\theta)} \alpha(\theta) + \frac{1}{T} \log \frac{\delta_1(\theta)}{\sigma(\theta)} - \frac{1}{2T} (\log \alpha(\theta) + \log T)
\]
\[
= \frac{1}{T} \log \Pr \{ \tilde{J}(\theta) \leq \mu_1 \} + \frac{1}{T} \log \frac{1}{\sqrt{2\pi}} - \frac{\delta_1^2(\theta')}{2\sigma^2(\theta')} \alpha(\theta') + \frac{1}{T} \log \frac{\delta_1(\theta')}{\sigma(\theta')} - \frac{1}{2T} (\log \alpha(\theta') + \log T). \quad (18)
\]

Letting \( T \to \infty \), we have
\[
\frac{\delta_1^2(\theta)}{\sigma^2(\theta)} \alpha(\theta) = \frac{\delta_1^2(\theta')}{\sigma^2(\theta')} \alpha(\theta'). \quad (19)
\]

Rearranging the terms, we have
\[
\frac{n(\theta)}{n(\theta')} = \frac{\alpha(\theta)}{\alpha(\theta')} = \left( \frac{\sigma(\theta)/\delta_1(\theta)}{\sigma(\theta')/\delta_1(\theta')} \right)^2. \quad (20)
\]

For the other choices of \( \theta \) and \( \theta' \), it is tedious but straightforward to show that we have
\[
\frac{n(\theta)}{n(\theta')} = \left( \frac{\sigma(\theta)/\delta_1(\theta)}{\sigma(\theta')/\delta_1(\theta')} \right)^2. \quad (21)
\]

Also note that Equation (14) requires that \( n(\theta) \geq 0 \) for all \( \theta \in \Theta \). Thus, Equation (20) provides an asymptotically optimal allocation for AP1. We summarize the above results into the following theorem.

**Theorem 1.** PCSm is asymptotically maximized when
\[
\frac{n(\theta)}{n(\theta')} = \left( \frac{\sigma(\theta)/\delta_1(\theta)}{\sigma(\theta')/\delta_1(\theta')} \right)^2,
\]
for all \( \theta, \theta' \in \Theta \).

Note that the difference between APCSm and PCSm depends on \( \mu_1 \), which is the boundary that separates the top-\( g \) designs from the rest of the designs. In order to minimize this difference we should pick \( \mu_1 \) such that APCSm is maximized. Following a similar analysis as in Chen et al. (2008), we have
\[
\mu_1 = \frac{\sigma(\theta_{i+1})/\tilde{J}(\theta_{i+1})/\sqrt{n(\theta_{i+1})} + \sigma(\theta_{i})/\tilde{J}(\theta_{i})/\sqrt{n(\theta_{i})}}{\sigma(\theta_{i+1})/\sqrt{n(\theta_{i+1})} + \sigma(\theta_{i+1})/\sqrt{n(\theta_{i+1})}}.
\]

Following Theorem 1, we have Algorithm 1.

**Algorithm 1** Optimal computing budget allocation for \( m \) simplest good designs in the ordinal sense (OCBAmSGO)

**Step 0.** Simulate each design by \( n_0 \) replications; \( l \leftarrow 0 \); \( n'(\theta_1) = n'(\theta_2) = \cdots = n'(\theta_k) = n_0 \).

**Step 1.** If \( \sum_{i=1}^k n(\theta_i) \geq T \), stop.

**Step 2.** Increase the total simulation time by \( \Delta \) and compute the new budget allocation \( n^{l+1}(\theta_1), \ldots, n^{l+1}(\theta_k) \) using Theorem 1.

**Step 3.** Simulate design \( i \) for additional \( \max(0, n^{l+1}(\theta_i) - n'(\theta_i)) \) time, \( i = 1, \ldots, k \); \( l \leftarrow l + 1 \). Go to Step 1.

