Submitted to *Operations Research*
manuscript OPRE-2013-06-297.R2

Fully Sequential Procedures for Large-Scale Ranking-and-Selection Problems in Parallel Computing Environments

Jun Luo
Antai College of Economics and Management, Shanghai Jiao Tong University, Shanghai, China, 200052
jluo_ms@sjtu.edu.cn

L. Jeff Hong
Department of Economics and Finance and Department of Management Sciences, College of Business, City University of Hong Kong, Kowloon, Hong Kong, jeffhong@cityu.edu.hk

Barry L. Nelson
Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, Illinois, 60208
nelsonb@northwestern.edu

Yang Wu
Department of Industrial Engineering and Logistics Management, Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong, wuyangnju@gmail.com

Fully sequential ranking-and-selection (R&S) procedures to find the best from a finite set of simulated alternatives are often designed to be implemented on a single processor. However, parallel computing environments, such as multi-core personal computers and many-core servers, are becoming ubiquitous and easily accessible for ordinary users. In this paper, we propose two types of fully sequential procedures that can be used in parallel computing environments. We call them vector-filling procedures and asymptotic parallel selection procedures, respectively. Extensive numerical experiments show that the proposed procedures can take advantage of multiple parallel processors and solve large-scale R&S problems.

Key words: fully sequential procedures, parallel computing, statistical issues, asymptotic validity
History: Received June 2013; revision received March 2014; accepted June 2015.

1. Introduction

Selecting the alternative with the largest or smallest mean performance from a finite number of alternatives is a common problem in many areas of operations research and management science. For instance, in designing a multi-stage manufacturing line one may need to determine the best allocation of the buffer space to maximize the average throughput; in controlling an inventory sys-

tem one may need to identify the best reorder point to minimize the average cost; and in managing an ambulance service one may need to select the optimal vehicle dispatching policy to minimize the average response time. In all of these examples the mean performances of the alternatives may be evaluated by running simulation experiments. This type of optimization problem is known as a ranking-and-selection (R&S) problem in the simulation literature.

Many R&S procedures have been developed (see, for instance, Kim and Nelson (2006b) for an introduction to the topic). These procedures typically allocate the simulation effort to all alternatives such that the best can be selected with certain statistical guarantees, e.g., a pre-specified probability of correct selection (PCS). However, these procedures are often designed to handle a relatively small number of alternatives. As pointed out by Kim and Nelson (2006b), the two-stage procedure of Rinott (1978), hereinafter called Rinott's procedure, is typically applied to fewer than 20 alternatives, and the fully sequential procedure of Kim and Nelson (2001), hereinafter called $\mathcal{KN}$, is considered useful for fewer than 500 alternatives. The NSGS procedure of Nelson et al. (2001) is designed specifically to solve large-scale R&S problems. However, the largest test problem reported in their paper has only 500 alternatives.

In practice, however, there are many R&S problems that have thousands to tens of thousands of alternatives. Traditionally, these problems are solved using optimization-via-simulation (OvS) algorithms (see, for instance, Hong and Nelson (2009) for a recent review of OvS). Many of the OvS algorithms for this type of problem guarantee global convergence, i.e., they guarantee selecting the best alternative as the simulation effort goes to infinity. To achieve global convergence, however, these algorithms evaluate all alternatives as the simulation effort goes to infinity, and therefore become essentially R&S procedures. When they stop short of infinity, as they always do, there is often no statistical guarantee on the quality of the selected solution and the solution may be significantly inferior to the optimal one. *The goal of this paper is to provide R&S procedures, where the objective is to select the system with the largest mean response, that are valid and effective in parallel computing environments.*

In the past few years there has been rapid adoption of parallel computing. Multiple-core processors are ubiquitous today; they are used not only in servers and personal computers, but also in tablet computers and smart phones. Moreover, large quantities of computing (e.g., parallel processors) delivered as a service through the Internet, often called cloud computing, is becoming readily available and affordable to ordinary users. This motivates us to consider how to solve large-scale R&S problems in parallel computing environments. In particular, we are interested in whether current R&S procedures are statistically valid and efficient in parallel computing environments, and if they are not, how to design new procedures that are.

R&S problems can fit easily into parallel computing environments. If most of the computing time is used to generate independent simulation observations from various alternatives, then this can be done by executing the simulation programs in a parallel scheme without requiring any synchronization among different processors. This level of parallelization is called "embarrassingly parallel" (see, for instance, Foster (1995)) and it makes parallel computing very attractive to solve R&S problems. This advantage of using parallel simulation technology for R&S problems has also been discussed by Chen (2005) and Yücesan et al. (2001).

The total computing effort required to solve a R&S problem typically increases only moderately as the problem size increases. Taking Rinott's procedure (which samples from each alternative in two stages and only compares results after all sampling is completed) as an example, we plot the expected total number of samples as a function of the number of alternatives k in the solid line in Figure 1 (see Rinott (1978) for the procedure and Nelson et al. (2001) for a similar figure). To make this result more intuitive, suppose that we have 100 parallel processors and each processor can handle a R&S problem with 500 alternatives on its own, as a single processor in the allowable amount of time using Rinott's procedure. Then, in the same amount of time, Rinott's procedure on all processors can handle a similar problem with at least 30,000 alternatives, which significantly enlarges the size of R&S problems that may be solvable. In Figure 1, we also plot the maximum (or worst-case) and the average expected total numbers of samples for the $\mathcal{KN}$ procedure (see Kim and Nelson (2001) for the details). Notice that, when the number of alternatives increases, the proportion of clearly inferior alternatives often increases much faster than that of good alternatives. Therefore, fully sequential procedures, e.g., $\mathcal{KN}$, that allow early elimination often require a much smaller expected sample size than the worst case, which makes fully sequential procedures more attractive for large-scale R&S problems than two-stage procedures such as Rinott's.

From an implementation point of view, two-stage procedures are easier to parallelize than sequential procedures. For instance, a naive approach to implementing Rinott's procedure is as follows: In the first stage, we distribute kn_0 replications equally among all processors and compute the first-stage sample variances and second-stage sample sizes of all alternatives after all processors finish their jobs. In the second stage, we again distribute all additional samples equally among all processors and select the alternative with best sample mean after all processors finish their jobs. Notice that the total time required to complete the procedure is determined by the processor that finishes its job last. When replication times of different alternatives are different or they are random, the total time of this approach may be quite long and cause many processors to be idle. To improve efficiency, one may estimate the replication time for each alternative after the first stage, formulate the sample allocation problem in the second stage as a stochastic parallel machine

