EFFICIENT SELECTION OF A SET OF GOOD ENOUGH DESIGNS WITH COMPLEXITY PREFERENCE

BY

SHEN YAN

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Industrial Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2011

Urbana, Illinois

Adviser:

Assistant Professor Enlu Zhou

ABSTRACT

This thesis briefly reviews the important methods involved in solving the best design selection problem in the discrete-event system simulation. The selection of one or several best designs is a common problem people meet in real situations. The research originally focused on the one best design selection problem, the two-stage procedure. Later, there was literature about multiple designs selection problems which are useful in the global optimization as well. Most recently, some researchers have studied the problem of selecting one simplest sufficiently good design applicable to the node activation rule in the wireless sensor networks. However, the problem for selecting several simplest good enough designs is still open for consideration. The second part of the thesis introduces two new algorithms for solving the selection problem related to the designs mentioned above. These two algorithms OCBA-mSG and OCBA-bSG allocate the simulation budget efficiently to identify a subset of m simplest and good enough designs among a total of K ($K > m$) designs. The numerical results show that both OCBA-mSG and OCBA-bSG outperform some other approaches on the test problems.

PREFACE

A preliminary version consisting only OCBA-mSG in Chapter 3 has been submitted to the 2010 Winter Simulation Conference as [1].

ACKNOWLEDGMENTS

I would like to thank my adviser, Prof. Enlu Zhou, for her support and advice on the thesis. I am grateful for my husband Yang and my parents who have showed their endless encouragement over years. This thesis is dedicated to them.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

OCBA	Optimal Computing Budget Allocation
OCBA-mSG	Optimal Computing Budget Allocation for *m* Simplest Good Enough Designs
OCBA-bSG	Optimal Computing Budget Allocation for the *best m* Simplest Good Enough Designs
R&S	Ranking and Selection Problem
ASA	Adaptive Sampling Algorithm
ETSS	Enhanced Two-Stage Selection Procedure
DES	Discrete Event System
APCS	Approximated Probability of Correct Selection
WSNs	Wireless Sensor Networks
LS	Levin Search
EA	Equal Allocation

CHAPTER 1

INTRODUCTION

This thesis studies the best design selection problems occurring in the discrete-event system (DES) simulations. The discrete event systems include lots of man-made systems such as transportation networks, communication networks and manufacturing workshops. Because these systems follow both physical laws and man-made rules, it is difficult to find out an analytical way to solve these problems directly. Therefore, discrete event system simulation becomes the most important way to study the performance of DES, such as finding the best designs in the selection problems that are discussed in this thesis. The obstacle to the problem is that the simulation for these man-made systems is usually complex and time-consuming due to their complicated man-made rules and their large search space. In order to save simulation budget and be effective in the simulation process, it is our interest to find ways to smartly allocate the simulation budget for selecting the best designs. In this thesis, the word "design" is used to refer to the object under consideration, such as the ordering policy and the node activation rule in Section 3.1.

The best designs selection problem can be formulated in different ways. Chapter 2 introduces the problems related to selecting one or multiple best designs from the total of K designs, and selecting one simplest good enough designs from the total of K designs. Based on these methods, we develop new algorithms for selecting multiple simplest and good enough designs in an efficient way in Chapter 3. As a general introduction to the best designs selection problems, we briefly discuss the relevant leading research in this area in the following.

In the past decades, the selection of the best design problem is closely related to many known results in the literature on ranking and selection (R&S). In 1958, Gupta and Sobel [4] considered the problem of selecting a random size subset from k experimental categories in order to find all the populations that are good enough with a given probability. Later Gupta [5] proposed the method of

selecting a random size subset containing the best population with a given probability P of correct selection. In Gupta's method, the size of the subset is unknown beforehand and the size can be as high as $P * k$, where k is the number of parameters for the population. Thus, it is difficult to control the size of the subset and if the resulting subset is very large, the method will be less effective for researchers since the best design still lies in a big set. In 1975, Santner [6] further extended Gupta's method by imposing a maximum size m on the selected subset that containing the best design, which helped to control the size of the random selected subset.

An important statistical method for selecting the best design was developed in 1975 by Dudewicz and Dalal [7]. They first introduced the so-called two-stage procedure for simulation budget allocation to all the designs (refer to Section 2.1). The procedure for the first stage is to equally assign the initial simulation budget to every design, and then use the sample variance to determine the simulation budget allocation at the second stage. Since the simulation budget is always limited, by efficiently allocating the simulation budget to different designs, researchers get a high probability for the correct selection of the best design. In 1978, Rinott [8] found another method to allocate the simulation budget at the second stage. In 1985, Koenig and Law [9] modified the two-stage procedure for selecting a set of z designs which contains the top m designs with best performance from a total of K designs ($1 \leqslant m \leqslant z < K$), following the results in Dudewicz and Dalal [**?**]. This method can either select the best design when $m = z = 1$ or select the top m designs when $m = z > 1$. Similarly, the variance of the design is assumed to be unknown. The modification of two-stage method by Koenig and Law extended the application of the method to a wider area, since in many real problems we need not only one best design but several best designs. In 1997, Chick [10] solved the best design selection problem by using a Bayesian decision-theoretic approach. Since Chick used the Bayesian approach in his work, he is able to work out the best design out of multiple designs with both independent replications and dependent replications, and both Gaussian simulation and non-Gaussian simulation work.

Another important method for selecting the best designs called optimal computing budget allocation (OCBA) was developed by Chen, who has been studied the efficient simulation budget allocation for several years. In 1995, Chen [11] found a way to smartly allocate the computing budget to discrete event simulation. The numerical results show that the method introduced by Chen can

further reduce the computing time compared to the ordinal optimization which selects and allocates the most computing budget to a subset of designs that has high probability to be the best design. In 1996, Chen [12] used the so called steepest-ascent method to solve the discrete event optimal selection problem. However, he mentioned that this method may be further enhanced by finding a way to reduce the computing cost occurred every iteration for searching a solution to the budget allocation problem. Later on, Chen et al. [13] applied the greedy heuristic to the discrete event simulation. But the problem is that the budget allocation determined by this greedy heuristic may not be the optimized simulation budget allocation ways since it allocated the budget to only one design at each iteration. The method OCBA was introduced in 2000 by Chen, Lin, Yücesan and Chick [14] for further increasing the efficiency of ordinal optimization for selecting one best design. They used the probability of correct selection of the best design as the objective function and finally solved it by applying an asymptotic limit to the objective function which helps to find an upper bound for the objective function. In the numerical experiments, they compared the OCBA with the other popular methods such as two-stage procedure. The comparison shows that OCBA is much faster than the other methods, and the larger the number of total designs is, the more efficient the OCBA becomes. They also mentioned that although they used the asymptotic limit to derive the simulation budget allocation, the numerical experiments showed that the performance is excellent under limited simulation budget. Later on, Chen and Yücesan [15] solved a similar simulation budget allocation problem by changing the objective function to minimize the total number of simulation budget in order to achieve a desired probability of correct selection for one best design. They still used the asymptotic limit to find the optimized solution and did several numerical testings on this method which outperformed in the testings considered in their paper. In 2008, Chen, He, Fu and Lee [16] extended OCBA to the selection of the m best designs. To simplify the problem, they approximated the objective function $P(CS)$ (probability of correct selection of m designs) by independence properties which was also used in my thesis. Their paper provided a new way to select m best designs with efficient computing budget allocation among all the designs. Most recently, Pujowidian, Lee, Chen and Yap [17] developed OCBA further for the problem of selecting one single best design under multiple constraints of secondary performance measures. In a word, the OCBA method introduced by Chen et al. is noticeable in the area of

selecting either one or *m* best designs in recent years just as what Branke et al. [18] had discussed and compared in his paper with several other R&S procedures which showed that OCBA is in fact very effective since it allocates the most simulation budget to the critical designs which are sensitive in the correct selection process and smartly allocates less simulation budget to the uncritical designs which do not affect the selection results. More detailed algorithms for OCBA are given in Section 2.2.

There is also research focusing on optimizing the primary performance measure subject to the feasibility of a secondary performance measure. Andradóttir, Goldsman, Schmeiser, Schruben and Yücesan [19] proposed a two-phase approach which identifies all the feasible systems at the first stage, and then selects the best from them at the second stage. Recently, Batur and Kim [20] further modified the two-phase approach by finding a feasible set or near-feasible set during the first stage. On the other hand, Glynn and Juneja [21] used the large deviation theory to find the best way to allocate the simulation budget. By the way, they also considered the case of non-normal distribution and its effect. Later in 2008, Szechtman and Yücesan [22] used large deviation theory to deal with feasibility determination in a stochastic problem by the performance estimation through Monte Carlo simulation.

The problem of considering both complexity and performance evaluation has only been studied recently. The motivation for considering the complexity is from the real world, where the simple designs with low complexity are always preferred if their performance is good enough. The most relevant one is to solve the problem of selecting one simplest sufficiently good design by Jia [23], who proposed an Adaptive Sampling Algorithm (ASA) to minimize the Type II error of the chosen simplest good enough design (refer to Section 2.3). ASA determines which design to simulate in the next iteration with the goal of minimizing the type II error. In his paper, he compared ASA with Equal Allocation and Levin Search methods, and he also applied ASA to node activation policy problem in the wireless sensor networks where ASA performed well in selecting one simplest good enough design. However, ASA is not suitable for extension to select multiple designs, because it screens the designs sequentially one by one until the goal is achieved. Therefore, the problem still remains open on how to select multiple designs that are simplest and good enough. This problem will be addressed in Chapter 3.

The rest of the thesis is organized as follows. In Chapter 2, we introduce some

major methods for solving the best design selection problem, including the two-stage procedure, optimal computing budget allocation and adaptive sample algorithm, which are closely related to my research in Chapter 3. In Chapter 3, we propose our new methods for solving the problem of selecting multiple simplest good enough designs. We present the motivation for considering this simplest good enough problem, and briefly introduce our new methods in Section 3.1. In Section 3.3, we describe the problem formulation for selecting m simplest good enough designs and introduce the OCBA-mSG algorithm, abbreviated for optimal computing budget allocation for m simplest good enough designs, which aimed to minimize the probability of correct selection. Based on OCBA-mSG, we develop another slightly different algorithm called OCBA-bSG for selecting the best m simplest good enough designs in Section 3.4. OCBA-bSG selects the designs with the best performance from all the simplest good enough designs, with a slightly increase of simulation budget than OCBA-mSG. Under some specific situations when the simulation budget increasing are little, we may prefer OCBA-bSG to OCBA-mSG to get a more accurate selection results for the best m simplest good enough designs. In the next Section 3.6, we introduce the formula to calculate the upper bound of the simulation budget allocation for every design, and study the effect of this upper bound to our algorithms by comparing it with the other fixed upper bounds. The results imply that our dynamic upper bound performs well in general, although it is not always the best, the overall performance is satisfied. The numerical results for new algorithms are given in Section 3.7 in which we conduct four experiments on the new methods and compare their performance with the other two methods. Numerical results indicate that both OCBA-mSG and OCBA-bSG allocate the simulation budget efficiently to achieve a high probability of correct selection. In the end, I conclude the thesis in Chapter 4.

CHAPTER 2

BACKGROUND

In this chapter, we present several major methods for the best design selection problems which are closely related to our new algorithms in Chapter 3, and discuss their advantages and drawbacks as well. All the methods here concentrated on how to allocate simulation budget efficiently in order to control the probability that the selected designs really are the best ones or to control its type II error. The two-stage procedure in Section 2.1 is an important statistical method, which was proposed in 1975, to select one best design from K designs. The basic two-stage procedure only considered the sample variance of the designs in the simulation budget allocation process which helped us to find more accurate estimation for the performance of every design. Unfortunately, it ignored the effect of sample mean which could reduce the unnecessary simulation budget allocation to less important designs. Later on, Chen and Kelton [24] modified the two-stage procedure by using the information of both sample mean and sample variance which enhanced the efficiency of the two-stage procedure. In Section 2.2, we introduce the optimal computing budget allocation (OCBA) procedures for selecting single best design and multiple best designs with the goal to maximize the probability of correct selection. Our new methods in Chapter 3 are based on OCBA procedure to solve a more complicated problem. In Section 2.3, we introduce the adaptive sampling algorithm (ASA) [23] to solve the selection problem for the one simplest good enough design. Since ASA is not suitable for selecting multiple simplest good enough designs, we develop new algorithms in Chapter 3 based on OCBA introduced in Section 2.2 to solve this multiple selection problem.

Before we started, we would like to introduce some important notations:
θ_i: the notation for the design i.
Θ: the set of all the K designs including $\theta_1, \theta_2, \ldots, \theta_K$.
$J(\theta_i)$ or J_i: the performance measure for design θ_i. For simplicity of the notation, we write $J(\theta_i)$ as J_i in the following.

σ_i^2: variance of design θ_i. Since the true variance is unknown, we use the sample variance to estimate it.
N_i: simulation budget for design θ_i.
X_i^k: the k^{th} simulation replication for design θ_i, $k = 1, \ldots, N_i$. We assume that X_i^k's are independent inside the design (i.e., with respect to k) and across the designs (i.e., with respect to i).
$\bar{J}_i$: sample mean of design θ_i, $\bar{J}_i = (1/N_i) \sum_{k=1}^{N_i} X_i^k$.
$P(CS)$: the probability of the selected best designs based on the current samples are the true best designs.

2.1 Two-Stage Procedure

2.1.1 Basic Two-Stage Procedure

Consider selecting one best design from a set of K designs. Suppose the design with smaller performance measure J_i is regarded as having better performance. We order the designs according to their sample performance measure $\bar{J}_i$ as $\bar{J}^1 \leqslant \bar{J}^2 \leqslant \ldots \leqslant \bar{J}^K$, and the best design we want to select is the one with the smallest sample performance measure $\bar{J}^1$. The indifference zone $d^*(d^* > 0)$ is given by the user to detect the smallest actual difference that they are interested in. Difference less than d^* is considered as not important, which means if the difference between $\bar{J}^1$ and $\bar{J}^2$ is less than d^*, $\bar{J}^2$ can also be regarded as the correct selection. Since the smallest difference between the best design and all the other designs are $\bar{J}^2 - \bar{J}^1$, we impose the constraint that $\bar{J}^2 - \bar{J}^1 \geqslant d^*$. In order to select the true best design, we want to find a procedure that the probability of correct selection satisfies $P(CS) \geqslant P^*$ and $\bar{J}^2 - \bar{J}^1 \geqslant d^*$, where P^* is the required probability of correct selection.