### 3.2. Selecting a bSG

The choice of \( S \) in Equation (7) also provides an estimate of the bSG under the given observed performance. We can divide the entire design space into four subsets:

\[
S_1 = \bigcup_{i=1, \theta \neq \theta_{i+1}}^M \tilde{G}_i, \quad (23)
\]

\[
S_2 = \bigcup_{i=1}^M \tilde{D}_i, \quad (24)
\]

\[
S_3 = \{ \theta_{t+1,1}, \ldots, \theta_{t+1,m-\sum_{i=1}^t |\tilde{G}_i|} \}, \quad (25)
\]

\[
S_4 = \tilde{G}_{t+1} \setminus S_3. \quad (26)
\]

In other words, \( S_1 \) contains all of the observed good designs except those in \( \tilde{G}_{t+1} \); \( S_2 \) contains all of the observed bad designs, including those in \( \tilde{D}_{t+1} \); \( S_3 \) = \( S \cap \tilde{G}_{t+1} \) \( S_4 \) contains the designs in \( \tilde{G}_{t+1} \) other than \( S_3 \). Then we have

\[
\text{PCSb} \equiv \Pr \{ S \text{ is bSG} \}
\]

\[
\geq \Pr \{ S_1 \cup S_2 \cup S_3 \text{ is good}, S_1 \text{ is not good}, S_2 \text{ is better than } S_1 \}
\]

\[
\geq \Pr \{ \tilde{J}(\theta) \leq \mu_1, \theta \in S_1; \tilde{J}(\theta) > \mu_1, \theta \in S_2 \}
\]

\[
\geq \frac{\mu_2}{\mu_1 + \mu_2} \Pr \{ \tilde{J}(\theta) < \mu_1, \theta \in S_1; \tilde{J}(\theta) \geq \mu_1, \theta \in S_2 \}
\]

\[
= \left( \prod_{\theta \in S_1} \Pr \{ \tilde{J}(\theta) \leq \mu_1 \} \right) \left( \prod_{\theta \in S_2} \Pr \{ \tilde{J}(\theta) > \mu_1 \} \right)
\]

\[
\times \left( \prod_{\theta \in S_3} \Pr \{ \tilde{J}(\theta) \leq \mu_2 \} \right) \left( \prod_{\theta \in S_4} \Pr \{ \tilde{J}(\theta) < \mu_1 \} \right)
\]

\[
:= \text{APCSb}, \quad (27)
\]

where \( \mu_2 \) is the boundary that separates the designs in \( S_1 \) from the designs in \( S_2 \). The difference between APCSm and PCSb depends on \( \mu_1 \) and \( \mu_2 \). In order to minimize this difference we should pick \( \mu_1 \) and \( \mu_2 \) such that APCSm is maximized. Following a similar analysis as in Chen et al. (2008), we use Equation (22) to determine the value of \( \mu_1 \),
Efficient computing budget allocation

and we have

\[ \mu_2 = \min_{\theta} \left( \frac{\sigma(\theta_{i+1,r}) J(\theta_{i+1,r})/\sqrt{n(\theta_{i+1,r})}}{\sigma(\theta_{i+1,r})/\sqrt{n(\theta_{i+1,r})} + \sigma(\theta_{i+1,r}) J(\theta_{i+1,r})/\sqrt{n(\theta_{i+1,r})}} \right) \]

(28)

where \( r = m - \sum_{i=1}^{t} |G_i| \). Then an approximate version of P2 is

\[ \text{(AP2)} \ max_{n(\theta), \theta \in \Theta} APCSb \ \text{s.t.} \ \sum_{i=1}^{k} n(\theta_i) = T. \]

Let \( L_b \) be the Lagrangian relaxation of AP2.

\[ L_b \equiv APCSb + \lambda_b \left( \sum_{i=1}^{k} n(\theta_i) - T \right). \]

(29)

The KKT conditions of AP2 are as follows.

For \( \theta \in S_1 \):

\[ \frac{\partial L_b}{\partial n(\theta)} = (\Pi_{\theta' \in S \atop \theta' \neq \theta} \text{Pr}[\tilde{J}(\theta') \leq \mu_1]) \]

\[ \times (\Pi_{\theta' \in S \atop \theta' \neq \theta} \text{Pr}[\tilde{J}(\theta') > \mu_1]) \]

\[ \times (\Pi_{\theta' \in S \atop \theta' \neq \theta} \text{Pr}[\tilde{J}(\theta') \leq \mu_2]) \]

\[ \times (\Pi_{\theta' \in S \atop \theta' \neq \theta} \text{Pr}[\mu_2 < \tilde{J}(\theta') \leq \mu_1]) \phi \]

\[ \left( \frac{-\delta_1(\theta)}{\sigma(\theta)/\sqrt{n(\theta)}} \right) - \frac{\delta_1(\theta)}{2\sigma(\theta)/\sqrt{n(\theta)}} + \lambda_b = 0, \]