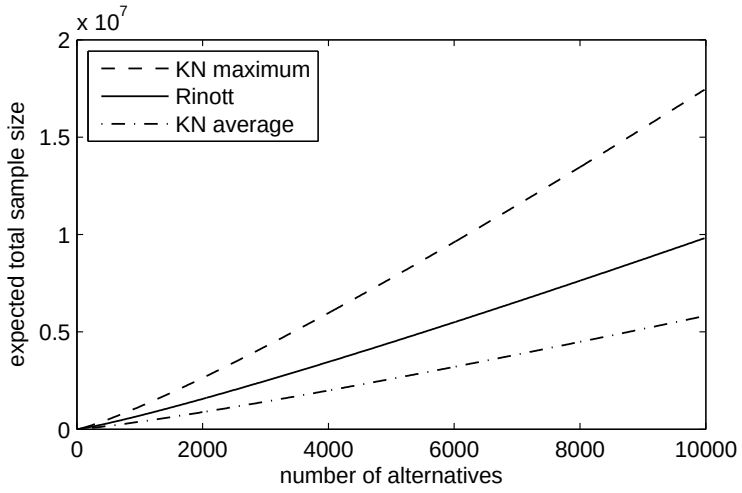


Figure 1 The expected total number of observations v.s. the number of alternatives for both Rinott and $\mathcal{KN}$ when the initial sample size $n_0 = 16$, variances across all alternatives $\sigma_i^2 = 1$, and the indifference-zone parameter $\delta = \sigma_i/\sqrt{n_0}$ in the slippage configuration where the difference between the means of the best and all other alternatives equals to δ.

scheduling problem and minimize its makespan (i.e., total time to completion). Interested readers may refer to Chapter 12 of Pinedo (2008) for more background on scheduling.

If the number of alternatives is less than or equal to the number of processors, it makes sense to use multi-stage procedures, such as Rinott's procedure, to reduce communication among processors. If the number of alternatives is much larger than the number of processors, however, it makes more sense to use fully sequential procedures with eliminations, such as $\mathcal{KN}$, to save simulation effort by eliminating inferior alternatives early. Since large-scale R&S problems typically have a number of alternatives that is a few orders-of-magnitude larger than the number of available processors, in this paper we focus on designing fully sequential procedures to solve such problems.

There are many different configurations of parallel computing environments, ranging from multi-core personal computers to many-core servers to local computer farms to clouds on the Internet. We focus mainly on designing statistically valid fully sequential procedures for multi-core personal computers and many-core servers. Then, without communications via the Internet, the time for loading simulation programs to different processors and transmitting data among processors is almost negligible. When implementing these procedures in clouds on the Internet, however, there may be packet delays or even losses, which may affect the validity of the procedures. Therefore, we leave the design of fully sequential procedures for cloud implementations as a topic for future research.

When designing fully sequential procedures for a parallel computing environment, a critical question is "*what makes fully sequential procedures on multiple processors different from their*

counterparts on a single processor?" A succinct answer to this question is that "*the input and output sequences of observations are different on multiple processors, while they are same on a single processor.*" A single processor system works like a single-server queue, the departure (i.e., output) sequence is same as the arrival (i.e., input) sequence; while a multiple processor system works like a multiple-server queue, the departure sequence is in general different from the arrival sequence when the service time (i.e., replication time of an observation in our situation) is random. In a simulation study we may control the input sequence deterministically. For instance, in $\mathcal{KN}$ we simulate all alternatives one-at-a-time according to a predetermined order. Therefore, the output sequence of a single processor system is also the same deterministic sequence. However, the same deterministic input sequence on a multiple-processor system may result in a random output sequence.

The randomness in the output sequence creates implementation issues as well as statistical issues when designing fully sequential R&S procedures. From an implementation point of view, randomness in the output sequence makes sample size synchronization difficult. For instance, when alternative 1 has 30 observations, alternative 2 may have 40 while alternative 3 may have only 20. Thus, procedures that require perfect synchronization of sample sizes from all alternatives are either difficult to implement or inefficient (i.e., using only a portion of the observations, such as setting the sample size to 20 in our three-alternative example). However, implementation issues may be easy to handle as there exist fully sequential procedures that allow unequal sample sizes from different alternatives (e.g., Hong (2006)). The statistical issues caused by randomness in output sequence are more critical. First, when the performance of an alternative is correlated with its replication time, observations with shorter replication time tend to be available earlier and the sequence of output observations may not be independent even though they use independent random numbers. This problem also exists when simulating a single alternative using multiple processors; see Heidelberger (1988). Second, even when the performance of the alternatives are independent of their replication times (even when the replication times are constant), sample sizes of surviving alternatives depend on elimination decisions which in turn depend on sample-mean information of the alternatives. This type of dependence destroys the independence between sample means and sample sizes that are exploited in R&S procedures using a famous result from Stein (1945). More details on the statistical issues caused by random output sequences are discussed in Section 2.

In this paper we propose two solutions to deal with these issues. If one insists on making existing fully sequential R&S procedures suitable for parallel simulation schemes, implying that we may only perform comparisons based on the input sequence of samples, then we suggest creating a vector to record the observations exactly in the order of the input sequence and make comparisons based on a pre-determined comparison rule. For instance, to implement $\mathcal{KN}$, one may perform a comparison after all surviving alternatives have their first r observations available for any $r =$

$n_0, n_0 + 1, \ldots$. Therefore, the procedure will have the same statistical validity as $\mathcal{KN}$. We call this type of procedure *vector-filling (VF)* as it fills the vector of observations based on the input sequence. Although the VF procedures have finite-sample statistical validity, they may not use all available observations at the time of comparison and may also add complexity in implementation as one needs to track the input order. Another issue with the VF procedures is that they may consume a large amount of memory to store the vector for R&S problems having a large number of alternatives. Even though the problem can be partly relieved by using a more effective memory management scheme, we may still encounter out-of-memory errors in some implementations. If one is content with asymptotic validity, we also design an asymptotic parallel selection (APS) procedure that allows unequal sample sizes for all alternatives and makes elimination decisions based on all available observations. The APS procedure can be shown to be asymptotically valid as the indifference-zone parameter goes to zero, an asymptotic regime also used by Kim and Nelson (2006b).

Our work is related to three streams of simulation literature. The first is the literature on R&S. In this paper we take a frequentist's view and consider the indifference-zone (IZ) formulation of the problem. The IZ formulation was first proposed by Bechhofer (1954) and related procedures are summarized in Bechhofer et al. (1995) and Kim and Nelson (2006b). There are also many Bayesian formulations and procedures for R&S problems. For instance, Chen et al. (2000) and Chick and Inoue (2001a,b) allocate a finite number of samples to different alternatives to maximize the posterior probability of correct selection. Instead of considering only the statistical measure of probability of correct selection, Chick and Gans (2009) and Chick and Frazier (2012) take sampling cost into account and formulate the R&S problems using dynamic programming techniques. A comprehensive comparison of the performance among different R&S procedures (designed under either a frequentist or a Bayesian formulation) has been conducted by Branke et al. (2007) in which they conclude that no R&S procedure can dominate in all situations.

The second stream of literature is on parallel and distributed simulation (PADS). According to Heidelberger (1988), PADS has two different approaches to parallelizing the simulation experiments: the first one is that each processor simulates multiple independent replications and the other one is that multiple processors cooperate on a single realization or replication. There is a vast literature on PADS from the 1980s and 1990s, where the focus was on the synchronization issues related to correct ordering of events in discrete-event simulations (see, for instance, Misra (1986) and Fujimoto (1990)). Recently, cloud computing has also been applied to handle PADS (Fujimoto et al. (2010)).