One important statistical procedure to solve this problem was introduced by [7]. This is a Two-Stage Procedure. The first stage is to do n_0 initialization sampling for all the K designs and calculate their sample means and variances

based on the current samples as

$$\bar{J}_i(n_0) = (1/n_0)\sum_{k=1}^{n_0} X_i^k,$$

$$S_i^2(n_0) = \frac{\sum_{k=1}^{n_0}[\bar{J}_i(n_0) - X_i^k]^2}{n_0 - 1},$$

for $i = 1, 2, \ldots, K$. At the second stage, allocate the simulation budget N_i to every design by

$$N_i = max\left\{n_0 + 1, \lceil \frac{h_i^2 S_i^2(n_0)}{(d^*)^2} \rceil\right\}, \tag{2.1}$$

where $\lceil x \rceil$ is the smallest interger that is greater than or equal to the real number x, and h_i is a given constant that depends on k, P^*, and n_0. Then compute the sample mean for design i based on all the N_i samples and select the design with the smallest sample mean. In equation (2.1), the higher the sample variance $S_i^2(n_0)$ for design i, the more simulation budget N_i will be allocated to that design, which means they could ensure an accurate estimation for the performance of all the designs. However, this two-stage procedure only considered the sample variance but not consider the sample mean in the simulation budget allocation procedure. As a result, it is not suitable for a large number of designs, since the simulation budget are allocated to every design to ensure that they all get an accurate enough estimation for performance which is not necessary. Therefore, this method can be further enhanced by taking the sample mean into consideration.

The two-stage procedure introduced here is an important statistical method for selecting the single best design aimed to control the probability of correct selection. Research has been done to modify this two-stage procedure as described in Chapter 1. We introduce the two-stage procedure here, because they considered the sample variance in the simulation budget allocation process, which we also considered in our new algorithms in Chapter 3. Based on the two-stage procedure, I started looking for other related literatures, such as OCBA, and finally proposed the new algorithms for solving the simplest good enough designs selection problems using the information of both sample mean and sample variance.

2.1.2 Enhanced Two-Stage Selection (ETSS) Procedure

An enhanced two-stage selection procedure called ETSS for selecting one best design was proposed by Chen and Kelton [24] who modified the previous two-stage procedure (equation 2.1) by considering the sample mean in the second stage. During the first stage they still do the initial simulation n_0 for every design. But at the second stage, they introduce the indifference zone d_i instead of using the user provided d^* directly.

$$d_i = max(d^*, \bar{J}_i(n_0) - \bar{J}_b(n_0)), \tag{2.2}$$

where design b is the selected best design based on the current samples. Then, N_i is calculated by using the new d_i,

$$N_i = max\left\{n_0 + 1, \lceil \frac{h_i^2 S_i^2(n_0)}{(d_i)^2} \rceil \right\}, \tag{2.3}$$

for $i = 1, 2, \ldots, K$. With the introducing of d_i in Equation (2.2), the designs far from the selected best design will be allocated less simulation budget, and more simulation budget will be allocated to designs close to the selected best design based on the current samples. This indicates that not every design is treated equally based on their sample variance to get simulation budget N_i, but they also consider the effect of sample mean to determine which design is more important in the selection process. This result is consistent with OCBA procedure in Section 2.2.1. Actually, Equation (2.3) is the same as Equation (2.4) in OCBA procedure [14] when $N_i \geqslant n_0 + 1$ and $\bar{J}_i(n_0) - \bar{J}_b(n_0) \geqslant d^*$.

2.2 Optimal Computing Budget Allocation (OCBA)

We introduce OCBA algorithm here, because we use a similar problem formulation in Chapter 3 to solve the simplest good enough design selection problem, including the idea of Probability of Correct Selection, Approximate Probability of Correct Selection and the assumption on the prior distribution of the performance for the design.

2.2.1 OCBA for Selecting One Best Design

Chen et al. [14] introduced the OCBA approach to smartly allocate the simulation budget to find the one best design by considering both the sample variance and the sample mean during the simulation process. Here, we use b to denote the best design we selected, although design b may not be the true best design. The problem becomes to find the best simulation budget allocation N_i for all the K designs in order to maximize the probability for design b to be the true best design ($P(CS)$) with a given total simulation budget T. The problem was solved by introducing an Approximate Probability of Correct Selection ($APCS$) as a lower bound for the probability of correct selection $P(CS)$. And then by using the asymptotic limit to maximize $APCS$, we obtain the following simulation budget allocation rule:

$$\frac{N_i}{N_j} = (\frac{\sigma_i/\delta_{b,i}}{\sigma_j/\delta_{b,j}})^2,\ i,j \in \{1,2,\ldots,K\}, and\ i \neq j \neq b, \tag{2.4}$$

$$N_b = \sigma_b \sqrt{\sum_{i=1,i\neq b}^{K} \frac{N_i^2}{\sigma_i^2}}, \tag{2.5}$$

where $\delta_{b,i} = \bar{J}_b - \bar{J}_i$, $\bar{J}_b \leqslant \min_i \bar{J}_i$, and σ^2 is estimated by the sample variance.

From the numerical experiments given in [14], the method is proved to be highly efficient for selecting the best design out of K designs. And this OCBA procedure can smartly allocate the simulation budget to the critical designs (i.e. the designs which are important in determining the best design) by considering both the current sample mean and sample variance. Later, Chen et.al [15] solved the one best selection problem in an alternative way by considering the problem of allocating the least total simulation budget to achieve a desired $P(CS)$.

2.2.2 $OCBA_m$ for Selecting m Best Designs

The problem of selecting m best designs has been studied by Chen et al. [16] recently. Define the optimal subset S_m as the subset containing the m designs with the smallest sample means based on the current simulation results. The event that the designs in S_m are the true best m designs is called the correct

selection (CS). Then the probability of correct selection can be given by

$$P(CS) = P\{\tilde{J}_i \leqslant \tilde{J}_j, \forall i \in S_m \text{ and } \forall j \notin S_m\}. \tag{2.6}$$

The problem is to find the best way to allocate simulation budget among N_i with limited total simulation budget T to maximize the probability of correct selection:

$$\begin{aligned} \max_{N_1,N_2,\ldots,N_K} \quad & P(CS) \\ \text{s.t.} \quad & N_1 + N_2 + \ldots + N_K = T. \end{aligned} \tag{2.7}$$

To solve the above optimization problem, they introduced the following Approximated Probability of Correct Selection ($APCS_m$) as a lower bound for $P(CS)$ by using the posterior distribution $\tilde{J}$ (Section 3.2) and the independent property among designs.

$$P(CS) \geqslant APCS_m = \prod_{i \in \hat{S}_m} P\{\tilde{J}_i < c\} \prod_{i \notin \hat{S}_m} P\{\tilde{J}_i \geqslant c\},$$

where c is a constant. By maximizing $APCS_m$ asymptotically as $T \to \infty$, they get the allocation rule for N_i as

$$\frac{N_i}{N_j} = (\frac{\sigma_i/\delta_i}{\sigma_j/\delta_j})^2, \; i, j \in \{1, 2, \ldots, K\}, \tag{2.8}$$

where $\delta_i = \bar{J}_i - c$, $i = 1, 2, \ldots, K$. This approach $OCBA_m$ for selecting a subset of optimal designs instead of only one best designs is useful in global optimization, where it needs to select a subset of good candidate designs in each iteration of the algorithm. In the following chapter, we proposed two new methods for solving a more complicated problem based on the OCBA introduced here and extended it to the selection of the m simplest good enough designs.

2.3 Adaptive Sampling Algorithm (ASA)

The problem considered here related closely to our works in Chapter 3. This simplest good enough selection problem has only been considered very recently. The motivation for considering this problem is that the simple designs are

perferred to the complex designs when both of them have similar good performance in real world. Adaptive sampling algorithm (ASA) was proposed by Jia [23], who defined the problem as minimizing the complexity of the design (i.e. find the simplest design) in the good enough set (designs with good enough performance):

$$\min_{\theta \in G} C(\theta), \tag{2.9}$$

where $G = \{\theta | J(\theta) < J_0, \theta \in \Theta\}$ is the good enough set, $C(\theta)$ is the complexity of design θ, and J_0 is the good enough threshold given by user. He analyzed in his paper that type II error only occurs under the following three situations: never simulates any simplest good enough designs; all the simplest good enough designs that are simulated are infeasible (classified as not good enough designs); and the selected simplest good enough designs are not truely good enough. Jia estimated the type II error with an upper bound which is composed of the three possibilities stated above. The upper bound for type II error is given as follows: If $\sum_{i=1}^{C(\theta_0)-1} |\Theta_i| > \sum_{i=1}^{C(\theta_0)-1} |N_i|$, then the upper bound is

$$\max\{\Phi(\frac{-\varepsilon}{\sigma(\theta_0)/\sqrt{n(\theta_0)}}), \frac{\begin{pmatrix} \sum_{i=1}^{C(\theta_0)-1} |\Theta_i| - 1 \\ \sum_{i=1}^{C(\theta_0)-1} |N_i| \end{pmatrix}}{\begin{pmatrix} \sum_{i=1}^{C(\theta_0)-1} |\Theta_i| \\ \sum_{i=1}^{C(\theta_0)-1} |N_i| \end{pmatrix}} + 1 - \Phi(\frac{\varepsilon}{\sigma(\theta^*)/\sqrt{n(\theta^*)}})\};$$

If $\sum_{i=1}^{C(\theta_0)-1} |\Theta_i| = \sum_{i=1}^{C(\theta_0)-1} |N_i|$, then the upper bound is

$$\max\{\Phi(\frac{-\varepsilon}{\sigma(\theta_0)/\sqrt{n(\theta_0)}}), 1 - \Phi(\frac{\varepsilon}{\sigma(\theta^*)/\sqrt{n(\theta^*)}})\},$$

where $\theta^* \equiv argmax_{\theta \in \cup_{i=1}^{C(\theta_0)-1} S_i} (\sigma(\theta)/\sqrt{n(\theta)})$. In ASA, Jia simulated the design that can obtain the smallest upper bound in every iteration. In the end, he found out the simplest good enough design with the minimun type II error. However, ASA is not suitable for selecting multiple simplest good enough designs, because it screens the designs sequentially one by one until the goal is achieved. Therefore, the problem still remains open on how to select multiple designs that are simplest and good enough. This problem will be solved by our new methods introduced in the next chapter.

CHAPTER 3

NEW ALGORITHMS FOR SELECTING SIMPLEST GOOD ENOUGH DESIGNS

3.1 Introduction and Motivation

This chapter proposes our new methods OCBA-mSG and OCBA-bSG for solving the problem of selecting the m simplest good enough designs from a total of K designs aimed to maximize the probability of correct selection. The motivation for considering the selection of simplest good enough designs comes from the real world, where simple designs (designs with low descriptive complexity) are preferred to the complex ones (designs with high descriptive complexity) if the simple designs are good enough to satisfy our requirements. We prefer simple designs because they have advantages such as requiring less computing and memory resources, easier to interpret and to implement, and less expensive to implement. As a result, in real world, users usually prefer simple designs when they are good enough or with acceptable performance.

The motivation for considering the selection problem of the simplest good enough designs are from its applications in industry and daily life. A typical example is to find an ordering policy in inventory control. We may consider the problem of ordering a certain amount of products at each period to meet a stochastic demand which follows a probability distribution. In order to minimize the expected cost (including holding cost for excess inventory and shortage cost for unfilled demand), we want to determine the optimal ordering policy in each period. The optimal policy can be found analytically for some models to have the structure of a base-stock policy or an (s,S) policy [25, 26, 27, 28]. The base-stock policy or (s,S) policy is a threshold function that maps the current stock into the ordering amount. Motivated by this simple structure of the optimal policy for some models, we can select the best policies from all the available threshold policies for other more complex inventory models. In general, we can approximate the optimal ordering policy better with more number of thresholds

given the right values of these thresholds. Then the problem becomes to decide how many thresholds should we choose and what their values are. It is clear that with more thresholds in the function we have a more complex ordering policy, which is harder to determine the optimal values of these thresholds and to implement in practice. Conversely, with a small number of thresholds, such as one (base-stock policy) or two ((s,S) policy), we have a simple ordering policy, which is easier to compute and to implement. If the simple ordering policy can achieve a required cost criterion which is regarded as the good enough performance constraint in the following chapter, it will be more preferable than a complex ordering policy, even though the complex ordering policy may yield a lower cost.

Figure 3.1: Inventory Control Problem [2].

Another example is the design of node activation rules in the wireless sensor networks (WSNs), as described in [29, 23]. It is a problem about optimization of the node activation rule in WSNs. Let us consider a solar power WSNs that monitors an interested place. Each node needs to collaborate with its neighbors in order to have enough power to monitor an area of interest. There are three stages for every node: activation, readiness and sleep. When the node is out of power, it is at the sleep stage and cannot be activated. When the node has power, then it is ready and can be activated. Also during each time interval, every node has chance to be recharged no matter what stage it is at now, and if and only if the node is activated then it has the chance to loss power. And only the node that are being activated can be used to monitor the interested area and detect the event. The problem is to decide which node to active and when to activate such that we can get the highest probability of correct detection for random event. It is clear that a larger communication radius (e.g. more nodes are under communication) gives us a more complex node activation rule and consumes

more power. Conversely, a smaller communication radius gives us a simpler node activation rule and consumes less power. Similarly, given that the required probability of correct detection can be achieved, we prefer small communication radius of each node (i.e., simple activation rule) to large radius (i.e., complex activation rule). In other words, the simple designs are preferred to complex designs when they are both adequately good.