(30)

and

\[ \frac{\partial L_b}{\partial \lambda_b} = \sum_{i=1}^{k} n(\theta_i) - T = 0, \]

(34)

for \( \theta \in S_2 \):

\[ \frac{\partial L_b}{\partial n(\theta)} = (\Pi_{\theta' \in S \atop \theta' \neq \theta} \text{Pr}[\tilde{J}(\theta') \leq \mu_1]) \]

\[ \times (\Pi_{\theta' \in S \atop \theta' \neq \theta} \text{Pr}[\tilde{J}(\theta') > \mu_1]) \]

\[ \times (\Pi_{\theta' \in S \atop \theta' \neq \theta} \text{Pr}[\tilde{J}(\theta') \leq \mu_2]) \]

\[ \times (\Pi_{\theta' \in S \atop \theta' \neq \theta} \text{Pr}[\mu_2 < \tilde{J}(\theta') \leq \mu_1]) \phi \]

\[ \left( \frac{\delta_1(\theta)}{\sigma(\theta)/\sqrt{n(\theta)}} \right) - \frac{\delta_1(\theta)}{2\sigma(\theta)/\sqrt{n(\theta)}} + \lambda_b = 0; \]

(31)

for \( \theta \in S_3 \):

\[ \frac{\partial L_b}{\partial n(\theta)} = (\Pi_{\theta' \in S \atop \theta' \neq \theta} \text{Pr}[\tilde{J}(\theta') \leq \mu_1]) \]

\[ \times (\Pi_{\theta' \in S \atop \theta' \neq \theta} \text{Pr}[\tilde{J}(\theta') > \mu_1]) \]

\[ \times (\Pi_{\theta' \in S \atop \theta' \neq \theta} \text{Pr}[\tilde{J}(\theta') \leq \mu_2]) \]

\[ \times (\Pi_{\theta' \in S \atop \theta' \neq \theta} \text{Pr}[\mu_2 < \tilde{J}(\theta') \leq \mu_1]) \phi \]

\[ \left( \frac{-\delta_2(\theta)}{\sigma(\theta)/\sqrt{n(\theta)}} \right) + \lambda_b = 0. \]

(32)

Simplifying Equation (35), we have

\[ \text{Pr}[\tilde{J}(\theta) \leq \mu_1] \phi \left( \frac{-\delta_1(\theta)}{\sigma(\theta)/\sqrt{n(\theta)}} \right) - \frac{\delta_1(\theta)}{\sigma(\theta)/\sqrt{n(\theta)}} \]

\[ = \text{Pr}[\tilde{J}(\theta) \leq \mu_1] \phi \left( \frac{-\delta_1(\theta)}{\sigma(\theta)/\sqrt{n(\theta)}} \right) - \frac{\delta_1(\theta)}{\sigma(\theta)/\sqrt{n(\theta)}}. \]

(36)
Taking the natural log of both sides, we have

\[
\log \Pr \{ \tilde{J}(\theta') \leq \mu_1 \} + \log \frac{1}{\sqrt{2\pi}} - \frac{\delta_1^2(\theta)}{2\sigma^2(\theta)} n(\theta) \\
+ \log \frac{\delta_1(\theta)}{\sigma(\theta)} - \frac{1}{2} \log n(\theta) \\
= \log \Pr \{ \tilde{J}(\theta) \leq \mu_1 \} + \log \frac{1}{\sqrt{2\pi}} - \frac{\delta_1^2(\theta')}{2\sigma^2(\theta')} n(\theta') \\
+ \log \frac{\delta_1(\theta')}{\sigma(\theta')} - \frac{1}{2} \log n(\theta'). \tag{37}
\]