The third stream of literature is on simulation output analysis in a parallel and distributed simulation environment. Heidelberger (1988) discusses a variety of statistical properties for sample

mean estimators calculated by observations from terminating simulations in a parallel simulation environment under three different stopping rules. Glynn and Heidelberger (1991) further study mean performance estimators for both terminating simulations and steady-state regenerative simulations under a completion-time constraint. Recently, Hsieh and Glynn (2009) proposed two new estimators for steady state simulations by weighting the sample average across replications on multiple processors according to some model selection criterion. To the best of our knowledge, there are only three papers using parallel and distributed simulation to solve R&S problems. The first is by Yücesan et al. (2001), who implement an optimal computing budget allocation (OCBA) algorithm in a web-based parallel environment to select the best alternative based on a Bayesian approach. The second is by Chen (2005), who applied a multi-stage R&S procedure with the simulation tasks of each stage distributed to multiple processors. Both papers test their procedures using only small-scale problems (both with only 10 alternatives), so it is not clear whether their procedures are capable of handling large-scale R&S problems. The third is by Ni et al. (2013), who proposed a "zipping" method to solve large-scale R&S problems in a high performance computing environment. The basic idea of their "zipping" method is to retrieve the independent and identically distributed (i.i.d.) sequence by controlling the seeds of the random number generator for each alternative, which is similar to the idea of our VF procedures.

As one of the pioneering works in handling R&S problems in parallel computing environments, we would like to highlight three main contributions of this paper. First, we demonstrate that large-scale R&S problems can be solved efficiently in parallel computing environments. Therefore, using parallel computing environments is a viable solution when there are a large number of alternatives. Second, we show that naive implementations of existing sequential R&S procedures in parallel computing environments may cause unexpected statistical issues that make these procedures inefficient or even invalid. To circumvent these issues, we propose the VF procedures that preserve the original statistical guarantees by carefully managing the output sequence of the simulation replications. Third, we propose the APS procedure that does not require active managing of the output sequence but is asymptotically valid. The APS procedure is simple to implement and the numerical study shows that it works well for the test problems.

The remainder of this paper is organized as follows: In Section 2, we use a queueing analogy to illustrate differences between using a single processor and using multiple processors to solve R&S problems. Based on the properties we identify in Section 2, we then propose two general approaches to designing R&S procedures, namely the VF procedures and the APS procedures, in Sections 3 and 4. In Section 4, we also show the statistical validity of the APS procedure in a meaningful asymptotic regime. Numerical implementation of these two procedures as well as the numerical results are shown in Section 5, followed by some concluding remarks in Section 6.

2. The Randomness of Output Sequence

Suppose there are k alternatives whose mean performance can be evaluated by simulation on m processors. Let $X_{i\ell}$ denote the ℓth observation from alternative i, and we assume that $X_{i\ell}$, $\ell = 1, 2, \ldots$, are i.i.d. random variables with a finite mean μ_i for all $i = 1, 2, \ldots, k$; and that $X_{i\ell}$ and X_{jd} are mutually independent for all ℓ, d and $i \neq j$. Under the IZ formulation, we further assume that $\mu_1 - \delta \geq \mu_2 \geq \ldots \geq \mu_k$, where δ is the IZ parameter, and our goal is to design fully sequential procedures that are statistically valid, can select alternative 1 as the best with a probability at least $1 - \alpha$ and that can be implemented on multi-core personal computers or many-core servers (with a total of $m > 1$ processors). For mathematical simplification, in this paper we assume that the m processors are identical in their processing speeds, and that the time for loading simulation programs into processors and the time for transmitting data among the processors is negligible.

2.1. Queueing Analogy

To better understand the difference between implementation of fully sequential procedures on a single processor (i.e., $m = 1$) and on multiple processors (i.e., $m > 1$), we describe the simulation process using a queueing model analogy. In this analogy, observations from alternative i are represented by class i customers, and m identical processors are represented by a server pool with m homogeneous servers. There is no arrival process in this queueing model, instead, all customers are waiting in the queue with a predetermined order at the beginning of the simulation process, and this predetermined order of customers is called the *input sequence*. When the simulation starts, the first m customers are assigned to the m servers. Once a server finishes the service of its current customer (i.e., generating the observation), the first customer waiting in queue will be immediately routed to that server. The departure process captures the order of customers who have finished service (i.e., the observations), and the order of departing customers is called the *output sequence*. When implementing fully sequential selection procedures, we perform comparisons and eliminations among the surviving alternatives based on the observations in the output sequence. When an alternative i is to be eliminated, the class i customers in the input sequence will abandon the queue and therefore will not be simulated.

For a fully sequential procedure implemented on a single processor, it is worthwhile noting that its input and output sequences are always the same. However, when the procedure is implemented on multiple processors, the output sequence may be different from the input sequence, because the simulation times of different alternatives may be different, and the output sequence may even be non-deterministic because the simulation times of alternatives may be stochastic. See Figure 2 for an illustration. Because the output sequence may be different from the input sequence, we

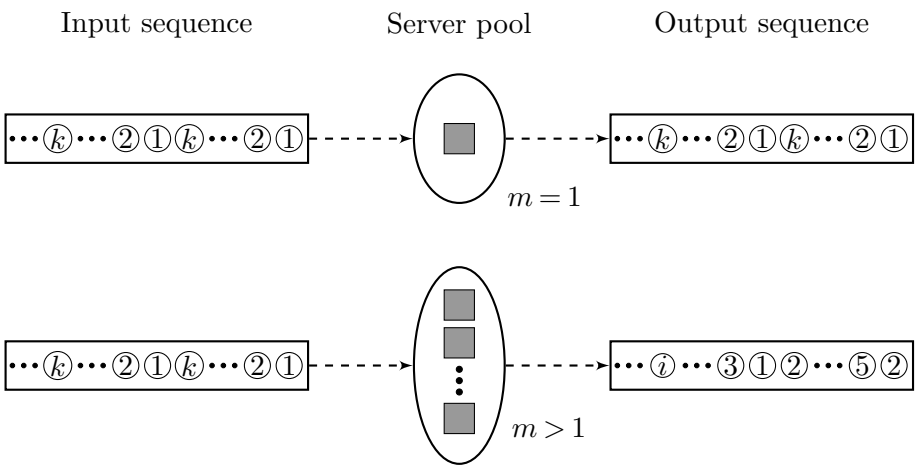


Figure 2 An illustration using queueing models.

define Y_{ij} as the jth observation of alternative i in the output sequence, in addition to $X_{i\ell}$ which represents the ℓth observation of alternative i in the input sequence. Notice that, when simulation is conducted on a single processor, $X_{i\ell} = Y_{i\ell}$. When simulation is conducted on multiple processors, however, it is possible that $X_{i\ell} \neq Y_{i\ell}$.