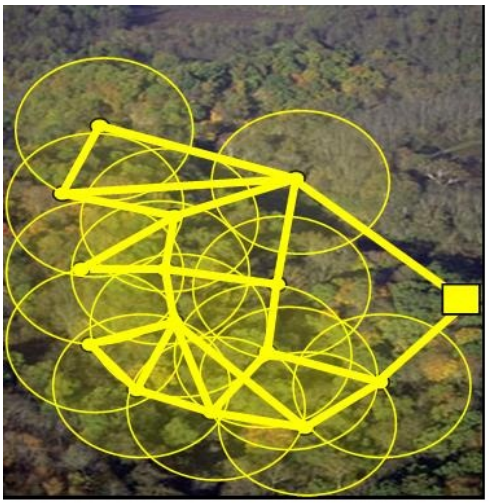

Figure 3.2: Wireless Sensor Network Problem [3].

In this chapter, the descriptive complexity of a design is represented by an nonnegative integer number, where simpler design has a smaller integer number. Since sometimes we may need to select more than one design in practice for robustness, we consider the general problem of selecting m ($m \geq 1$) designs that are simplest (with smallest complexity) and good enough (satisfying a constraint on the performance measure). The complexity of a design is a deterministic value, and we know the value once we simulate that design and the value of the complexity will remain the same afterwards. However, the performance of a design is subject to system noise, and hence, it can only be estimated from simulation, and the simulation is computationally expensive. For example, it takes a significant amount of computational effort to simulate the inventory system in order to evaluate the cost of a particular ordering policy due to the system noise. Here, the performance is measured by the sample mean, and the smaller the sample mean, the better the performance is. Hence, our goal is to find a way to allocate a given simulation budget efficiently to the designs so as to maximize the probability of correctly selecting the m simplest good enough

designs out of a total of K designs.

By using simulation to estimate the performance of a design, the simulation budget T cannot reach infinity in reality. Although we can use super computer to increase the computing ability and the simulation budget, it cannot be guarantted that the estimated performance for every design is accurate enough since the number of designs K under consideration can also be increased heavily and the simulation budget for every design which is T/K can even decrease under equal allocation method. In order to get a high propability of correct selection for the m simplest good enough designs, we need to find an effecient simulation budget allocation procedure for designs instead of equally allocating the budget to every design. To be more precise, we consider allocating the larger part of the limited amount of simulation budget to the designs which are critical or sensative in the process of the selection of the m simplest good enough designs, and allocating little simulation budget to all the other non-critical designs in the selection process. Folllowing this clear idea, we ensure that the limited simulation budget can be allocated efficiently to all the K designs, especially when the value of K is extremely large.

The chapter is organized as follows. In Section 3.2, we introduce the assumption we made for the prior distribution and the result we got for the posterior distribution which are used in the remaining chapter. In Section 3.3, we use the above allocation idea to discuss how to efficiently allocate the limited simulation budget to select the m simplest good enough designs by introducing a new algorithm OCBA-mSG. In Section 3.4, we concentrate on a slightly differencet problem of selecting best m simplest good enough designs and introduce the algorithm OCBA-bSG. The allocation procedures for both OCBA-mSG and OCBA-bSG are given in Section 3.5. After that in Section 3.6, we test the dynamic upper bound introduced in Section 3.5 with fixed upper bounds, and the results imply that the performance of our dynamic upper bound is acceptable in most experiments. Finally, we demonstrate our methods OCBA-mSG and OCBA-bSG, and compare them with other two methods Equal Allocation and Levin Search in Section 3.7. The experimental results show that our methods perform well especially under the large scale problem and when the uncertainty of the system is serious.

3.2 Preliminary

Here, we introduce the assumption we made for the prior distribution and the result we got for the posterior distribution for the performance of the design which will be used in the following chapter. Consider the performance of a general simulation problem for a complex discrete event design θ_k ($\theta_k \in \Theta$ and $|\Theta| = K$) defined as

$$J_k = E[L(\theta_k, \zeta)],$$

where ζ is a random vector for the uncertainty in the system, and J_k is the performance of the design which is the expectation of $L(\theta_k, \zeta)$. Since it is a complex system, $L(\theta_k, \zeta)$ is hard to calculate through small simulations. Therefore we estimate $E[L(\theta_k, \zeta)]$ by the sample mean of the performance as follows:

$$\bar{J}_k = (1/N_k) \sum_{i=1}^{N_k} [L(\theta_k, \zeta_{ki})],$$

where N_k is the number of samples for design θ_k. To construct the posterior distribution of J, we assume that there is no information before simulation for J_k and J_k has a prior distribution $N(0, \nu^2)$ with ν^2 extremely large. We also assume that the sample $\hat{J}_k$ for design k is normally distributed as $N(J_i, \sigma_i^2)$, then the posterior distribution is still a normal distribution [12]:

$$\tilde{J}_k \sim N(\frac{1}{N_k} \sum_{i=1}^{N_k} \hat{J}_k(i), \frac{{\sigma_k}^2}{N_k}),$$

where $\frac{1}{N_k} \sum_{i=1}^{N_k} \hat{J}_k(i) = \bar{J}_k$. Here $\bar{J}_k$ is the sample mean for design k and the variance ${\sigma_k}^2$ is approximated by the sample variance in simulation.

3.3 Selecting *m* Simplest Good Enough Designs

3.3.1 Problem Formulation

In this section, we formulate the problem of selecting m simplest good enough designs from the total of K designs and introduce the simulation budget allocation algorithm called OCBA-mSG. We find an efficient simulation budget allocation procedures to maximize the probability of correct selection for the m

designs with limited simulation budget T based on the OCBA introduced in Section 2.2. In other words, we want to find m simplest designs whose performance are good enough or acceptable with the maximized probability that the selected m designs are really the simplest good enough designs.

The *good enough design* satisfies $J_k < J_0$, where J_0 is a given constraint on the performance. Hence, the good enough set (or the feasible set) is defined as

$$F = \{k | J_k < J_0, k = 1, 2, \ldots, K\}.$$

We use the descriptive complexity $C(\theta_k)$, which is considered to be a discrete value in the set $\{0, 1, \ldots, n\}, n < K$, to describe the complexity of the design θ_k. Assume that we know the value of $C(\theta_k)$ once we simulate that design, and $C(\theta_k)$ remains the same afterwards. And also we define the complexity set C_i as

$$C_i = \{k | C(\theta_k) = i, k = 1, 2, \ldots, K\},$$

which contains all the designs with complexity i.

Then we can define the set of m simplest and good enough designs as

$$S_m = \{m_1, m_2, \ldots, m_m \mid C(\theta_{m_i}) \leqslant C(\theta_k), \forall k \in F \setminus S_m\},$$

where $F \setminus S_m = \{k \in F | k \notin S_m\}$. There may be several different S_m, however any S_m that satisfies the above definition will be considered as the set of m simplest good enough designs although they may not be the best as we will discuss for OCBA-bSG in Section 3.4.

Fig. 3.3 gives a pictorial view of the general case. We first ordered all the K designs according to their complexity $C(\theta_k)$ and their performance J_k. Suppose that all the designs in the complexity sets C_0, C_1, ..., C_{t-1} ($1 \leq t \leq n$) are infeasible (or good enough) and the first feasible design appears in the set C_t. If the set C_t has m feasible designs, then we get m simplest good enough designs in C_t. If there are less than m feasible designs in C_t, then we continue searching the sets C_{t+1}, C_{t+2}, ..., until we find m feasible designs. After searching the set C_n, if there are still less than m feasible designs, then we consider all the feasible designs as the optimal designs. In general there are three types of subsets we need to consider as shown in Fig. 3.3: infeasible simplest subsets S_{d_i}, $i = 0, 1, \ldots, t-1$; simplest good enough subsets S_{s_i}, $i = 0, 1, \ldots, p$; and infeasible non-simplest subsets S_{e_i}, $i = 0, 1, \ldots, p$. And the simplest good enough subsets

$S_{s_0}, S_{s_1}, \ldots, S_{s_p}$ satisfy

$$\sum_{i=0}^{p-1} |S_{s_i}| < m, \quad \sum_{i=0}^{p} |S_{s_i}| \geqslant m,$$

where $|\cdot|$ denotes the cardinality of the set. As a result, S_m should include all the designs in the subsets $S_{s_0}, S_{s_1}, \ldots, S_{s_{p-1}}$ and *any* $(m - \sum_{i=0}^{p-1} |S_{s_i}|)$ designs in the subset S_{s_p}, which implies that S_m may have several possibilities.

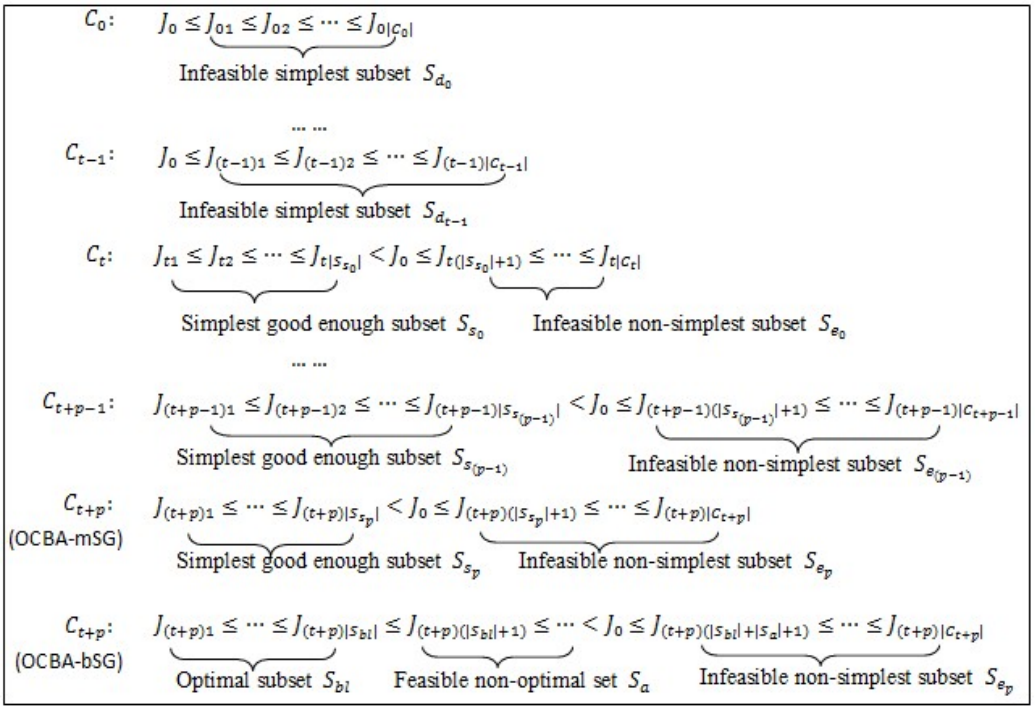

Figure 3.3: Relationship between subsets. J_{ij} denotes the performance of a design whose complexity is i and whose performance is the j^{th} smallest in its complexity set C_i.

In simulation, we estimate the performance J_k with its sample mean $\bar{J}_k$ at every iteration, and then order the designs as Fig. 3.3 to find the subsets $\{\hat{S}_{s_i}, i = 0, 1, \ldots, \hat{p}\}$, $\{\hat{S}_{d_i}, i = 0, 1, \ldots, \hat{t}-1\}$ and $\{\hat{S}_{e_i}, i = 0, 1, \ldots, \hat{p}\}$ as estimates for S_{s_i}, S_{d_i} and S_{e_i} respectively based on our current samples.

Then the correct selection CS_m of the designs selected in $\hat{S}_s$ are the true simplest good enough designs can be defined as, i.e.,

$$CS_m = \{J_i < J_0 \ \& \ J_j \geqslant J_0, \forall i \in \bigcup_{i=0}^{\hat{p}} \hat{S}_{s_i}, \forall j \in \left\{\cup_{i=0}^{\hat{p}} \hat{S}_{e_i}\right\} \bigcup \left\{\cup_{i=0}^{\hat{t}-1} \hat{S}_{d_i}\right\}\}.$$

In order to get a higher probability of correct selection $P(CS_m)$ for m simplest good enough designs, we want the estimates of the sets $\hat{S}_{s_i}$, $\hat{S}_{d_i}$ and $\hat{S}_{e_i}$ be as precise as possible. As we know, the more simulation budget we allocate to a particular design, the closer the sample mean to the true mean is. However, the simulation budget T is limited, we do not have enough simulation budget for every design, especially when the total number of design is large. As a result, in order to get a precise selection of the simplest good enough set S_m, we need to get a better estimation for the performance of critical designs which will provide us a precise estimation of the sets $\hat{S}_{s_i}$, $\hat{S}_{d_i}$ and $\hat{S}_{e_i}$. Therefore, the problem of OCBA-mSG is to determine the simulation budget allocation $N_1, N_2, \ldots, N_K$ for every design to maximize the probability of correct selection with a given total simulation budget T. Assuming noninformative normal prior distribution for every design and using the result from Section 3.2, the OCBA-mSG problem can be formulated as:

$$\begin{aligned} \max_{N_1,N_2,\ldots,N_K} \quad & P(CS_m) = P\{\tilde{J}_i < J_0 \ \& \ \tilde{J}_j \geqslant J_0\} \\ \text{s.t.} \quad & N_1 + N_2 + \ldots + N_K = T, \end{aligned} \tag{3.1}$$

where $\forall i \in \bigcup_{i=0}^{\hat{p}} \hat{S}_{s_i}, \forall j \in \left\{\cup_{i=0}^{\hat{p}} \hat{S}_{e_i}\right\} \bigcup \left\{\cup_{i=0}^{\hat{t}-1} \hat{S}_{d_i}\right\}$. Here, we follow a similar way to formulate the problem as OCBA in Section 2.2 due to its clear problem definition and explanation. By using APCS in [1], we maximize $P(CS_m)$ asymptotically as $T \to \infty$ in Theorem 1.