Let \(n(\theta) = \alpha(\theta)T\) and \(n(\theta') = \alpha(\theta')T\). Dividing both sides by \(T\), we have

\[
\frac{1}{T} \log \Pr \{ \tilde{J}(\theta') \leq \mu_1 \} + \frac{1}{T} \log \frac{1}{\sqrt{2\pi}} - \frac{\delta_1^2(\theta)}{2\sigma^2(\theta)} \alpha(\theta) \\
+ \frac{1}{T} \log \frac{\delta_1(\theta)}{\sigma(\theta)} - \frac{1}{2T} \log \alpha(\theta)T \\
= \frac{1}{T} \log \Pr \{ \tilde{J}(\theta) \leq \mu_1 \} + \frac{1}{T} \log \frac{1}{\sqrt{2\pi}} - \frac{\delta_1^2(\theta')}{2\sigma^2(\theta')} \alpha(\theta') \\
+ \frac{1}{T} \log \frac{\delta_1(\theta')}{\sigma(\theta')} - \frac{1}{2T} \log \alpha(\theta')T. \tag{38}
\]

Letting \(T \to \infty\), we have

\[
\frac{\delta_1^2(\theta)}{\sigma^2(\theta)} \alpha(\theta) = \frac{\delta_1^2(\theta')}{\sigma^2(\theta')} \alpha(\theta'). \tag{39}
\]

Rearranging the terms, we have

\[
\frac{n(\theta)}{n(\theta')} = \frac{\alpha(\theta)}{\alpha(\theta')} = \left( \frac{\sigma(\theta)/\delta_1(\theta)}{\sigma(\theta')/\delta_1(\theta')} \right)^2. \tag{40}
\]

Following similar analysis we have that for \(\theta, \theta' \in \mathcal{S}_1 \cup \mathcal{S}_2 \cup \mathcal{S}_3\):

\[
\frac{n(\theta)}{n(\theta')} = \left( \frac{\sigma(\theta)/\nu(\theta)}{\sigma(\theta')/\nu(\theta')} \right)^2, \tag{41}
\]

where

\[
\nu(\theta) = \begin{cases} 
\delta_1(\theta), & \theta \in \mathcal{S}_1 \cup \mathcal{S}_2, \\
\delta_2(\theta), & \theta \in \mathcal{S}_3.
\end{cases}
\]

Case 2: \(\theta \in \mathcal{S}_1, \theta' \in \mathcal{S}_4\). Following Equations (30) and (33), we have

\[
(\Pi_{\theta' \in \mathcal{S}_1, \theta' \neq \theta} \Pr \{ \tilde{J}(\theta'') \leq \mu_1 \}) (\Pi_{\theta' \in \mathcal{S}_4} \Pr \{ \tilde{J}(\theta'') > \mu_1 \}) \\
\times (\Pi_{\theta' \in \mathcal{S}_4} \Pr \{ \tilde{J}(\theta'') \leq \mu_1 \})
\]

Simplifying Equation (42), we have

\[
\Pr \{ \mu_2 < \tilde{J}(\theta') \leq \mu_1 \} \phi \left( \frac{-\delta_1(\theta)}{\sigma(\theta)/\sqrt{n(\theta)}} \right) \\
= \Pr \{ \tilde{J}(\theta) \leq \mu_1 \} \cdot A, \tag{43}
\]

where

\[
A = \phi \left( \frac{-\delta_1(\theta)}{\sigma(\theta)/\sqrt{n(\theta)}} \right) - \phi \left( \frac{-\delta_2(\theta)}{\sigma(\theta)/\sqrt{n(\theta)}} \right) \\
\times \frac{\delta_1(\theta)}{\sigma(\theta)/\sqrt{n(\theta)}}. \tag{44}
\]

Taking the natural log of both sides, we have

\[
\log \Pr \{ \mu_2 < \tilde{J}(\theta') \leq \mu_1 \} + \log \frac{1}{\sqrt{2\pi}} - \frac{\delta_1^2(\theta)}{2\sigma^2(\theta)} n(\theta) \\
+ \log \frac{\delta_1(\theta)}{\sigma(\theta)} - \frac{1}{2} \log n(\theta) = \log \Pr \{ \tilde{J}(\theta) \leq \mu_1 \} + \log A. \tag{45}
\]

Rearranging the terms, we have

\[
-\frac{\delta_1^2(\theta) \alpha(\theta)}{2\sigma^2(\theta)} = \lim_{T \to \infty} \frac{1}{T} \log A. \tag{46}
\]