Let $\Gamma_{i\ell}$ denote the (random) amount of time it takes to run $X_{i\ell}$, the ℓth replication of alternative i in the input sequence. In the queueing analogy, $\Gamma_{i\ell}$ is the service time of the ℓth customer of class i in the queue. We assume that $\Gamma_{i\ell} > 0$ almost surely (a.s.) and it has a finite mean $\gamma_i > 0$. Then, $\{(X_{i\ell}, \Gamma_{i\ell}), \ell = 1, 2, \ldots\}$ is a sequence of i.i.d. bivariate random vectors. However, as the comparisons and elimination decisions for a sequential procedure are made based on the output sequence $\{Y_{i\ell}, \ell = 1, 2, \ldots\}$, it is critical to understand the statistical properties of $\{Y_{i\ell}, \ell = 1, 2, \ldots\}$. In the remainder of this section, we show that $\{Y_{i\ell}, \ell = 1, 2, \ldots\}$ may not be an i.i.d. sequence and it may compromise the statistical validity of existing fully sequential selection procedures.

2.2. Random Sample Sizes

When implementing a fully sequential selection procedure, one needs to specify the input sequence, which we call the sample allocation rule (SAR). SARs describe how observations from different alternatives are repeated in the input sequence. The most straightforward SAR is the round-robin rule that takes one observation from each surviving alternative in a predetermined order (say alternatives $1, 2, \ldots, k$); it is used in the $\mathcal{KN}$ procedures of Kim and Nelson (2001, 2006a). For simplicity of presentation, we only consider the round-robin SAR in this paper.

Let $t \geq 0$ denote the run time from the start of the procedure and $N_i(t)$ denote the number of completed observations of alternative i by time t for all $i = 1, 2, \ldots, k$. Then, $\{N_i(t), t \geq 0\}$ are continuous-time stochastic processes for all $i = 1, 2, \ldots, k$ and $\{N_i(t), t \geq 0\}$ and $\{N_j(t), t \geq 0\}$

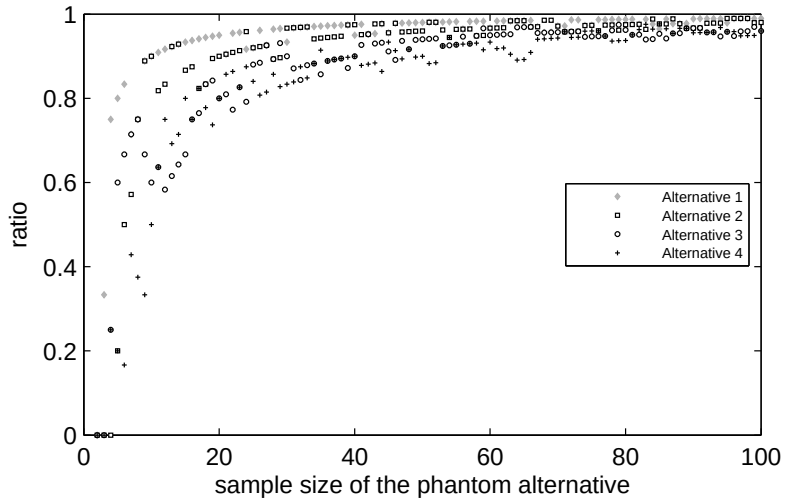


Figure 3 Ratio of the sample sizes of alternative i, $i = 1, 2, 3, 4$, and phantom alternative p when the number of processors is $m = 8$ and the number of alternatives is $k = 4$.

are typically dependent. To better understand $\{N_i(t), t \geq 0\}$ and its characteristics under single processor and multiple processors, we create a new *phantom alternative* and call it alternative p. In the input sequence, alternative p is queued at the end of each round-robin cycle. For instance, when the alternatives are simulated in the order of alternative 1 to k, alternative p is queued right after every alternative k. Furthermore, alternative p has a simulation time of 0 and its observations are not compared to other alternatives. Therefore, it is clear that alternative p does not affect the implementation of the procedure and this is why it is called a phantom alternative.

Let $N_p(t)$ denote the number of observations of the phantom alternative in the output sequence by time t. Let $t_r = \inf\{t \geq 0 : N_p(t) = r\}$ for all $r = 1, 2, \ldots$, which is the time that the rth observation from the phantom alternative is obtained, and let $N_{ir} = N_i(t_r)$ be the number of completed observations of alternative i at time t_r for all $i = 1, 2, \ldots, k$ and $r = 1, 2, \ldots$. Then, we convert the continuous-time process $\{N_i(t), t \geq 0\}$ into a discrete-time process $\{N_{ir}, r = 1, 2, \ldots\}$ for all $i = 1, 2, \ldots, k$. Notice that, when $\mathcal{KN}$ is conducted on a single processor, $t_r, r = 1, 2, \ldots$ are the time points at which the alternatives are compared and eliminated and $N_{ir} = N_{pr} = r$, i.e., $N_{ir}/N_{pr} = 1$ for all $r = 1, 2, \ldots$ and for any surviving alternative i. Therefore, the discrete-time processes $\{N_{ir}, r = 1, 2, \ldots\}$ are deterministic for all surviving alternatives, which makes the statistical validity of $\mathcal{KN}$ easier to analyze.

When there are multiple processors, however, the discrete-time process $\{N_{ir}, r = 1, 2, \ldots\}$ becomes a stochastic process and N_{ir} is typically different from N_{jr} when both alternatives i and j are surviving, i.e., $N_{ir}/N_{pr} \neq N_{jr}/N_{pr}$. To illustrate this, we simulate 4 alternatives with 8 processors using a round-robin rule, where the time to generate an observation of alternatives i follows

an exponential distribution with mean i units of time, and we plot N_{ir}/N_{pr} for all $i=1,2,3,4$ and $r=1,2,\ldots,100$ in Figure 3. From the figure, we see clearly that N_{ir}/N_{pr} are random and typically different from 1, but they appear to converge to 1 as r increases.

2.3. Loss of I.I.D. Property

For each alternative i, even though $\{X_{i\ell}, \ell=1,2,\ldots\}$ is an input sequence of i.i.d. random variables, the output sequence $\{Y_{i\ell}, \ell=1,2,\ldots\}$ may no longer be i.i.d. when $X_{i\ell}$ and $\Gamma_{i\ell}$ are correlated. As a result, the sample mean estimator $\bar{Y}_i(n)=n^{-1}\sum_{\ell=1}^{n} Y_{i\ell}$ calculated using the first n observations of alternative i in the output sequence, is often biased and is in general difficult to analyze.