3.3.2 Methodology

Theorem 1. *Given a total number of T simulation runs, we allocate the simulation budget N_i to each design i, $i = 1, 2, \ldots, K$ by maximizing $P(CS_m)$ asymptotically (as $T \to \infty$). Here, J_0 is the good enough constraint. σ_i^2 and $\bar{J}_i$ are sample variance and sample mean for designs θ_i. Then N_i is computed by the following allocation rule:*

$$\frac{N_i}{\sigma_i^2/(\bar{J}_i - J_0)^2} = \frac{N_j}{\sigma_j^2/(\bar{J}_j - J_0)^2}, \tag{3.2}$$

for all $i \in \bigcup_{i=0}^{\hat{p}} \hat{S}_{s_i}$ and $j \in \left\{\cup_{i=0}^{\hat{p}} \hat{S}_{e_i}\right\} \bigcup \left\{\cup_{i=0}^{\hat{t}-1} \hat{S}_{d_i}\right\}$. $N_k = 0$ for all other $k \in \{1, 2, \ldots, K\}$.

From the relations in Theorem 1, we know that the simulation budget for every design changes according to its corresponding sample variance. If a design has a larger sample variance, then more simulation budget will be allocated to that particular design in order to find a more accurate sample mean to estimate its true performance. We also notice that at the critical point of the good enough performance constraint J_0, the designs nearer to J_0 will be assigned more simulation budget, since they are more sensitive to the feasibility test of J_0. This result is consistent with what we get in Section 2.1 and 2.2 for enhanced two-stage procedure and OCBA algorithm. We all consider both sample mean and sample variance which have similar proportional relationship to simulation budget N_i as well.

3.4 Selecting the Best *m* Simplest Good Enough Designs

3.4.1 Problem Formulation

If we modify the previous problem slightly, we can get the *best m* simplest good enough designs not only *m* simplest good enough designs. The optimal set is defined as

$$S_b = \{b_1, \ldots, b_m \in F \mid C(\theta_{b_i}) < C(\theta_k) \text{ OR } J_{b_i} < J_k \text{ if } C(\theta_{b_i}) = C(\theta_k), \forall k \in F \setminus S_b\},$$

where $F \setminus S_b = \{k \in F | k \notin S_b\}$.

The definition for S_b means that we select the best m simplest good enough designs from the good enough designs in the small descriptive complexity set first until we reach a descriptive complexity set that there are more good enough designs than we need, and then we select the good enough deigns with small performance in this descriptive complexity to complete our mission. In other words, the only difference between S_m and S_b is that S_b selects the best $(m - \sum_{i=0}^{p-1} |S_{s_i}|)$ designs in the subset S_{s_p} instead of any $(m - \sum_{i=0}^{p-1} |S_{s_i}|)$ number of designs in the subset S_{s_p}.

Fig. 3.3 gives a pictorial for the general case where there are five subsets to be considered in this problem: infeasible simplest subsets S_{d_i}, $i = 0, 1, \ldots, t-1$; simplest good enough subsets S_{s_i}, $i = 0, 1, \ldots, p-1$; infeasible non-simplest

subsets S_{e_i}, $i = 0, 1, \ldots, p$; optimal subset S_{bl}; and feasible non-optimal subset S_a.

Similar to the previous problem, we can use the estimates of the subsets to define the correct selection CS_b as

$$CS_b = \{J_i < J_0 \ \& \ J_j \leqslant J_k < J_0 \ \& \ J_s \geqslant J_0, \forall i \in \bigcup_{i=0}^{\hat{p}-1} \hat{S}_{s_i},$$
$$\forall j \in \hat{S}_{bl}, \forall k \in \hat{S}_a, \forall s \in \left\{\cup_{i=0}^{\hat{p}} \hat{S}_{e_i}\right\} \bigcup \left\{\cup_{i=0}^{\hat{t}-1} \hat{S}_{d_i}\right\}\}.$$

Using the posterior distribution $\tilde{J}$ to approximate the true performance J, we formulate the OCBA-bSG problem to efficiently allocate the simulation budget $N_1, N_2, \ldots, N_K$ to every design with fixed total budget T to maximize the probability of correct selection for best m simplest good enough designs as follows:

$$\begin{aligned} \max_{N_1,N_2,\ldots,N_K} \quad & P(CS_b) = P\{\tilde{J}_i < J_0 \ \& \ \tilde{J}_j \leqslant \tilde{J}_k < J_0 \ \& \ \tilde{J}_s \geqslant J_0\} \\ \text{s.t.} \quad & N_1 + N_2 + \ldots + N_K = T, \end{aligned} \tag{3.3}$$

where $\forall i \in \bigcup_{i=0}^{\hat{p}-1} \hat{S}_{s_i}, \forall j \in \hat{S}_{bl}, \forall k \in \hat{S}_a, \forall s \in \left\{\cup_{i=0}^{\hat{p}} \hat{S}_{e_i}\right\} \bigcup \left\{\cup_{i=0}^{\hat{t}-1} \hat{S}_{d_i}\right\}$.

3.4.2 Methodology

The simulation budget allocation rules are stated in Thm 2, and the detail explanation is given in [1].

Theorem 2. *Given a total number of T simulation runs, we allocate the simulation budget N_i to each design i, $i = 1, 2, \ldots, K$ by maximizing the lower bound of $P(CS_b)$ asymptotically (as $T \to \infty$).*

Case 1: If there are more than m feasible designs, then allocate the simulation budget according to

$$\begin{aligned} \frac{N_i}{\sigma_i^2/(\bar{J}_i - J_0)^2} &= \frac{N_j}{\sigma_j^2/(\bar{J}_j - \mu)^2} = \frac{N_s}{\sigma_s^2/(\bar{J}_s - J_0)^2} \\ &= \frac{N_x}{\sigma_x^2/(\bar{J}_x - \mu)^2} = \frac{N_y}{\sigma_y^2/(\bar{J}_y - J_0)^2}, \end{aligned} \tag{3.4}$$

for all $i \in \bigcup_{i=0}^{\hat{p}-1} \hat{S}_{s_i}$, $j \in \hat{S}_{bl}$, $s \in \left\{\cup_{i=0}^{\hat{p}} \hat{S}_{e_i}\right\} \bigcup \left\{\cup_{i=0}^{\hat{t}-1} \hat{S}_{d_i}\right\}$, $x \in \{k \in \hat{S}_a | \bar{J}_k \leqslant \frac{\mu + J_0}{2}\}$, $y \in \{k \in \hat{S}_a | \bar{J}_k > \frac{\mu + J_0}{2}\}$. $N_k = 0$ for all other $k \in \{1, 2, \ldots, K\}$. Here μ is

determined by

$$\mu = \frac{\hat{\sigma}_{(t+p)(|S_{bl}|+1)}\bar{J}_{(t+p)|S_{bl}|} + \hat{\sigma}_{(t+p)|S_{bl}|}\bar{J}_{(t+p)(|S_{bl}|+1)}}{\hat{\sigma}_{(t+p)|S_{bl}|} + \hat{\sigma}_{(t+p)(|S_{bl}|+1)}}, \text{ where } \hat{\sigma}_{(t+p)i} = \frac{\sigma_{(t+p)i}}{\sqrt{N_{(t+p)i}}}. \tag{3.5}$$

Case 2: If there are no more than m feasible designs, then allocate the simulation budget according to

$$\frac{N_i}{\sigma_i^2/(\bar{J}_i - J_0)^2} = \frac{N_s}{\sigma_s^2/(\bar{J}_s - J_0)^2} \tag{3.6}$$

for all $i \in \bigcup_{i=0}^{\hat{p}-1} \hat{S}_{s_i} \bigcup \hat{S}_{bl}$ *and* $s \in \left\{\cup_{i=0}^{\hat{p}} \hat{S}_{e_i}\right\} \bigcup \left\{\cup_{i=0}^{\hat{t}-1} \hat{S}_{d_i}\right\}$. $N_k = 0$ *for all other* $k \in \{1,2,\ldots,K\}$.

As we seen from above theorem, there are two critical points μ and J_0, where μ is the critical point for the optimality and J_0 is the critical point for the feasibility. The designs closer to these two points will be assigned more simulation budget among all the designs in $C_0,\ldots,C_{t+p}$ we actually considered. For the optimality critical points, we need to consider the subset $\hat{S}_{bl}$, since the designs in this subset are close to μ. For the feasibility critical points, we need to consider the subsets $\bigcup_{i=0}^{\hat{p}-1} \hat{S}_{s_i}$ and $\left\{\cup_{i=0}^{\hat{p}} \hat{S}_{e_i}\right\} \bigcup \left\{\cup_{i=0}^{\hat{t}-1} \hat{S}_{d_i}\right\}$, since the designs in these subsets are near J_0. The most particular subset is $\hat{S}_a$, in which the designs are both close to μ and J_0. Therefore, we divide this subset $\hat{S}_a$ by the middle point $\frac{\mu+J_0}{2}$ and compare them seperately as shown in Fig. 3.4.

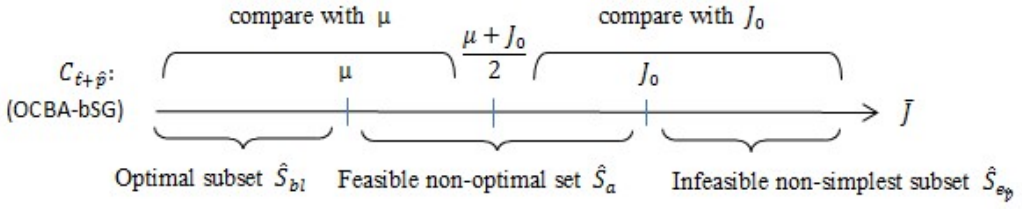

Figure 3.4: Simulation allocation in the set $C_{\hat{t}+\hat{p}}$.

3.5 Allocation Procedure

We describe the allocation procedures for both OCBA-mSG and OCBA-bSG in the following. The discussion about the upper bound NU is given in Sec 3.6.

OCBA-mSG and OCBA-bSG

Input:

- the total number of the designs K;
- the number of designs needed m;
- the total simulation budget T;
- the simulation budget increase at each iteration Δ;
- the initial simulation budget for every design n_0;
- the good enough performance constraint J_0;
- the upper bound of the total simulation budget for one design NU.

Initialize: $l = 0$.

- Perform n_0 simulation replications for all designs to generate samples $X_i^k,\ k = 1,2,\ldots,n_0,\ i = 1,2,\ldots,K$. Set $N^l = Kn_0$.
- Group the designs according to their complexities to obtain the complexity sets $C_0, C_1, \ldots, C_n$.

Loop: while $N^l < T$, do

1. Update:
   - For each design i, compute the sample mean $\bar{J}_i = \frac{1}{N_i^l}\sum_{k=1}^{N_i^l} X_i^k$;
   - For each design i, compute the sample standard deviation $\sigma_i = \sqrt{\sum_{k=1}^{N_i^l}(X_i^k - \bar{J}_i)^2/(N_i^l - 1)}$;
   - Sort the designs in each complexity set according to their sample means in the increasing order.
   - Increase the computing budget $N^{l+1} = \min\{N^l + \Delta, T\}$.
2. Allocate:

   OCBA-mSG

   - For each design θ_i, compute the simulation budget N_i^{l+1} according to (3.2).

OCBA-bSG

- If the total number of feasible designs is greater than m, compute μ according to (3.5), and compute the simulation budget N_i^{l+1} for each design θ_i according to (3.4).
- Otherwise, compute the simulation budget N_i^{l+1} for each design θ_i according to (3.6).

3. Simulate:

   - If $N_i^{l+1} \geqslant NU$ or $N_i^{l+1} \leqslant N_i^l$, we set $N_i^{l+1} = N_i^l$, and do not simulate design θ_i at this iteration.
   - Otherwise, perform $(N_i^{l+1} - N_i^l)$ simulations for design θ_i to generate more samples $X_i^k,\ k = N_i^l + 1, N_i^l + 2, \ldots, N_i^{l+1}$.

4. Update: $l \to l+1$.

End of loop

3.6 Upper Bound Analysis

In the last section, we impose an upper bound on the simulation budget in the allocation procedure for both the algorithms of OCBA-mSG and OCBA-bSG. The reason that we introduce this upper bound is that because we use the asymptotic limit $T \to \infty$ to get the simulation budget allocation rules in Thm 1 and Thm 2. By using the asymptotic limit $T \to \infty$, we assume that the total simulation budget is infinity, however in the real situations the total simulation budget T is always finite. Therefore, we are unable to allocate a very large amount of simulation budget to one design in real situations. As a result, we decide to use an upper bound to control the simulation budget for every design in order to ensure that there are enough simulation budget to be allocated to the other critical designs. In this section, we first introduce how the upper bound is calculated and then we compare it with the other fixed upper bound to see its performance.

3.6.1 Dynamic Upper Bound

For both OCBA-mSG and OCBA-bSG, we impose an dynamic upper bound NU on the simulation budget for every single design: if $N_i \geqslant NU$, we stop allocating new simulation budget to that design. Since the designs near the critical points need more simulation budget, we need to ensure each of such critical designs will be simulated at least once. Hence, we approximate the upper bound by counting the number of subsets related to the critical points after initialization, where those subsets are $\hat{S}_{s_i}$, $\hat{S}_{d_i}$, $\hat{S}_{e_i}$ in OCBA-mSG or $\hat{S}_{s_i}$, $\hat{S}_{bl}$, $\hat{S}_a$, $\hat{S}_{d_i}$, $\hat{S}_{e_i}$ in OCBA-bSG (c.f. Fig. 3.3).

$$NU = \frac{(T - Kn_0)}{total\ number\ of\ sets} + n_0. \tag{3.7}$$

Here, we compare this dynamic upper bound with other fixed upper bounds in the test problems. The test results show that our dynamic upper bound may not be the best upper bound under every situation, but in general it obtains acceptable performance for all the tests in this section. However, the performance of the fixed upper bound is highly dependent on the parameter setting of each test, and does not guarantee an overall good performance for all the tests.