Note that by L'Hôpital's rule we have

\[
\lim_{T \to \infty} \frac{1}{T} \log A = \lim_{T \to \infty} \frac{dA/dT}{A}. \tag{47}
\]
Efficient computing budget allocation

Combining Equations (46), (47), and (52), we have

\[
\frac{n(\theta)}{n(\theta')} = \begin{cases} 
\left(\frac{\sigma(\theta)/\delta_1(\theta)}{\sigma(\theta')/\delta_1(\theta')}\right)^2 & \text{if } \bar{J}(\theta') > \frac{\mu_1 + \mu_2}{2}, \\
\left(\frac{\sigma(\theta)/\delta_2(\theta)}{\sigma(\theta')/\delta_2(\theta')}\right)^2 & \text{if } \bar{J}(\theta') \leq \frac{\mu_1 + \mu_2}{2}.
\end{cases}
\]

(54)

Combining Equations (51) and (53), we have

\[
\frac{n(\theta)}{n(\theta')} = \begin{cases} 
\left(\frac{\sigma(\theta)/\delta_1(\theta)}{\sigma(\theta')/\delta_1(\theta')}\right)^2 & \text{if } \bar{J}(\theta') > \frac{\mu_1 + \mu_2}{2}, \\
\left(\frac{\sigma(\theta)/\delta_2(\theta)}{\sigma(\theta')/\delta_2(\theta')}\right)^2 & \text{if } \bar{J}(\theta') \leq \frac{\mu_1 + \mu_2}{2}.
\end{cases}
\]

(55)

Similarly, for \( \theta \in S_1 \cup S_2 \cup S_3 \) and \( \theta' \in S_4 \), we have

\[
\frac{n(\theta)}{n(\theta')} = \begin{cases} 
\left(\frac{\sigma(\theta)/\nu(\theta)}{\sigma(\theta')/\nu(\theta')}\right)^2 & \text{if } \bar{J}(\theta') > \frac{\mu_1 + \mu_2}{2}, \\
\left(\frac{\sigma(\theta)/\nu(\theta)}{\sigma(\theta')/\nu(\theta')}\right)^2 & \text{if } \bar{J}(\theta') \leq \frac{\mu_1 + \mu_2}{2}.
\end{cases}
\]

(56)

In other words, we can further split \( S_4 \) into two subsets:

\[
S_{41} = \left\{ \theta \in S_4 \text{ and } \bar{J}(\theta) > \frac{\mu_1 + \mu_2}{2} \right\},
\]

(56)

\[
S_{42} = \left\{ \theta \in S_4 \text{ and } \bar{J}(\theta) \leq \frac{\mu_1 + \mu_2}{2} \right\}.
\]

(57)

Combining Equations (41), (54), and (55) together, we have

\[
\frac{n(\theta)}{(\sigma(\theta)/\delta_1(\theta))^2}_{\theta \in S_1 \cup S_2 \cup S_4} = \frac{n(\theta)}{(\sigma(\theta)/\delta_2(\theta))^2}_{\theta \in S_3 \cup S_2}.
\]

(58)

Then we have the following theorem.

**Theorem 2.** PCSb is asymptotically maximized when

\[
\frac{n(\theta)}{(\sigma(\theta)/\delta_1(\theta))^2}_{\theta \in S_1 \cup S_2 \cup S_4} = \frac{n(\theta)}{(\sigma(\theta)/\delta_2(\theta))^2}_{\theta \in S_3 \cup S_2}.
\]

(59)

We then have Algorithm 2.

**Algorithm 2** Optimal computing budget allocation for the best \( m \) simplest good designs in the ordinal sense (OCBAb-SGO)

1. **Step 0.** Simulate each design by \( n_0 \) replications; \( l \leftarrow 0 \); \( n'(\theta_1) = n'(\theta_2) = \ldots = n'(\theta_k) = n_0 \).
2. **Step 1.** If \( \sum_{i=1}^{k} n(\theta_i) \geq T \), STOP.
3. **Step 2.** Increase the total simulation time by \( \Delta \) and compute the new budget allocation \( n'(\theta_1), \ldots, n'(\theta_k) \) using Theorem 2.
4. **Step 3.** Simulate design \( i \) for additional \( \max(0, n'(\theta_i) - n'(\theta_i)) \), \( i = 1, \ldots, k; l \leftarrow l + 1 \). Go to Step 1.