We use a simple example from Heidelberger (1988) as an illustration. Suppose there is only one alternative, alternative 1, to be simulated and $X_{1\ell}=\Gamma_{1\ell}$ follows an exponential distribution with mean μ_1. Then Heidelberger (1988) shows that the first observation from the output sequence, Y_{11}, is the shortest of m i.i.d. exponential random variables, that is, $Y_{11}=\min\{X_{11},X_{12},\ldots,X_{1m}\}$ which is exponentially distributed with mean $\mu=\mu_1/m$. In Appendix EC.1.1, we derive closed-form expressions for $Y_{1\ell}$, $\ell=1,2,\ldots$. Based on these closed-form expressions, we can easily verify that $Y_{1\ell}$, $\ell=1,2,\ldots$, are not i.i.d. The mean of $Y_{1\ell}$ is

$$\mathbb{E}[Y_{1\ell}]=\mu_1\left[1-\left(1-\frac{1}{m}\right)^{\ell}\right],$$

and the expectation of the sample mean estimator is

$$\mathbb{E}\left[\bar{Y}_1(n)\right] \;=\; \mu_1\left\{1-\frac{m-1}{n}\left[1-\left(1-\frac{1}{m}\right)^{n}\right]\right\},$$

which means $\bar{Y}_1(n)$ has a downward bias. However, the bias goes to 0 as the sample size $n\to\infty$. Furthermore, the moment generating function (MGF) of Y_{1i} is

$$M_{Y_{1\ell}}(t)=\frac{1}{1-m\mu t}-\frac{m\mu t}{1-m\mu t}\left(\frac{m-1}{m}\cdot\frac{1}{1-\mu t}\right)^{\ell},$$

where t is in a sufficiently small neighborhood of 0. Notice that $\lim_{\ell\to\infty} M_{Y_{1\ell}}(t)=1/(1-m\mu t)$, which is exactly the MGF of an exponential random variable with mean $m\mu=\mu_1$ (seeing Appendix EC.1.1 for detailed derivations).

Furthermore, let $M_{Y_{1,\ell+n}}(t)$ and $M_{Y_{1\ell}+Y_{1,\ell+n}}(t)$ denote the MGFs of $Y_{1,\ell+n}$ and $Y_{1\ell}+Y_{1,\ell+n}$, respectively, for any $\ell=1,2,\ldots$ and $n=1,2,\ldots$ In Appendix EC.1.1, we also show that

$$\lim_{n\to\infty}\left\{M_{Y_{1\ell}}(t)\cdot M_{Y_{1,\ell+n}}(t)-M_{Y_{1\ell}+Y_{1,\ell+n}}(t)\right\}=0.$$

Therefore, the dependence between $Y_{1\ell}$ and $Y_{1,\ell+n}$ also vanishes as $n\to\infty$. In this sense, as $n\to\infty$, the output sequence may be viewed as i.i.d. and statistically equivalent to the input sequence.

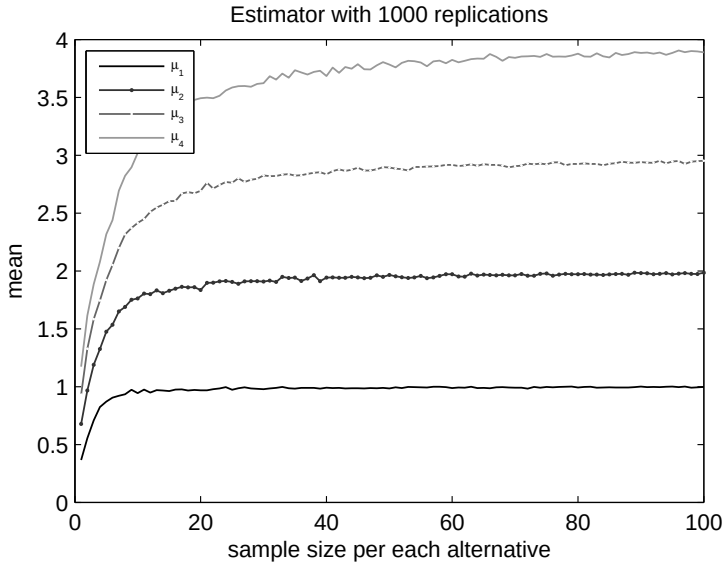


Figure 4 The sample mean estimators with 1000 sample paths when the number of processors is $m = 8$ and the number of alternatives is $k = 4$.

When there are multiple alternatives, however, the system dynamics are much more complicated, and we are not able to derive closed-form expressions for the distribution of $Y_{i\ell}$. Nevertheless, it is still quite clear that $Y_{i\ell}, \ell = 1, 2, \ldots$ are no longer i.i.d. observations, and that $\{Y_{i\ell}, \ell = 1, 2, \ldots\}$ and $\{Y_{j\ell}, \ell = 1, 2, \ldots\}$ are likely dependent on each other. As an example, we simulate 4 alternatives on 8 processors where $X_{i\ell} = \Gamma_{i\ell}$ follows an exponential distribution with mean i time units. In Figure 4, we plot $\mathbb{E}\left[\bar{Y}_i(n)\right]$, $i = 1, 2, 3, 4$, with the sample size n varying from 1 to 100, estimated with from 1,000 macroreplications.

2.4. Dependence Caused by Eliminations

If $X_{i\ell}$ is independent of $\Gamma_{i\ell}$ (or even if $\Gamma_{i\ell}$ is constant) for all $i = 1, 2, \ldots, k$, $\bar{Y}_i(n)$ becomes an unbiased estimator of μ_i. However, the elimination decisions inherent to sequential selection procedures may still introduce dependence among the sample sizes of surviving alternatives, and thus introduce dependence among their sample means. To illustrate this type of dependence, suppose there are three alternatives to be simulated on two processors and the replication times of alternatives $1, 2, 3$ are fixed as $2, 1, 1$ time units, respectively. Furthermore, suppose that the input sequence is in a round-robin order of 1, 2 and 3. Then the simulation process can be described as in Figure 5. At each time point t_r (which corresponds to the completion time of the rth phantom alternative), we conduct comparisons among all surviving alternatives. Notice that, when all three alternatives are surviving, they all have an equal sample size $N_1(t_r) = N_2(t_r) = N_3(t_r) = r$ at t_r for all $r = 1, 2, \ldots$. Suppose at some time point t_n, alternative 2 is eliminated. Then, at the next time point t_{n+1},

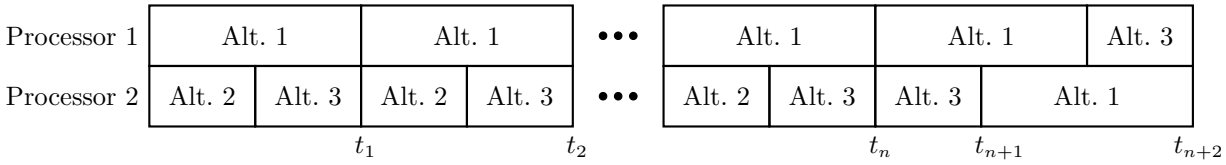


Figure 5 Three alternatives on two processors with constant replication times.

$N_1(t_{n+1}) = n$ but $N_3(t_{n+1}) = n + 1$. Therefore, sample sizes of surviving alternatives depend on elimination decisions which depend on sample means of all alternatives. This type of dependence may cause the sample means of the surviving alternatives to be dependent on each other.