Our experiments are conducted using Matlab. We use the probability of correct selection $P(CS)$ as the efficiency measurement. Since we cannot directly calculate $P(CS)$ analytically, we choose Monte Carlo simulation to estimate $P(CS)$ in the experiments. Thus $P(CS)$ will be computed as the ratio of the number of simulation runs with correct selections to the total number of simulation runs. As our target is to maximize the probability of correct selection in the objective function with given total simulation budget T, the faster the $P(CS)$ converges, the better the corresponding method is. In addition, for convenience, we assume that design θ_i has complexity $\lfloor \log_2 i \rfloor$, so the complexity is non-decreasing in i. And we compare the dynamic upper bound NU with the other four fixed upper bounds which equal to 600, 1000, 2000 and 3000 respectively in OCBA-mSG and OCBA-bSG.

3.6.2 Upper Bound Comparison for Example 1

OCBA-mSG We want to find 5 simplest good enough designs with good enough constraint $J_0 = 6.3$ from a total of 20 designs with the i^{th} design having

$L(\theta_i, \zeta)$ distributed according to the normal distribution $N(0.5i, i^2)$. Notice that the mean increases as the complexity increases, and the variance increases as the mean increases. The initial simulation budget $n_0 = 20$, simulation budget

Current Budget	P(CS) for Dynamic Upper Bound = NU	P(CS) for Upper Bound = 600	P(CS) for Upper Bound = 1000	P(CS) for Upper Bound = 2000	P(CS) for Upper Bound = 3000
600	0.73	0.7	0.66	0.71	0.72
800	0.95	0.89	0.95	0.94	0.93
1000	0.98	0.94	0.99	0.98	0.98
1200	1	0.97	0.99	0.98	0.99
1400	1	0.95	0.99	0.98	1
1600	1	0.97	0.99	0.98	1
1800	1	0.96	1	1	1
2000	1	0.97	1	1	1
2200	1	0.98	1	1	1
2400	1	0.98	1	1	1
2600	1	0.99	1	1	1
2800	1	1	1	1	1
3000	1	0.99	1	1	1
3200	1	0.99	1	1	1
3400	1	0.99	1	1	1
3600	1	0.98	1	1	1
3800	1	0.99	1	1	1
4000	1	0.99	1	1	1
4200	1	1	1	1	1
4400	1	1	1	1	1
4600	1	1	1	1	1
4800	1	0.99	1	1	1
5000	1	0.99	1	1	1

Figure 3.5: Upper bound comparison for OCBA-mSG in Example 1 of selecting 5 simplest good enough designs from 20 designs with distribution $N(0.5i, i^2)$ and $J_0 = 6.3$.

increment $\Delta = 200$, total simulation budget $T = 10000$, and total number of simulation runs = 200. The complexity sets are $C_0 = \{\theta_1\}$, $C_1 = \{\theta_2, \theta_3\}$, $C_2 = \{\theta_4, \theta_5, \theta_6, \theta_7\}$, $C_3 = \{\theta_8, \ldots, \theta_{15}\}$ and $C_4 = \{\theta_{16}, \ldots, \theta_{20}\}$. We record their probability of correct selection $P(CS)$ at each simulation budget level (every iteration). Since most part of $P(CS)$ in the record after budget exceeding 5000

equal to 1, we eliminate those and concentrate on the $P(CS)$ before simulation budget reaches 5000.

From the observation of the simulation results for OCBA-mSG, the $total\ number\ of\ sets$ related to the critical points are mainly between 3 to 5. Thus, in this example where $T = 10000$, $K = 20$ and $n_0 = 20$, we get the most frequent values for upper bound NU equal to 3200, 2420, 1940 as calculated below,

$$NU = \frac{(10000 - 20 * 20)}{total\ number\ of\ sets} + 20, \text{where total number of sets} = 3, 4, 5.$$

If we look at the performance for the five different upper bounds shown in Figure 3.5, it is clear that when upper bound equals to 600, $P(CS)$ converges slower compared to $P(CS)$ under the other four upper bounds. The performance of the dynamic upper bound NU is similar to that of the upper bound = 3000. Although when simulation budget equals to 1400 and 1600, $P(CS)$ for upper bound = 1000 and 2000 are not equal to 1, it is very close to 1. Thus, we are almost sure that the dynamic upper bound NU, upper bound = 1000, upper bound = 2000, and upper bound = 3000 have similar good performance. This conclusion is consistent with the frequent values we have calculated for upper bound NU above, since NU, with values 1940, 2420 and 3200 most of the time, are almost between the fixed upper bounds 2000 and 3000. Although when upper bound = 1000 we still get a good enough performance, our dynamic upper bound NU has the same good performance as well.

OCBA-bSG The parameter of this experiment is the same as the previous example except that the total simulation budget is 8000.

From the observation of the simulation results for OCBA-bSG, the $total\ number\ of\ sets$ related to the critical points takes values mostly between 3 to 6. Using the same formula (3.7) for NU by setting $T = 8000$, $K = 20$ and $n_0 = 20$, we get the most frequent values for NU which are 2553, 1920, 1540 and 1286.

The comparison result is given in Figure 3.6. $P(CS)$ for NU, upper bound = 1000 and upper bound = 2000 have higher value compared to that with upper bound = 600 and upper bound = 3000, which means that the OCBA-bSG under NU and upper bound = 1000 and 2000 have better performance. As calculated above, because the most frequent values for NU are 2553, 1920, 1540 and 1286

which are close to upper bound = 1000 and 2000, so the performance of dynamic upper bound NU is consistent with the performance of upper bound = 1000 and 2000. The performance for upper bound = 600 is relatively worse, because it does not allocate enough simulation budget to the critical designs since the largest simulation budget it allows to allocate to a particular design is only 600. On the other hand, the performance for upper bound = 3000 is also not good enough, because it allows up to 3000 simulation budget to be allocated to one

Current Budget	P(CS) for Dynamic Upper Bound = NU	P(CS) for Upper Bound = 600	P(CS) for Upper Bound = 1000	P(CS) for Upper Bound = 2000	P(CS) for Upper Bound = 3000
600	0.43	0.3	0.39	0.46	0.4
800	0.67	0.63	0.57	0.6	0.58
1000	0.78	0.67	0.66	0.72	0.65
1200	0.76	0.73	0.74	0.79	0.71
1400	0.77	0.74	0.79	0.8	0.74
1600	0.8	0.77	0.8	0.81	0.76
1800	0.85	0.77	0.82	0.84	0.77
2000	0.85	0.77	0.84	0.86	0.79
2200	0.84	0.79	0.83	0.86	0.8
2400	0.87	0.81	0.84	0.88	0.82
2600	0.87	0.85	0.85	0.88	0.83
2800	0.87	0.87	0.85	0.9	0.84
3000	0.87	0.87	0.86	0.9	0.85
3200	0.88	0.87	0.86	0.9	0.84
3400	0.88	0.86	0.88	0.9	0.86
3600	0.88	0.87	0.89	0.9	0.86
3800	0.88	0.89	0.87	0.9	0.86
4000	0.87	0.88	0.87	0.9	0.86
4200	0.88	0.88	0.88	0.9	0.86
4400	0.88	0.88	0.89	0.91	0.86
4600	0.89	0.89	0.9	0.9	0.86
4800	0.89	0.89	0.9	0.9	0.86
5000	0.9	0.89	0.91	0.91	0.86
6000	0.92	0.9	0.93	0.93	0.86
7000	0.93	0.92	0.93	0.95	0.87
8000	0.93	0.92	0.93	0.95	0.89

Figure 3.6: Upper bound comparison for OCBA-bSG in Example 1 of selecting 5 simplest good enough designs from 20 designs with distribution $N(0.5i, i^2)$ and $J_0 = 6.3$.

design and this amount is a bit higher in this particular example. Although the performance for upper bound = 2000 is a little better than the NU, the performance for NU is the second best in this example which considered to be acceptable.

3.6.3 Upper Bound Comparison for Example 2

Current Budget	P(CS) for Dynamic Upper Bound = NU	P(CS) for Upper Bound = 600	P(CS) for Upper Bound = 1000	P(CS) for Upper Bound = 2000	P(CS) for Upper Bound = 3000
600	0.42	0.36	0.34	0.46	0.41
800	0.62	0.56	0.49	0.6	0.54
1000	0.66	0.67	0.69	0.72	0.67
1200	0.78	0.79	0.83	0.79	0.75
1400	0.84	0.77	0.85	0.83	0.85
1600	0.84	0.8	0.87	0.85	0.86
1800	0.86	0.79	0.88	0.88	0.88
2000	0.87	0.81	0.88	0.92	0.91
2200	0.9	0.81	0.85	0.93	0.91
2400	0.9	0.82	0.88	0.93	0.93
2600	0.9	0.83	0.88	0.94	0.95
2800	0.91	0.82	0.89	0.97	0.95
3000	0.91	0.82	0.89	0.97	0.96
3200	0.91	0.83	0.88	0.96	0.96
3400	0.91	0.83	0.89	0.98	0.96
3600	0.92	0.83	0.89	0.98	0.96
3800	0.93	0.84	0.89	0.98	0.97
4000	0.94	0.84	0.9	0.98	0.97
4200	0.94	0.84	0.9	0.98	0.97
4400	0.94	0.84	0.9	0.98	0.97
4600	0.94	0.85	0.9	0.98	0.98
4800	0.95	0.85	0.9	0.99	0.98
5000	0.95	0.85	0.9	0.99	0.98
6000	0.95	0.85	0.91	0.99	0.98
7000	0.95	0.85	0.92	0.99	0.99
8000	0.95	0.85	0.92	0.99	1

Figure 3.7: Upper bound comparison for OCBA-mSG in Example 2 of selecting 5 simplest good enough designs from 20 designs with distribution $N(21-i,(0.5i)^2)$ and $J_0 = 7.3$.

OCBA-mSG We want to find 5 simplest good enough designs with good enough constraint $J_0 = 7.3$ from a total of 20 designs with the i^{th} design having $L(\theta_i, \zeta)$ distributed according to the normal distribution $N((21-i),(0.5i)^2)$. It is clear that the mean decreases as the complexity increases, and the variance increases as the mean decreases.The initial simulation budget $n_0 = 20$, simulation budget increment $\Delta = 200$, total simulation budget $T = 8000$, and total number of simulation runs = 200. The complexity sets are $C_0 = \{\theta_1\}$, $C_1 = \{\theta_2, \theta_3\}$, $C_2 = \{\theta_4, \theta_5, \theta_6, \theta_7\}$, $C_3 = \{\theta_8, \ldots, \theta_{15}\}$ and $C_4 = \{\theta_{16}, \ldots, \theta_{20}\}$.

From the observation of the simulation result for OCBA-mSG, the *total number of sets* related to the critical points falls between 5 to 7 in most times. By applying the formula (3.7) for NU with $T = 8000$, $K = 20$ and $n_0 = 20$, we get the most frequently values for NU are 1540, 1286 and 1105.

The comparison result is shown in Figure 3.7. The performance for dynamic upper bound NU is in the middle above the performance for the upper bounds = 600 and 1000, and below the performance for the upper bound = 2000 and 3000. This is reasonable since the most frequent values for NU are 1540, 1286 and 1105, which are between the upper bounds 1000 and 2000. Because the different pattens for mean and variance in this example, so the performance for upper bounds = 2000 and 3000 are the best. As we know, these 20 designs follow the normal distribution $N(21-i,(0.5i)^2)$, where the mean decreases as the design number increases and the variance increases as the design number increases. The correct selection of the five simplest good enough designs include $\{\theta_{14}, \theta_{15}\}$ and any three from $\{\theta_{16}, \theta_{17}, \theta_{18}, \theta_{19}, \theta_{20}\}$. Because design i follows distribution $N(21-i,(0.5i)^2)$ and the good enough constraint $J_0 = 7.3$, so the designs with small numbers are actually not as critical as we assumed when we build formula (3.7) since their performance are far from good enough and their variance are small. As a result, the total number of sets related to the critical points counted in this example is a bit higher than the reasonable critical points in the budget allocation procedure. Therefore, the performance of dynamic upper bound NU is not as good as that of the upper bounds = 2000 and 3000, and it is just in the middle of the performance.

OCBA-bSG From the observation of the simulation result for OCBA-bSG, the *total number of sets* related to the critical points falls between 5 to 8 at most times. By applying the Equation (3.7) for NU with $T = 8000$, $K = 20$ and $n_0 = 20$, we get the most frequently values for NU are 1540, 1286, 1105 and 970.

Current Budget	P(CS) for Dynamic Upper Bound = NU	P(CS) for Upper Bound = 600	P(CS) for Upper Bound = 1000	P(CS) for Upper Bound = 2000	P(CS) for Upper Bound = 3000
600	0.15	0.09	0.08	0.1	0.11
800	0.26	0.21	0.23	0.22	0.19
1000	0.39	0.3	0.36	0.32	0.35
1200	0.48	0.41	0.47	0.4	0.42
1400	0.58	0.43	0.5	0.46	0.54
1600	0.61	0.54	0.52	0.56	0.55
1800	0.6	0.54	0.6	0.6	0.65
2000	0.62	0.56	0.63	0.64	0.66
2200	0.63	0.56	0.65	0.68	0.7
2400	0.63	0.58	0.65	0.68	0.75
2600	0.67	0.61	0.66	0.68	0.78
2800	0.67	0.62	0.65	0.68	0.75
3000	0.69	0.6	0.67	0.69	0.77
3200	0.68	0.62	0.65	0.7	0.77
3400	0.71	0.62	0.66	0.71	0.8
3600	0.7	0.62	0.65	0.72	0.8
3800	0.72	0.63	0.67	0.74	0.8
4000	0.73	0.63	0.68	0.75	0.8
4200	0.73	0.64	0.7	0.75	0.8
4400	0.76	0.66	0.7	0.74	0.81
4600	0.76	0.66	0.71	0.76	0.82
4800	0.76	0.67	0.73	0.76	0.82
5000	0.78	0.68	0.78	0.76	0.82
6000	0.8	0.7	0.78	0.78	0.83
7000	0.81	0.72	0.81	0.79	0.84
8000	0.82	0.75	0.83	0.8	0.85

Figure 3.8: Upper bound comparison for OCBA-bSG in Example 2 of selecting 5 simplest good enough designs from 20 designs with distribution $N(21-i,(0.5i)^2)$ and $J_0 = 7.3$.