Note that though the above choices of \( S \) in mSG and bSG are the same, their allocations are clearly different. OCBAmSGO tries to make sure that \( \bar{G} \) are truly top-g. Then the choice of \( S \) in Equation (7) will make sure that the simplest \( m \) designs in \( \bar{G} \) are picked. OCBAbSGO tries to furthermore make sure that designs in \( S_3 \) are better than designs in \( S_4 \).
4. Numerical results

We compare OCBAmSGO (Algorithm 1, or A1 for short) and OCBAbSGO (Algorithm 2, or A2 for short) with the Equal Allocation (EA) approach over two groups of examples. The first group includes academic examples. The second group includes smoke detection problems in WSNs.

4.1. Academic examples

In order to capture different relationships between cardinal performance and ordinal indexes of designs, the following types of ordered performance are considered:

- neutral, \( J(\theta[i]) = i - 1, i = 1, \ldots, 10, \)
- flat, \( J(\theta[i]) = 9 - 3\sqrt{10 - i}, i = 1, \ldots, 10, \)
- steep, \( J(\theta[i]) = 9 - ((10 - i)/3)^2, i = 1, \ldots, 10, \)

where \( \theta[i] \) represents the top \( i \)th design. In the neutral type of problems, the performance difference between neighboring designs is equal. In the flat type of problems, most designs have a good performance. On the contrary, in the steep type of problems, most designs have poor performance. The following two types of relationships between performance and complexity are considered.

1. Simpler is better; i.e., if \( C(\theta) < C(\theta'), \) then \( J(\theta) < J(\theta'). \)
2. Simpler is worse; i.e., if \( C(\theta) < C(\theta'), \) then \( J(\theta) > J(\theta'). \)

When simpler is better, we have \( \theta[i] = \theta_i, i = 1, \ldots, 10. \) When simpler is worse, we have \( \theta[i] = \theta_{11-i}, i = 1, \ldots, 10. \)

Combining the above two considerations, we then have six types of problems. In each problem, \( \Theta_1 = \{\theta_1, \theta_2\}, \Theta_2 = \{\theta_3, \theta_4\}, \Theta_3 = \{\theta_5, \theta_6\}, \Theta_4 = \{\theta_7, \theta_8\}, \) and \( \Theta_5 = \{\theta_9, \theta_{10}\}. \) The performance of the six problems are shown in Figs. 1 and 2.

Fig. 1. The three examples where simpler is better (color figure provided online).

Fig. 2. The three examples where simpler is worse (color figure provided online).

Regard the top-3 designs as good; i.e., \( g = 3. \) We are interested in the \( m \)-simplest good designs, where \( m = 2. \) Suppose the observation noise for each design is independent and identically distributed Gaussian \( N(0, \sigma^2). \) We use OCBAmSGO, OCBAbSGO, and EA to find the \( m \)SG and \( b \)SG, respectively, where \( n_0 = 30 \) and \( \Delta = 10. \) Their PCSm and PCSb for different \( T \) values were estimated over 100,000 independent runs and shown in Figs. 3–8, respectively. We make the following remarks.

Remark 1. In all of the six problems, OCBAmSGO (Algorithm 1) achieves higher PCSm values than EA and
OCBAbSGO (Algorithm 2) achieves higher PCSb values than EA.

Remark 2. In all of the six problems, when $T$ is fixed, the PCSm that is achieved by OCBAmSGO is higher than the PCSb that is achieved by OCBAbSGO. Similarly, EA achieves higher PCSm values than PCSb values. This is consistent with the fact that a bSG must be an mSG but an mSG is not necessarily a bSG.

Remark 3. When $T$ increases, OCBAmSGO, OCBAbSGO, and EA all achieve higher PCS values. This is consistent with intuition because more computing budget should lead to higher PCS values.
Remark 4. For a given $T$, OCBAmSGO, OCBAbSGO, and EA achieve the highest PCS values in the steep problems, achieve lower PCS values in the neutral problems, and achieve the lowest PCS values in the flat problems. This is consistent with intuition because the performance differences among the good designs in the steep problems are larger than that in the neutral problems, which in turn are larger than that in the flat problems.