When there are a large number of alternatives with random replication times simulated on many processors, the dynamics of elimination decisions can be more complicated, leading to more complicated dependence among the sample sizes and sample means of surviving alternatives. It is worthwhile noting that this problem is caused by the use of multiple processors. When a fully sequential procedure is implemented on a single processor, eliminations do not cause dependence because the output sequence of surviving alternatives remains the same as that without elimination.

In Sections 2.2–2.4, we have shown that the use of multiple processors may create various statistical issues when the observations in the output sequence are used to implement sequential selection procedures. To solve the problem, we take two different approaches and discuss them in the next two sections. In the first approach, we implement sequential procedures using the observations in the output sequence based on their order in the input sequence. Therefore, the finite-time statistical validity of these procedures may be guaranteed. However, this approach requires significant accounting and often a large amount of memory for storing the observations, and it may use only a portion of the observations in the output sequence (thus may not be efficient). Therefore, we propose another approach that designs sequential selection procedures that are asymptotically valid. This is possible because, as shown in Sections 2.2–2.4, the statistical properties of the sample-mean estimators $\bar{Y}_i(n)$ tend to behave nicely as the sample sizes go to infinity.

3. Vector Filling Procedures

As mentioned above, if we restrict our attention to the use of observations exactly according to the predetermined order in the input sequence, then all existing fully sequential selection procedures are statistically valid when implemented in a parallel computing environment. To achieve this goal, we may create a vector to record the observations in the same order of the input sequence and place phantom alternative p in the positions where elimination decisions are scheduled. Then we can conduct comparison and elimination decisions when all observations from surviving alternatives

ahead of every position for the phantom alternative have been collected in the vector. We call this type of procedures a *vector-filling procedure*. In this section, we provide a simple vector-filling procedure which extends the well-known $\mathcal{KN}$ to a parallel computing environment.

To simplify the presentation, we suppose there are $m+1$ parallel processors (or threads), which we call processors $0, 1, \ldots, m$. Processor 0 is used to manage the input sequence and to conduct comparisons and eliminations, while processors $1, 2, \ldots, m$ are used to simulate the alternatives.

Procedure 1 (Vector-Filling $\mathcal{KN}$ Procedure)

Step 0. **Setup:** Select confidence level $1/k < 1-\alpha < 1$, IZ parameter $\delta > 0$, and first-stage sample size $n_0 \geq 2$. Let $h^2 = (n_0-1)\left[\left(\frac{2\alpha}{k-1}\right)^{-2/(n_0-1)} - 1\right]$.

Step 1. **Initialization:** Let $I = \{1, 2, \ldots, k\}$ be the set of alternatives still in contention. Processor 0 manages the input sequence in which all alternatives are stored in a round-robin order from 1 to k. Processor 0 also performs the tasks listed in the following Steps 2–4, including conducting pairwise comparisons and eliminations. The first m replications in the input sequence are assigned to the remaining m processors, processors $1, 2, \ldots, m$, to be simulated. Processors $1, 2, \ldots, m$ work as follows: take the first alternative queued in the input sequence, generate an observation, and submit the result to processor 0.

The ℓth replication from alternative i in the input sequence is denoted as $X_{i\ell}$. These available observations are stored in a vector in the same order as the input sequence. Let $n_a = \max\{n \geq 0 : X_{i\ell} \text{ is available for all } \ell \leq n \text{ and for } i \in I\}$. Notice that $n_a = 0$ at the beginning of simulation. Start the simulation.

Step 2. **Variance Estimation:** When $n_a \geq n_0$, compute the sample variance of the difference between alternatives $i \neq j$,

$$S_{ij}^2 = \frac{1}{n_0 - 1} \sum_{\ell=1}^{n_0} \left(X_{i\ell} - X_{j\ell} - [\bar{X}_i(n_0) - \bar{X}_j(n_0)]\right)^2,$$

where $\bar{X}_\ell(n_0)$ is the first-stage sample mean of alternative ℓ with n_0 observations. Set $r = n_0$.

Step 3. **Elimination:** Set $I^{\text{old}} = I$. Let

$$I = I^{\text{old}} \setminus \left\{ i \in I^{\text{old}} : \bar{X}_i(r) - \bar{X}_j(r) < \min\left\{0, -\frac{h^2 S_{ij}^2}{2r\delta} + \frac{\delta}{2}\right\} \text{ for some } j \in I^{\text{old}}, j \neq i \right\},$$

where $A \setminus B = \{x : x \in A \text{ and } x \notin B\}$, and remove alternative i from the input sequence for all $i \in I^{\text{old}} \setminus I$.

Step 4. **Stopping Rule:** If $|I| = 1$, then stop all processors and select the alternative whose index is in I as the best. Otherwise, processor 0 checks whether it is ready for the next elimination. Let $r = r + 1$.

(a). If $r \leq n_a$, go to Step 3;

(b). Otherwise, wait for a new observation from any alternative j in I, say $X_{j\ell}$, record $X_{j\ell}$ in the vector, update n_a, and go to (a).

REMARK 1. The statistical validity of the vector-filling $\mathcal{KN}$ (VKN) procedure is the same as that of $\mathcal{KN}$ because, statistically, the two procedures are identical in conducting comparisons and making elimination decisions.

REMARK 2. The VKN procedure may require a large amount of memory to store simulation outputs exactly following the order in the input sequence, especially when the R&S problem has a large number of alternatives and the variances of replication times are high. For instance, when we apply the VKN procedure to solve a test problem with 10^4 alternatives on a personal computer with 4.00 GB RAM (see Section 5 for detailed settings), we encountered situations where the procedure has to be terminated after using up all of the memory. One may design careful accounting schemes that would dramatically reduce the memory requirements, e.g., by storing cumulative sums up to the point where all replications in the input sequence have returned.

4. Asymptotic Parallel Selection Procedures

The nice asymptotic properties of the sample-mean estimators $\bar{Y}_i(n)$ in Section 2 motivate us to design fully sequential selection procedures that are asymptotically valid and are computationally more efficient than VF procedures. Our goal is to design a simple and easily executable fully sequential procedure that uses all simulation observations in the output sequence, allows different surviving alternatives to have different sample sizes, and has a provable asymptotic validity in a meaningful asymptotic regime.

To design such a procedure the key is to decide when to compare surviving alternatives and make elimination decisions. For that we introduce the concept of a *phantom alternative*, which is an alternative that does not need to be simulated (i.e., simulation time is zero) and compared, and is used only for counting purposes. We add the phantom alternative after each round-robin cycle in the input sequence and then start simulation. Whenever the phantom alternative completes (i.e., appears in the output sequence), we compare all surviving alternatives using all their available observations (in the output sequence) and make elimination decisions. Notice that, when there is only a single processor, the phantom alternative completes at the moment that the last surviving alternative in the round-robin cycle completes. Then, comparing and eliminating according to the phantom alternative is exactly the same as what the $\mathcal{KN}$ procedure does. When there are multiple processors, different surviving alternatives may have different sample sizes at the moment that a phantom alternative completes. However, the difference between the sample size of a surviving

alternative and that of the phantom alternative is always bounded by the number of processors. Therefore, the difference between the sample means of a surviving alternative computed based on input and output sequences may vanish as the sample size of the alternative goes to infinity. This provides a key to insuring the asymptotic validity of the procedure. Therefore, the completion times of the phantom alternative serves as a drumbeat process that synchronizes the comparisons and eliminations and insures the asymptotic validity.