The comparison result is displayed in Figure 3.8. When upper bound = 3000, the $P(CS)$ is higher than the other four cases. It is because that the correct selection are the set $\{\theta_{14},\theta_{15},\theta_{18},\theta_{19},\theta_{20}\}$ which have relatively large variance. As we explained in OCBA-mSG of previous example, by allowing more simulation budget to be allocated to these designs, we get more accurate estimation for their performance. Therefore, in this example, upper bound =

3000 has the best performance. Although the dynamic upper bound does not get the best performance, it is in the second place together with upper bound = 1000 and 2000. In addition, since the difference of performance between dynamic upper bound NU and upper bound = 3000 is not significant, we conclude that NU is acceptable in this example.

3.6.4 Remark on Upper Bound

The examples above indicate that the dynamic upper bound NU has a satisfied performance in comparison to the other fixed upper bounds. However, when some critical designs are not actually so critical as what we see in Section 3.6.3, the performance of NU will not be so excellent under this situation. This is because of the formulation for NU in Equation (3.7), where all the subsets $\hat{S}_{s_i}$, $\hat{S}_{d_i}$, $\hat{S}_{e_i}$ in OCBA-mSG or $\hat{S}_{s_i}$, $\hat{S}_{bl}$, $\hat{S}_a$, $\hat{S}_{d_i}$, $\hat{S}_{e_i}$ in OCBA-bSG are counted as the $total\ number\ of\ sets$ related to the critical points. Therefore, when the designs in some of these subsets are not critical at all, the $total\ number\ of\ sets$ will be counted more than their actual value is, which results in a lower value for NU.

3.7 Experimental Results

In this section, we compare OCBA-mSG and OCBA-bSG with two other methods - Equal Allocation and Levin Search to test the efficiency of our methods. The experiments will be demonstrated on four test cases using Matlab, including one test case from inventory control problem mentioned in the introduction. The two other methods in the experiments are briefly introduced below.

Equal Allocation (EA) allocates the total simulation budget T equally among all the designs and do not consider any characteristics of the designs such as the mean, the variance or the complexity of the design. At iteration l, it allocates the Δ simulation budget according to

$$N_i^{l+1} - N_i^l = \Delta/K, \ \ \forall i \in \{1,2,\ldots,K\}.$$

Levin Search (LS) method [30] allocates simulation budget to the designs sequentially in the order of the complexity. It is useful when applied to find one

simplest and good enough design [23]. LS first simulates the designs with smallest complexity until obtaining a certain accuracy for the estimates of the performance, based on which the good enough designs are selected. If only less than m good enough designs are found, it then continues to simulate the designs in the next complexity set until finding m simplest good enough designs eventually. In our implementation, we simulate every design for n_0 times at initialization, and order them according to their sample means and complexities. Since it is hard to specify a given accuracy for the estimates in our examples, we evenly allocate the total remaining budget $(T - Kn_0)/K$ to all the designs beforehand, but simulate the designs sequentially, i.e., start simulating the first simplest design for $(T - Kn_0)/K$ times and then move on to the next one to repeat the same procedure. LS method will be the same as the EA in the end when all the T simulation budget is used, but it performs differently during the process. In general, LS method performs better if the performance deteriorates as the complexity increases.

3.7.1 Example 1 (Mean increases as complexity increases)

The example is the same as in Section 3.6.2, except the total simulation budget is 8000 and the total number of simulation runs = 10^4.

OCBA-mSG The correct selection of the five simplest good enough designs should include $\{\theta_1, \theta_2, \theta_3\}$ and any two from $\{\theta_4, \theta_5, \theta_6\}$. Fig. 3.9 shows that OCBA-mSG converges faster than EA and LS. EA performs well in this example because of the small total number of designs K and the small variance σ^2. LS searches from the simplest sets $\{\theta_1\}$, $\{\theta_2, \theta_3\}$, ..., and the critical designs are θ_6 and θ_7, so LS converges in about 7 interations.

OCBA-bSG The correct selection is $\{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5\}$. Fig. 3.10 shows the simulation result. OCBA-bSG obtains a high $P(CS)_b$ faster than EA and LS, which means OCBA-bSG can select the optimal designs more accurate. Similarly, LS converges in about 7 iterations due to the mean is increasing as the complexity increases.

Analysis for OCBA-mSG and OCBA-bSG in Example 1 Although both OCBA-mSG and OCBA-bSG are choosing 5 simplest good enough designs,

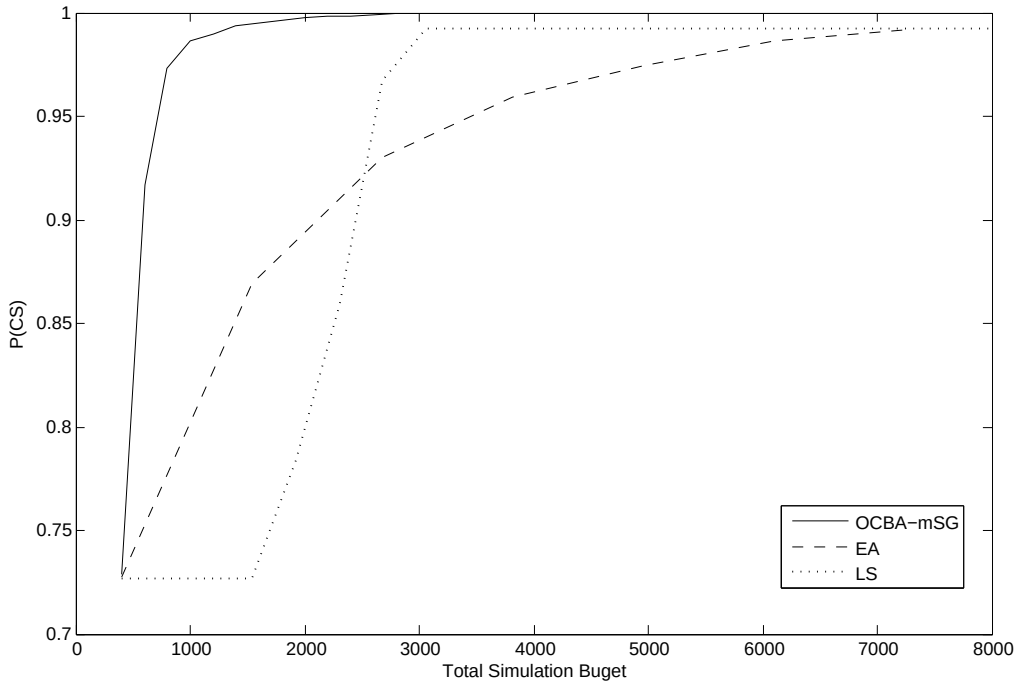

Figure 3.9: Example 1 - selecting 5 simplest good enough designs from 20 designs with distribution $N(0.5i, i^2)$ and $J_0 = 6.3$.

OCBA-mSG has a larger correct set since any two designs chosen from $\{\theta_4, \theta_5, \theta_6\}$ will be counted as correct selection. Thus, OCBA-mSG converges faster compared to OCBA-bSG in Example 1. With the same simulation budget $T = 8000$, OCBA-mSG has about 5% higher probability of correct selection compared to that of OCBA-bSG. For similar reason, EA and LS in Fig. 3.9 converges faster than that of Fig. 3.10. Besides, because the variances are increasing as the design number increasing, so the selected 5 simplest good enough designs are all with small variances. Thus, when we are using equal allocation, we can obtain a relatively correct estimation for the mean of these five designs, and the performance of EA is acceptable in this example.

3.7.2 Example 2 (Mean decreases as complexity increases)

The example is the same as that in Section 3.6.3 except the total number of simulation runs = 3000.

OCBA-mSG Correct selection of the five simplest good enough designs should include $\{\theta_{14}, \theta_{15}\}$ and any three from $\{\theta_{16}, \theta_{17}, \theta_{18}, \theta_{19}, \theta_{20}\}$. Fig. 3.11

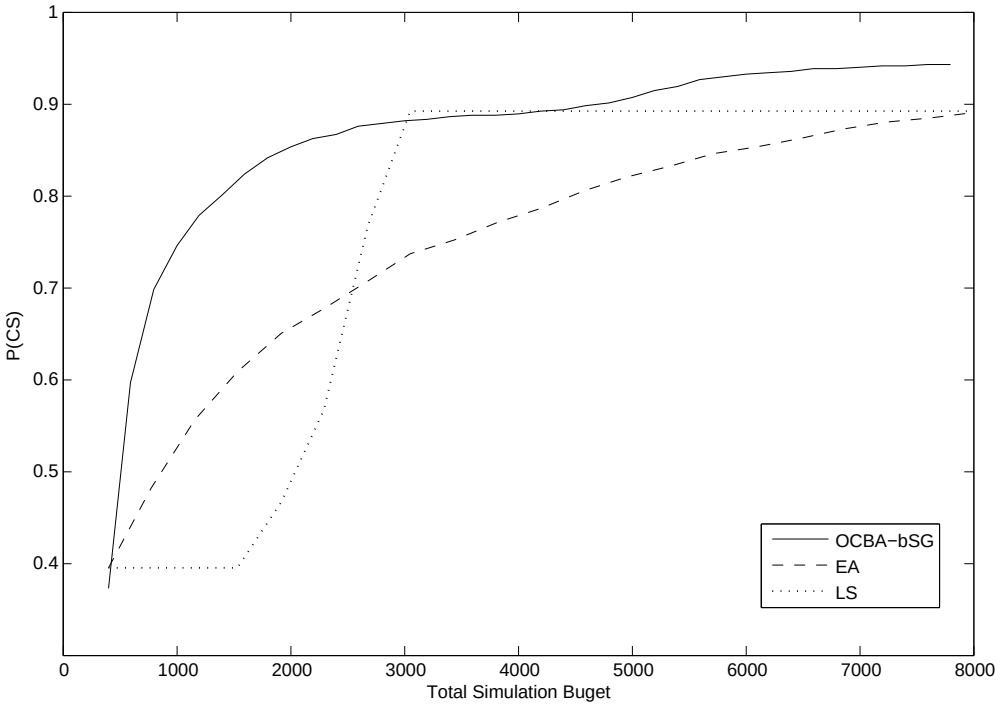

Figure 3.10: Example 1 - selecting the best 5 simplest good enough designs from 20 designs with distribution $N(0.5i, i^2)$ and $J_0 = 6.3$.

shows the simulation result. All three methods converge slower than that in Example 1, but OCBA-mSG still converges faster than EA and LS. LS still searches from the simplest sets while the correct selection is in the higher complexity sets, so LS method also converges slower than that of Example 1.

OCBA-bSG The correct selection is $\{\theta_{14}, \theta_{15}, \theta_{18}, \theta_{19}, \theta_{20}\}$. Fig. 3.12 shows the simulation result. Thus, LS converges near the critical design θ_{14} and θ_{18} as shows in the figure below since it searches from the simplest set. It is clear that OCBA-bSG outperforms.

Analysis for OCBA-mSG and OCBA-bSG in Example 2 For the similar reason explained in Example 1, OCBA-mSG converges faster than OCBA-bSG since OCBA-mSG has a higher probability to obtain the correct selection set. With the same simulation budget for OCBA-mSG and OCBA-bSG, OCBA-mSG gets 10% higher probability of correct selection compared to OCBA-bSG. Because the 5 simplest good enough designs are with relatively large variances, so it is hard to get an accurate estimation for their means. As a result, in Example 2 under Equal Allocation method, it converges slower and has a smaller correct

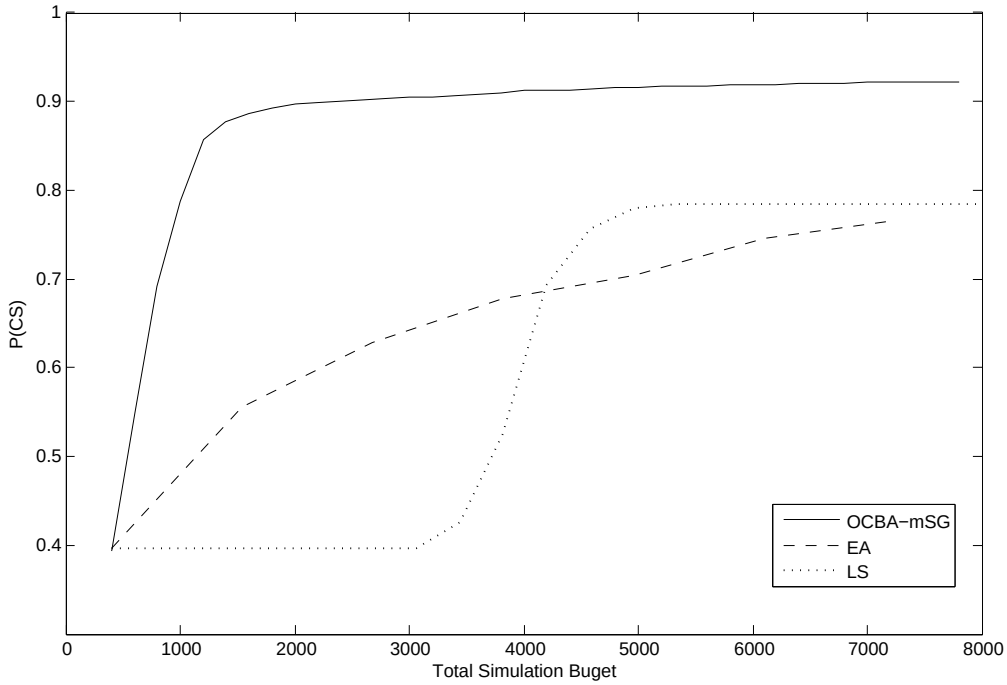

Figure 3.11: Example 2 - selecting 5 simplest good enough designs from 20 designs with distribution $N((21-i),(0.5i)^2)$ and $J_0 = 7.3$.

selection probability in the last iteration compared to that of Example 1.