Remark 5. Note that in our academic examples where simpler is better PCSm only requires two of the actual top-3 designs to be within the observed top-3. But when simpler is worse PCSm requires all of the actual top-3 designs to be observed as top-3. Similarly, when simpler is better PCSb only requires the actual top-2 designs to be observed as top-2. However, when simpler is worse PCSb requires all of the actual top-3 designs to be observed as top-3 and furthermore requires the truly best design to be observed as better than the truly second-best design. This explains why the PCS’s in Figs. 3 to 5 are higher than the PCS’s in Figs. 6 to 8. Because the performance differences between the good and bad designs in the steep problems are larger than those in the neutral problems, which in turn are larger than those in the flat problems, the flat problem where simpler is worse (Fig. 7) has a significantly lower PCS than where simpler is better (Fig. 4). Note that though the PCS values are lower in problems where simpler is worse, they still converge to unity when there is an infinite computing budget. It will be an interesting future research topic to explore the knowledge of “simpler is worse” or “simpler is better” to develop a better allocation procedure. It will also be interesting to study how to change the complexities of the designs so that a higher PCS may be achieved under a given computing budget.

4.2. Smoke detection in WSNs

Consider a WSN with three nodes that are used to detect smoke in an Area of Interest (AoI). The AoI is discretized into $10 \times 10$ grids in Fig. 9. A fire may start at any point on the grid in the AoI with an equal probability. Once initiated, a smoke particle is generated at the fire source at any unit of time. An existing smoke particle randomly walks to a neighboring point along the grid with a positive probability. There are four possible directions in which to walk. Each direction is taken with some probability; that is,

\begin{align}
\Pr[x_{t+1} = x_t + 1, y_{t+1} = y_t] &\propto d((x_t + 1, y_t), (x_0, y_0)), \\
\Pr[x_{t+1} = x_t - 1, y_{t+1} = y_t] &\propto d((x_t - 1, y_t), (x_0, y_0)), \\
\Pr[x_{t+1} = x_t, y_{t+1} = y_t + 1] &\propto d((x_t, y_t + 1), (x_0, y_0)), \\
\Pr[x_{t+1} = x_t, y_{t+1} = y_t - 1] &\propto d((x_t, y_t - 1), (x_0, y_0)),
\end{align}

where $(x_t, y_t)$ is the position of the smoke particle at time $t$, $(x_0, y_0)$ is the position of the fire, and $d(\cdot, \cdot)$ is the distance between two positions. In other words, the fire is generating smoke with the specified probabilities. Once initiated, a smoke particle is generated at the fire source at any unit of time. An existing smoke particle randomly walks to a neighboring point along the grid with a positive probability. There are four possible directions in which to walk. Each direction is taken with some probability; that is,
smoke particle arrives at the sensor (this is purely passive detection); $r > 1$ means that the sensor detects the smoke if a smoke particle is within $r$ grids away from the sensor. Power consumption increases as the active sensing radius increases. Thus, we are interested in a good deployment with a short sensing radius. When we pick out $m$ simplest good deployments, it is up to the final user to determine which deployment to use. Usually only one deployment is eventually selected and implemented. There are 84 possible ways to deploy the three sensors; however, removing all of the symmetric possibilities leaves us with only 16 to analyze, as seen in Fig. 10. Let $r$ be the complexity measure. Thus, $|\Theta_i| = 16; i = 1, 2, 3, 4$; and $|\Theta| = 64$. For each deployment and sensing radius we are interested in the probability that a smoke particle is detected within time $T_0$, with $T_0 = 5$ in our numerical experiments. We evaluated the performance of the 64 designs over 10 000 independent simulations and the results are shown in Fig. 11 together

![Fig. 11. The performance of the 64 designs (color figure provided online).](image)

![Fig. 12. PCSm values for the EA and A1 ($m = 19, 20$) (color figure provided online).](image)

![Fig. 13. PCSb values for the of EA and A2 ($m = 19, 20$) (color figure provided online).](image)
Remark 9. \(PCS_b\) are not necessarily decreasing functions w.r.t. \(m\). We consider the computing budget allocation problem in several numerical examples. Though we assume Gaussian observation noise in this article, the numerical results indicate that both OCBAmSGO and OCBAbSGO have a good performance even when the observation noise is not Gaussian and when the computing budget is finite. Note that APCSm and APCSb are lower bounds for PCSm and PCSb, respectively. They are derived by considering only the case that the observed top-\(g\) designs are truly top-\(g\). It is possible for the choice of \(S\) in Equation (7) to be an mSG (or bSG) even though \(S \neq G\); i.e., when an observed top-g design is not truly top-g. Exploring this case may lead to tighter lower bounds and better allocation procedures, which is an important further research direction. Also, note that only a single objective function with no simulation-based constraints is considered in this article. It is an interesting future research topic to extend the work in this article to problems with multiple objective functions and simulation-based constraints. We hope this work brings more insight to finding simple and good designs in general.