Procedure 2 (Asymptotic Parallel Selection (APS) Procedure)

Step 0. **Setup:** Select confidence level $1/k < 1-\alpha < 1$, IZ parameter $\delta > 0$, and first-stage sample size $n_0 \geq 2$. Let $a = -\log[2\alpha/(k-1)]$.

Step 1. **Initialization:** Let $I = \{1, 2, \ldots, k\}$ be the set of alternatives still in contention. Processor 0 manages the input sequence in which all alternatives are stored in a round-robin order from 1 to k. Processor 0 also performs the tasks listed in the following Steps 2–4, including conducting pairwise comparisons and eliminations. The first m replications in the input sequence are assigned to the remaining m processors, processors $1, 2, \ldots, m$, to be simulated. Processors $1, 2, \ldots, m$ work as follows: take the first alternative queued in the input sequence, generate an observation, and submit the result to processor 0.

Add a phantom alternative p after each round-robin cycle in the input sequence (but not the set I). Let r denote the stage which is the current sample size of the phantom alternative in the output sequence. Let $Y_{i\ell}$ denote the ℓth completed observation from alternative i in the output sequence, and let N_{ir} denote the number of completed observations from alternative i in the output sequence at the time when the rth observation of the phantom alternative is added to the output sequence. Record the triple $\left(N_{ir}, \sum_{\ell=1}^{N_{ir}} Y_{i\ell}, \sum_{\ell=1}^{N_{ir}} Y_{i\ell}^2\right)$ for all $i \in I$.

Step 2. **Collecting Initial Observations:** Start simulations on processors $1, 2, \ldots, m$ and wait until $r = n_0$.

Step 3. **Elimination:** For all $i \in I$, let

$$\bar{Y}_i(N_{ir}) = \frac{1}{N_{ir}} \sum_{\ell=1}^{N_{ir}} Y_{i\ell},$$

$$S_i^2(N_{ir}) = \frac{1}{N_{ir}-1} \left[\sum_{\ell=1}^{N_{ir}} Y_{i\ell}^2 - \frac{1}{N_{ir}} \left(\sum_{\ell=1}^{N_{ir}} Y_{i\ell} \right)^2 \right].$$

For all $i, j \in I$ and $i \neq j$, if $N_{ir} \geq n_0$ and $N_{jr} \geq n_0$, let

$$\tau_{ij,r} = \left[\frac{S_i^2(N_{ir})}{N_{ir}} + \frac{S_j^2(N_{jr})}{N_{jr}} \right]^{-1};$$

otherwise, let $\tau_{ij,r} = 0$. This ensures that the comparisons are done between alternatives that have at least n_0 observations. Let $I^{\text{old}} = I$ and let

$$I = I \setminus \left\{ i \in I^{\text{old}} : \tau_{ij,r} \left[\bar{Y}_i(N_{ir}) - \bar{Y}_j(N_{jr}) \right] < \min \left\{ 0, -\frac{a}{\delta} + \frac{\delta}{2} \tau_{ij,r} \right\} \text{ for some } j \in I^{\text{old}} \text{ and } j \neq i \right\}.$$

Remove alternative i from the input sequence for all $i \in I^{\text{old}} \setminus I$.

Step 4. **Stopping Rule:** If $|I| = 1$, then stop all processors and select the alternative whose index is in I as the best. Otherwise, wait for a new observation. If the new observation is from any alternative $i \in I$, then update $\left(N_{ir}, \sum_{\ell=1}^{N_{ir}} Y_{i\ell}, \sum_{\ell=1}^{N_{ir}} Y_{i\ell}^2 \right)$ and wait for the next observation; if the observation is from alternative p, then update $r = r + 1$ and go to Step 3.

REMARK 3. In the APS procedure, we keep updating the sample variances for the surviving alternatives, which is similar to $\mathcal{KN}$++ of Kim and Nelson (2006a) designed for R&S problems in steady-state simulations. In order to show the asymptotic validity of sample-variance updating, we need some technical condition on the first-stage sample size n_0, which is stated in Theorem 1. While the theoretical condition facilitates the asymptotic proof, it does not prescribe a specific choice of first-stage sample size in practice.

Notice that, in the APS procedure, we only make elimination decisions when a phantom alternative completes, but we use all available observations at that time. Therefore, the APS procedure has several advantages when compared to the VKN procedure. First, it makes use of all available observations, which leads to a higher efficiency, especially when the simulation effort for generating each observation is substantial (e.g., the replication time is relatively long). Second, it requires significantly less memory to store the observations, which makes it feasible to solve large-scale R&S problems.

The following theorem establishes the asymptotic validity of the APS procedure.

THEOREM 1. *Let $X_{i\ell}$ denote the ℓth replication from alternative i in the input sequence and $\Gamma_{i\ell}$ denote the replication time to generate the observation $X_{i\ell}$, with unknown means $\mu_i = \mathbb{E}[X_{i\ell}]$ and unknown (but finite) variance $\sigma_i^2 = \text{Var}[\text{X}_{\text{i}\ell}]$, for all $i = 1, 2, \ldots, k$ and $\ell = 1, 2, \ldots$. Assume that $X_{i\ell}$ and X_{jn} are independent of each other when $i \neq j$ or $\ell \neq n$; that $\Gamma_{i\ell} > 0$ a.s. for all i and ℓ; and that $\mu_1 - \delta \geq \mu_2 \geq \ldots \geq \mu_k$, where δ is the IZ parameter. Moreover, let the first-stage sample size $n_0 = n_0(\delta)$ be a function of δ such that $n_0 \to \infty$ and $\delta^2 n_0 \to 0$ as $\delta \to 0$. Then, as $\delta \to 0$, the APS procedure selects alternative 1 as the best with a probability at least $1 - \alpha$.*

Notice that the asymptotic regime of $\delta \to 0$ (as well as the true difference between the best and the second-best alternatives $\mu_1 - \mu_2 \to 0$) is also used by Kim and Nelson (2006a) in analyzing the $\mathcal{KN}$++ procedure, where the observations are taken from steady-state simulations which are

stationary but not independent. In Theorem 1, however, we assume that the observations ordered by the input sequence $\{X_{i\ell}, \ell = 1, 2, \ldots\}$ are i.i.d. but, as shown in Section 2.3, the observations ordered based on the output sequence $\{Y_{i\ell}, \ell = 1, 2, \ldots\}$ may not be.

We prove Theorem 1 in the next two subsections.

4.1. Brownian Motion Construction

For a clear presentation, in the remainder of this section we consider only the situation where the difference between the mean of the best and all other alternatives equals the IZ parameter δ, i.e., $\mu_2 = \mu_3 = \cdots = \mu_k = \mu_1 - \delta$, which is called the slippage configuration (SC) in the R&S literature.