3.7.3 Example 3 (Large-scale problem)

We want to find 5 simplest good enough designs with good enough constraint $J_0 = 6.3$ from a total of 65 designs with the i^{th} design having $L(\theta_i, \zeta)$ distributed according to the normal distribution $N((66-i),(0.05i)^2)$. The initial simulation budget $n_0 = 20$, simulation budget increment $\Delta = 200$, and total simulation budget $T = 8000$. The complexity sets are $C_0 = \{\theta_1\}$, $C_1 = \{\theta_2, \theta_3\}$, $C_2 = \{\theta_4, \ldots, \theta_7\}$, $C_3 = \{\theta_8, \ldots, \theta_{15}\}$, $C_4 = \{\theta_{16}, \ldots, \theta_{31}\}$, $C_5 = \{\theta_{32}, \ldots, \theta_{63}\}$ and $C_6 = \{\theta_{64}, \theta_{65}\}$.

OCBA-mSG The correct selection of the five simplest good enough designs should include $\{\theta_{60}, \theta_{61}, \theta_{62}, \theta_{63}\}$ and any one from $\{\theta_{64}, \theta_{65}\}$. For this large-scale problem , OCBA-mSG performs much better than EA and LS as shown in Fig. 3.13. Detailed explanation is similar to that for OCBA-bSG in the following.

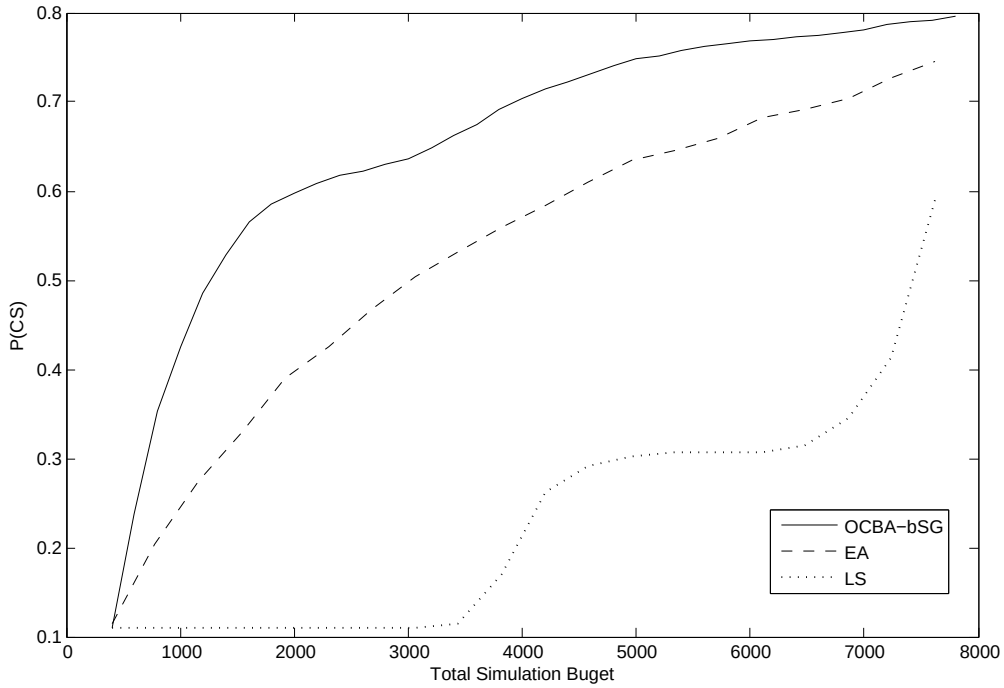

Figure 3.12: Example 2 - selecting the best 5 simplest good enough designs from 20 designs with distribution $N((21-i),(0.5i)^2)$ and $J_0 = 7.3$.

OCBA-bSG The correct selection is $\{\theta_{60}, \theta_{61}, \theta_{62}, \theta_{63}, \theta_{65}\}$. For this large-scale problem, OCBA-bSG performs much better than EA and LS as shown in Fig. 3.14. When the total design number K is large, EA converges slowly since each design is assigned with less simulation budget at every iteration compared to examples 1 and 2. For LS, the first time LS converges is the time that the total simulation budget reaches 4900, which is when it first starts to simulate designs in the set $\{\theta_{32}, \ldots, \theta_{63}\}$ with means $\{34, \ldots, 3\}$. As we assign the simulation budget according to the order of the designs in the same complexity set, here we simulate designs in the order of $\theta_{63}, \theta_{62}, \ldots$. Since designs $\theta_{63}, \theta_{62}, \theta_{61}$ and θ_{60} belong to the correct selection set, LS converges fast at this point. The second fast convergence for LS happens in the end due to the simulation budget allocation to the design θ_{65}.

Analysis for OCBA-mSG and OCBA-bSG in Example 3 In this large scale problem, when the total number of designs K is large and the number of critical designs are small (the most critical designs in Example 3 are $\theta_{59}, \theta_{60}, \theta_{64}, \theta_{65}$), by the allocation Theorem 1 and Theorem 2, more simulation budget will be allocated to these critical designs under OCBA-mSG and OCBA-bSG instead of

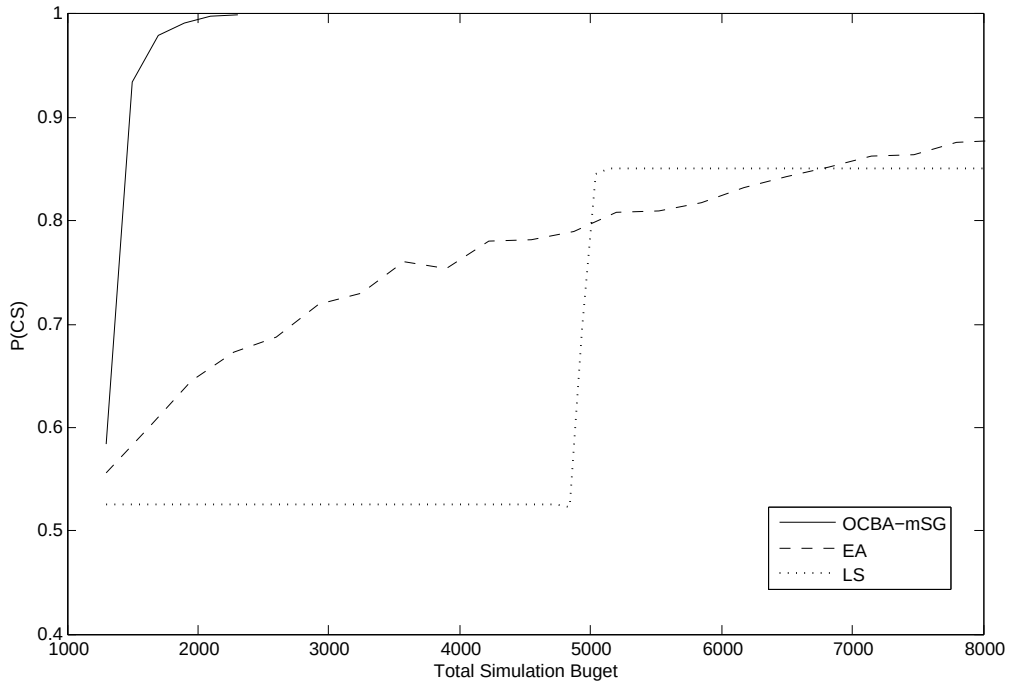

Figure 3.13: Example 3 - selecting 5 simplest good enough designs from 65 designs with distribution $N((66-i),(0.05i)^2)$ and $J_0 = 6.3$.

equal allocation to every design. Thus, OCBA-mSG and OCBA-bSG converges much faster than EA and LS.

3.7.4 Example 4 (Inventory control problem)

There are 9 stores, every store has the same holding cost $h = 0.05$, purchase cost $c = 0.4$, shortage cost $s = 0.1$ and initial inventory $x_0 = 10$. The demand at each period is a random variable satisfies $d = min(0, N(8, 4^2))$. Every store uses different thresholds functions to calculate the reorder amount at each period. Due to the number of thresholds, we classify them into three complexity sets as $C_0 = \{\theta_1, \theta_2, \theta_3\}$, $C_1 = \{\theta_4, \theta_5, \theta_6\}$ and $C_2 = \{\theta_7, \theta_8, \theta_9\}$. To be simplicity, we write the reorder threshold policy into one function but with different value for

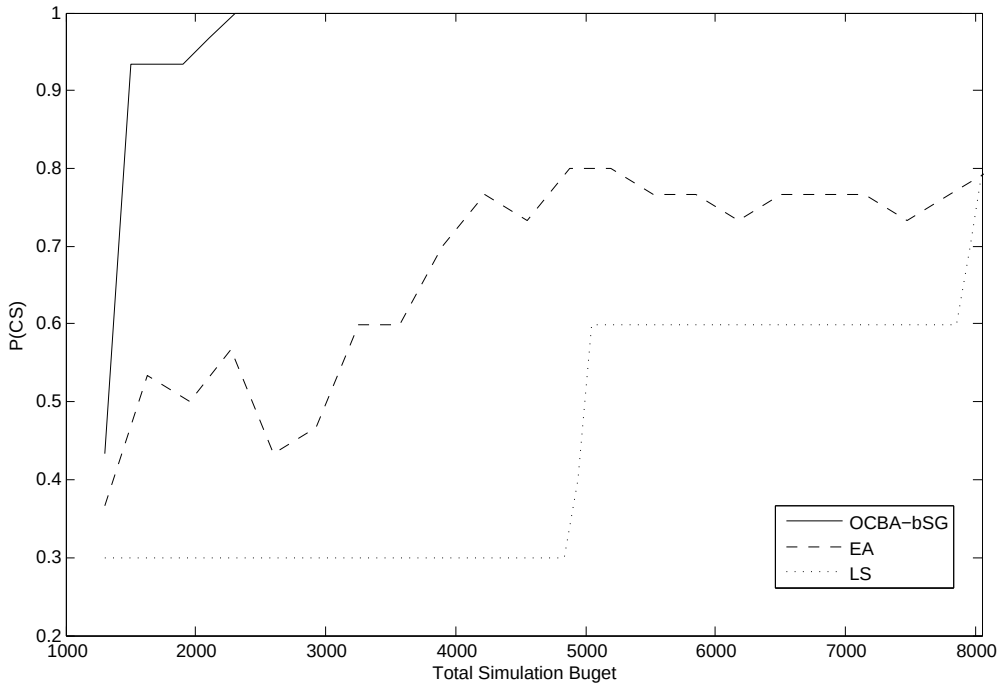

Figure 3.14: Example 3 - selecting the best 5 simplest good enough designs from 65 designs with distribution $N((66-i),(0.05i)^2)$ and $J_0 = 6.3$.

parameters.

$$u_k = \begin{cases} 0,\ if\ x_k \geqslant a_3. \\ a_3 - x_k,\ if\ a_2 \leqslant x_k < a_3. \\ a_2 - x_k,\ if\ a_1 \leqslant x_k < a_2. \\ a_1 - x_k,\ if\ x_k < a_1. \end{cases}$$

The value of (a_1, a_2, a_3) for 9 stores are $(0,0,15)$, $(0,0,20)$, $(0,0,25)$, $(0,10,15)$, $(0,10,20)$, $(0,20,45)$, $(5,10,15)$, $(10,15,20)$, $(15,20,25)$ respectively. In this example, we use the expected cost below as the performance measure.

$$E\{\sum_{k=1}^{n-1}(cu_k + |x_{k+1}|s\max(0,-x_{k+1}) + |x_{k+1}|h\max(0,x_{k+1}))\},$$

where $x_{k+1} = x_k + u_k - d,\ k = 0,1,\ldots,n-1$. Here u_k is the reorder amount at k_{th} period. We simulate every store for 10000 periods to get the estimation for the expected cost for 9 stores as 13.59, 9.56, 16.78, 25.64, 25.05, 71.65, 8.34, 2.78 and 4.63. And then we compare our results with these estimated expected costs

to calculate the $P(CS)$. The good enough constraint J_0 = 4.6 for OCBA-mSG and J_0 = 8.2 for OCBA-bSG, initial simulation budget $n_0 = 100$, simulation budget increment $\Delta = 1000$, total simulation budget $T = 10,000$, and total number of simulation runs = 100.