Acknowledgements

The authors would like to thank Grover Laporte, the editor, and the anonymous reviewers for their constructive comments on previous versions of this article.

This work has been supported in part by National Natural Science Foundation of China under grants 60704008, 60736027, 61174072, 61223202, 91224008, and 90924001; the Specialized Research Fund for the Doctoral Program of Higher Education (under grants 20070003110); the Tsinghua National Laboratory for Information Science and Technology (TNLIST) Cross-Discipline Foundation; the National 111 International Collaboration Project (under grant B06002); the National Science Foundation under grants ECCS-0901543 and CMMI-1130273; the Air Force Office of Scientific Research under grant FA-9550-12-1-0250; the Department of Energy under Award DESC0002223; the NIH under grant 1R21DK088368–01; and the National Science Council of Taiwan under Award NSC-100-2218-E-002-027-MY3.

References


Efficient computing budget allocation


Biographies

Qing-Shan Jia received a B.E. degree in Automation in July 2002 and a Ph.D. in Control Science and Engineering in July 2006, both from Tsinghua University, Beijing, China. He is an Associate Professor at the Center for Intelligent and Networked Systems, Department of Automation, TNLIST, Tsinghua University, Beijing, China. He was a Visiting Scholar at Harvard University in 2006 and a Visiting Scholar at the Hong Kong University of Science and Technology in 2010, and is a Visiting Scholar at the Massachusetts Institute of Technology. His research interests include theories and applications of discrete event dynamic systems and simulation-based performance evaluation and optimization of complex systems. He is a senior member of IEEE and serves as a co-guest editor and an associate editor of Discrete Event Dynamic Systems: Theory and Applications, co-guest editor and area editor of IIE Transactions, and an associate editor of IEEE Transactions on Automation and Engineering.

Enlu Zhou received a B.S. degree with highest honors in Electrical Engineering from Zhejiang University, China, in 2004 and a Ph.D. in Electrical Engineering from the University of Maryland, College Park, in 2009. Since then she has been an Assistant Professor at the Industrial & Enterprise Systems Engineering Department at the University of Illinois Urbana-Champaign. Her research interests include Markov decision processes, stochastic control, and simulation optimization. She is a recipient of the Best Theoretical Paper award at the 2009 Winter Simulation Conference and the 2012 AFOSR Young Investigator award.

Chun-Hung Chen received his Ph.D. degree in Engineering Sciences from Harvard University in 1994. He is a Professor of Systems Engineering & Operations Research at George Mason University (GMU) and is also affiliated with National Taiwan University. He was an Assistant Professor of Systems Engineering at the University of Pennsylvania before joining GMU. Sponsored by the NSF, NIH, DOE, NASA, MDA, and
Jia et al., FAA, he has worked on the development of very efficient methodology for stochastic simulation optimization and its applications to air transportation systems, semiconductor manufacturing, health care, security networks, power grids, and missile defense systems. He received the “National Thousand Talents” Award from the central government of China in 2011, the Best Automation Paper Award from the 2003 IEEE International Conference on Robotics and Automation, the 1994 Eliahu I. Jury Award from Harvard University, and the 1992 MasPar Parallel Computer Challenge Award. He has served as Co-Editor of the Proceedings of the 2002 Winter Simulation Conference and Program Co-Chair for 2007 Informs Simulation Society Workshop. He has served as a Department Editor for IIE Transactions, Associate Editor of IEEE Transactions on Automatic Control, Area Editor of Journal of Simulation Modeling Practice and Theory, Associate Editor of International Journal of Simulation and Process Modeling, and Associate Editor of IEEE Conference on Automation Science and Engineering.