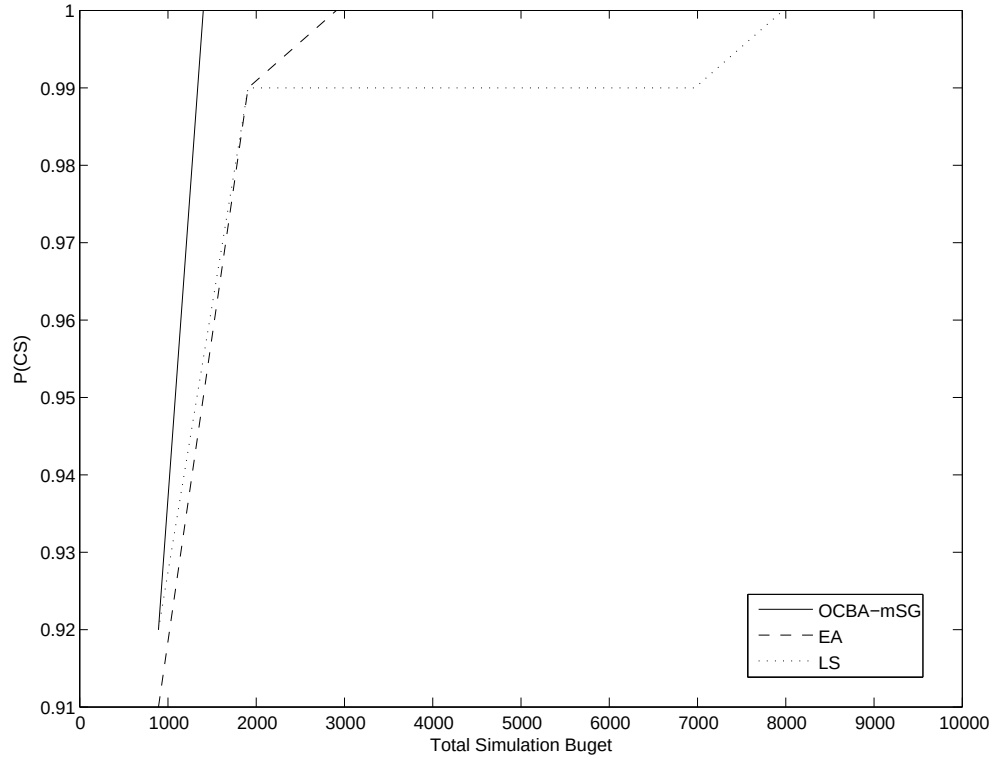

Figure 3.15: Example 4 - selecting 1 simplest good enough designs from 9 designs in inventory control problem.

OCBA-mSG The correct selection of the one desirable designs should be any one from $\{\theta_8, \theta_9\}$. For this inventory control problem, all the three methods perform well as shown in Fig. 3.15.

OCBA-bSG The correct selection of two desirable designs should be $\{\theta_8, \theta_9\}$. In Fig. 3.16, OCBA-bSG performs better than EA and LS. However, there is a small decreasing in $P(CS)$ of OCBA-bSG in the end. One possible reason is the variability of the performance measure for this inventory control problem. By performing test cases, the expected cost for the critical design θ_7 should be above J_0 in the long term, however, when the simulation amount is not that large, its

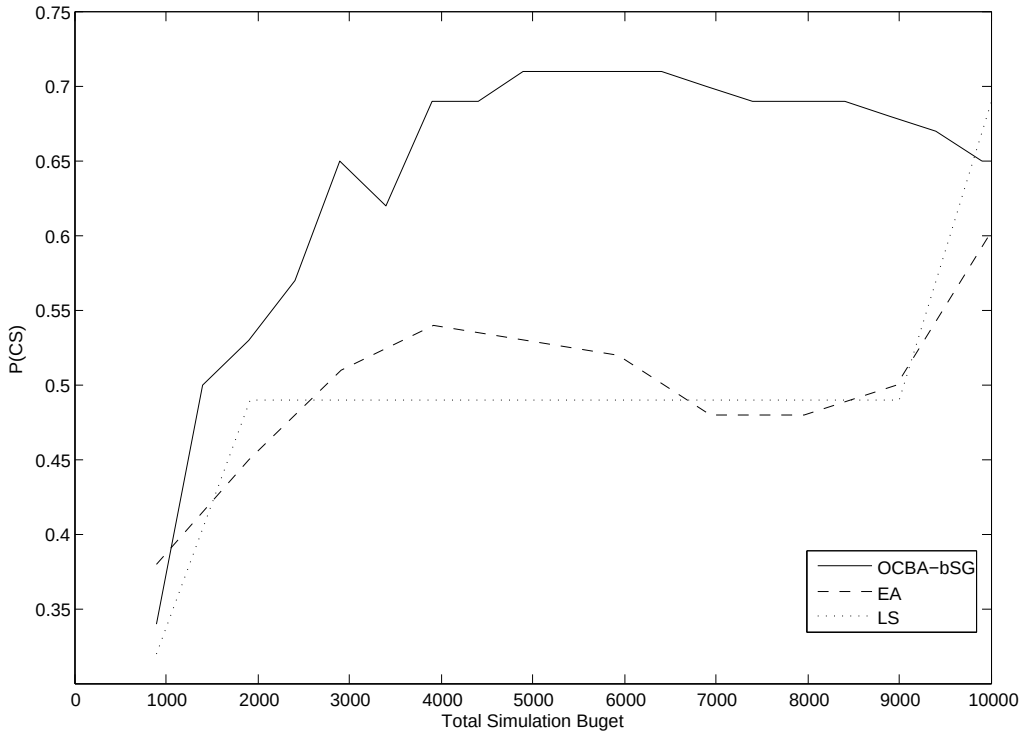

Figure 3.16: Example 4 - selecting the best 2 simplest good enough designs from 9 designs in inventory control problem.

expected cost may drop below J_0 as simulation budget increases which makes the $P(CS)$ decreases in OCBA-bSG. The similar situation may happen to design θ_2 as well. Therefore, if the simulation budget is large enough, we should obtain a stabilized $P(CS)$ for OCBA-bSG in an ideal situation. The other possible reason is that the total number of simulation runs is relatively small compare to the previous examples due to the complexity of this inventory control problem. As a result, several outliers may affect the result of Fig. 3.16 as well. On the other hand, since design θ_7 is the critical design, with similar explanation in the previous example, $P(CS)$ for LS also increases at the last round in this figure.

3.7.5 Comparison between OCBA-mSG and OCBA-bSG

Comparing selecting S_b with S_m, the difference is in C_{t+p}, where the subset S_{s_p} in selecting S_m is divided into two subsets S_{bl} and S_a in selecting S_b. Theorem 2 implies that in addition to allocating more simulation budget to the designs near

J_0 in the sets $C_0, C_1 \ldots, C_{t+p}$, we also allocate simulation budget to designs near μ in the set C_{t+p}. As a result, the extra simulation budget assigned to designs near μ in selecting S_b is approximately $2/(t+2p+2)$ of the total simulation budget in selecting S_m. If $t=0$ and $p=0$ (i.e., there are more than m feasible designs in the lowest complexity set C_0), then selecting S_b needs approximately double simulation budget of that in selecting S_m for the same accuracy of the sample means of the design performance. On the other hand, if $t+2p$ is large, selecting S_b requires little extra simulation budget, and will be preferred since it yields the *best m* designs among all simplest and good enough designs.

3.8 Conclusion for OCBA-mSG and OCBA-bSG

In this chapter, we introduce two new algorithms OCBA-mSG and OCBA-bSG for solving the problem of selecting m simplest good enough designs and best m simplest good enough designs respectively, which are also one kind of best design selection problems as we discussed in Section 2.3 except we select multiple simplest good enough designs. Our new algorithms are developed from the idea of the original OCBA algorithm introduced in Section 2.2. We use a similar problem formulation and approximation way as in OCBA, however we solve a new problem which has wide applications in WSNs and inventory control. The experimental results indicate that OCBA-mSG and OCBA-bSG can efficiently allocate simulation budget to critical designs by using the statistical information of both sample mean and sample variance of the design. Therefore, our algorithms provide an alternative way to efficiently select multiple simplest good enough designs from a total of K designs.

CHAPTER 4

CONCLUSION

In this thesis, we reviewed several methods related to my research on the best selection problem, such as two-stage procedure, optimal computing budget allocation and adaptive sampling algorithm. Based on these methods we developed new algorithms for the selection of m simplest good enough designs, which is useful in the node activation rule in wireless sensor networks and inventory control problems. The first algorithm we introduced is called OCBA-mSG which can efficiently allocate the simulation budget to achieve the high probability of correct selection. However, this algorithm only selects the m simplest good enough designs instead of the best m simplest good enough designs, which means that the selected designs do not guarantee to have the best performance. Since sometimes we may need the best m simplest good enough designs, we proposed another algorithm called OCBA-bSG to select the best designs by modifying OCBA-mSG a little bit. The comparison between OCBA-mSG and OCBA-bSG in Section 3.7.5 showed that under some situations, the choice of OCBA-bSG only slightly increase the simulation budget but get better results compared to OCBA-mSG. The upper bound analysis in Section 3.6 indicated that the upper bound derived in the two algorithms are in general effective and obtained a satisfied performance in examples considered in this thesis. The numerical results in Section 3.7 showed that both methods converge fast on all the experiments, which implied the efficiency of our methods.

CHAPTER 5

REFERENCES

[1] S. Yan, E. Zhou, and C.-H. Chen, "Efficient simulation budget allocation for selecting the best set of simplest good enough designs," *Proceedings of the 2010 Winter Simulation Conference*, pp. 1152–1159, 2010.

[2] "Inventory control," http://www.kereonsolutions.it/ksa/index.php?option=com_content&view=article&id=93&Itemid=62&lang=en.

[3] H. Banizaman, "Wireless sensor network," http://banizaman.ir/?cat=3, 2009.

[4] S. S. Gupta and M. Sobel, "On selecting a subset which contains all populations better than a standard," *The Annals of Mathematical Statistics*, vol. 29, pp. 235–244, 1958.

[5] S. S. Gupta, "On some multiple decision (selection and ranking) rules," *Technometrics*, vol. 7, no. 2, pp. 225–245, 1965.

[6] T. J. Santner, "A restricted subset selection approch to ranking and selection problems," *The Annals of Statistics*, vol. 3, no. 2, pp. 334–349, 1975.

[7] E. J. Dudewicz and S. R. Dalal, "Allocation of observation in ranking and selection with unequal variances," *Sankhya: The Indian Journal of Statistics*, vol. 37, pp. 28–78, 1975.

[8] Y. Rinott, "On two-stage selection procedures and related probability-inequalities," *Communications in Statistics*, vol. A7, pp. 799–811, 1978.

[9] L. W. Koenig and A. M. Law, "A procedure for selecting a subset of size m containing the l best of k independent normal populations," *Communications in Statistics - Simulation and Computation*, vol. 14, pp. 719–734, 1985.

[10] S. E. Chick, "Selecting the best system: A decision-theoretic approach," *Proceedings of the 1997 Winter Simulation Conference*, pp. 326–333, 1997.

[11] C.-H. Chen, "An effective approach to smartly allocate computing budget for discrete event simulation," *Proceddings of the 34th IEEE Conference on Decision and Control*, vol. 3, pp. 2598–2603, 1995.

[12] C.-H. Chen, "A lower bound for the correct subset-selection probability and its application to discrete-event system simulations," *IEEE Transactions on Automatic Control*, vol. 41, no. 8, pp. 1227–1231, 1996.

[13] H.-C. Chen, C.-H. Chen, L. Dai, and E. Y*ü*cesan, "New development of optimal computing budget allocation for discrete event simulation," *Proceddings of the 1997 Winter Simulation Conference*, pp. 334–341, 1997.

[14] C.-H. Chen, J. Lin, E. Y*ü*cesan, and S. E. Chick, "Simulation budget allocation for further enhancing the efficiency of ordinal optimization," *Journal of Discrete Event Dynamic Systems: Theory and Applications*, vol. 10, pp. 251–270, 2000.

[15] C.-H. Chen and E. Y*ü*cesan, "An alternative simulation budget allocation scheme for efficient simulation," *International Journal of Simulation and Process Modeling*, vol. 1, no. 1/2, pp. 49–57, 2005.

[16] C.-H. Chen, D. H. He, M. Fu, and L. H. Lee, "Efficient simulation budget allocation for selecting an optimal subset," *INFORMS Journal on Computing*, vol. 20, no. 4, pp. 579–595, 2008.

[17] N. A. Pujowidianto, L. H. Lee, C.-H. Chen, and C. M. Yap, "Optimal computing budget allocation for constrained optimization," *Proceedings of 2009 Winter Simulation Conference*, pp. 584–589, 2009.

[18] J. Branke, S. E. Chick, and C. Schmidt, "Selecting a selection procedure," *Management Science*, vol. 53, no. 12, pp. 1916–1932, 2007.

[19] S. Andrad*ó*ttir, D. Goldsman, B. W. Schmeiser, L. W. Schruben, and E. Y*ü*cesan, "Analysis methodology: Are we done?" *Proceedings of the 37th conference on Winter simulation*, pp. 790–796, 2005.

[20] D. Batur and S.-H. Kim, "Finding the set of feasible systems when the number of systems or constraints is large," *In Proceedings of the 2005 Winter Simulation Conference*, pp. 692–698, 2005.

[21] P. Glynn and S. Juneja, "A large deviations perspective on ordinal optimization," *In Proceedings of the 2005 Winter Simulation Conference*, pp. 692–698, 2005.

[22] R. Szechtman and E. Y*ü*cesan, "A new perspective on feasibility determination," *Proceedings of the 2008 Winter Simulation Conference*, pp. 273–280, 2008.

[23] Q. S. Jia, "An adaptive sampling algorithm for simulation-based optimization with descriptive complexity preference," 2009, submitted to *IEEE Transactions on Automation Science and Engineering*.

[24] E. J. Chen and W. D. Kelton, "An enhanced two-stage selection procedure," *Proceddings of the 2000 Winter Simulation Conference*, pp. 727–735, 2000.

[25] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Athena Scientific, 2005.

[26] H. P. Galliher, P. M. Morse, and M. Simond, "Dynamics of two classes of continuous-review inventory systems," *Operations Research*, vol. 7, no. 3, pp. 362–384, 1959.

[27] I. Sahin, "On the objective function behavior in (s, S) inventory models," *Operations Research*, vol. 30, no. 4, pp. 709–724, 1992.

[28] A. Federgruen and Y. S. Zheng, "An efficient algorithm for computing an optimal (r, Q) policy in continuous review stochastic inventory system," *Operations Research*, vol. 40, no. 4, pp. 808–813, 1992.

[29] K. Kar, A. Krishnamurthy, and N. Jaggi, "Dynamic node activation in networks of rechargeable sensors," *IEEE/ACM Transactions on Networking*, vol. 14, no. 1, pp. 15–26, 2006.

[30] L. A. Levin, "Universal sequential search problems," *Problems of Information Transmission*, vol. 9, no. 3, pp. 265–266, 1973.