Consider any pair of alternatives, alternatives i and j. Let $N_{ij}^{\delta} = \lceil 2a(\sigma_i^2 + \sigma_j^2)/\delta^2 \rceil$, where $a = -\log[2\alpha/(k-1)]$ is defined in Step 0 of the APS procedure and $\lceil x \rceil$ denotes the smallest integer not less than x. In Section 4.2, it will be clear that N_{ij}^{δ} is the maximum number of observations needed from either alternative i or j when comparing alternatives i and j. Let s be any number in $[0, 1]$ and let $r = \lfloor sN_{ij}^{\delta} \rfloor$, where $\lfloor x \rfloor$ denotes the largest integer not greater than x. Define

$$Z_{ij}(s) = \begin{cases} \frac{\sigma_i^2/r + \sigma_j^2/r}{S_i^2(N_{ir})/N_{ir} + S_j^2(N_{jr})/N_{jr}} \cdot s\sqrt{\frac{N_{ij}^{\delta}}{\sigma_i^2+\sigma_j^2}} \left[\bar{Y}_i(N_{ir}) - \bar{Y}_j(N_{jr})\right], & s \in [n_0/N_{ij}^{\delta}, 1] \\ 0, & s \in [0, n_0/N_{ij}^{\delta}) \end{cases} \tag{1}$$

where $N_{\ell r}$ is the sample size of alternative ℓ, $\ell = 1, \ldots, k$, when the sample size of the phantom alternative p is $N_{pr} = r$. To make $Z_{ij}(\cdot)$ well-defined, we set $Z_{ij}(\cdot) = 0$ when either $N_{ir} = 0$ or $N_{jr} = 0$. In fact, we are only interested in the case that $r \geq n_0$, i.e., $s \in [n_0/N_{ij}^{\delta}, 1]$, because that the APS procedure starts eliminating systems at least n_0 samples. For mathematical completeness in the neighbourhood of $s = 0$, we can artificially set $N_{\ell r} = r$, $S_{\ell}^2(N_{\ell r}) = \sigma_{\ell}^2$ and $\bar{Y}_{\ell}(N_{\ell r}) = 0$, $\ell = i$ or j, in order to make sure that $Z_{ij}(s) = 0$ for $s \in [0, n_0/N_{ij}^{\delta})$. However, without additional specifications, all statements about $Z_{ij}(s)$ will refer to the process for $s \in [n_0/N_{ij}^{\delta}, 1]$.

Hong (2006) shows that $Z_{ij}(s)$ with the random sample size $N_{\ell r}$ replaced by the deterministic n_{ℓ}, the sample variance $S_{\ell}^2(N_{\ell r})$ replaced by σ_{ℓ}^2, and $\bar{Y}_{\ell}(N_{\ell r})$ replaced by $\bar{X}_{\ell}(n_{\ell})$, $\ell = i$ or j, has the same distribution as a Brownian motion when the $X_{\ell n}$'s are normally distributed. This result motivates the definition of $Z_{ij}(s)$ in Equation (1). The following lemma shows that $Z_{1j}(\cdot)$ converges to a Brownian motion process for all $j = 2, 3, \ldots, k$.

LEMMA 1 **(Convergence to a Brownian Motion Process)**. *Let $\mathbb{D}[0, 1]$ be the Skorohod space of all right-continuous real-valued functions on $[0, 1]$ with limits from the left everywhere, endowed with the Skorohod J_1 topology (see Appendix EC.1.3 for the definition of the standard J_1 metric and Section 3.3 in Whitt (2002) for more background of the space $\mathbb{D}$). Thus, $Z_{1j}(\cdot)$ defined by Equation*

(1) *with* $j = 2, 3, \ldots, k$ *is an element of the Skorohod space* $\mathbb{D}[0,1]$. *Suppose that the conditions in Theorem 1 are all satisfied. Then, under the SC, i.e.,* $\mu_2 = \mu_3 = \cdots = \mu_k = \mu_1 - \delta$, *we have*

$$Z_{1j}(\cdot) \Rightarrow \mathcal{B}_\Delta(\cdot), \quad as\ \delta \to 0,$$

where $\mathcal{B}_\Delta(t) = \mathcal{B}(t) + \Delta t$, *a standard Brownian motion process with a constant drift* $\Delta = \sqrt{2a}$.

Proof of Lemma 1: Let $\epsilon^\delta = n_0/N_{1j}^\delta$, where $N_{1j}^\delta = \lceil 2a(\sigma_1^2 + \sigma_j^2)/\delta^2 \rceil$. Recall that $n_0 \to \infty$ and $\delta^2 n_0 \to 0$ as $\delta \to 0$. Then, $N_{1j}^\delta \to \infty$ and $\epsilon^\delta \to 0$ as $\delta \to 0$. Notice that $Z_{1j}(s) = 0$ for $0 \le s < \epsilon^\delta$, which implies that

$$Z_{1j}(0) = \mathcal{B}_\Delta(0) = 0 \text{ and } Z_{1j}(\cdot) \text{ is right-continuous at } s = 0 \text{ for all } \delta. \tag{2}$$

We next focus only on $s \in [\epsilon^\delta, 1]$, that is, $r \in [n_0, N_{1j}^\delta]$.

We start by analyzing the first term on the right-hand side (RHS) of Equation (1). Recall that, at stage r, for alternative ℓ ($\ell = 1$ or j), totally r replications have been sent to the m processors with $N_{\ell r}$ replications completed and $r - N_{\ell r}$ still in simulation. Then, we have $r - m \le N_{\ell r} \le r$ for all $n_0 \le r \le N_{1j}^\delta$, which implies that, w.p.1,

$$\sup_{s \in [\epsilon^\delta, 1]} \left| \frac{N_{\ell r}}{r} - 1 \right| = \sup_{r \in [n_0, N_{1j}^\delta]} \left| \frac{N_{\ell r} - r}{r} \right| \le \frac{m}{n_0} \to 0,$$

as $\delta \to 0$. In other words, as $\delta \to 0$, $n_0 \to \infty$, so that $r \to \infty$ and $N_{\ell r}/r \to 1$ w.p.1 as functions of s on $(0,1]$. Notice that this result can be easily extended to the closed space $[0,1]$ given the definition that $N_{\ell r} = r$ for $s \in [0, \epsilon^\delta)$. In fact, the pointwise convergence guarantees the uniform convergence on $[0,1]$ under the definition that $N_{\ell r} = r$ on $[0, \epsilon^\delta)$ and the condition that $\epsilon^\delta \to 0$ and $\epsilon^\delta N_{1j}^\delta \to \infty$ as $\delta \to 0$. Similar arguments can also be applied to $S_\ell^2(N_{\ell r})$ and $\bar{Y}_\ell(N_{\ell r})$, $\ell = 1$ or j. Therefore, we will establish uniform convergence by showing pointwise convergence in the following proof.

Let $\Omega_{\ell r} \subset \{1, 2, \ldots, r\}$ be the set of the indices of incomplete replications from alternative ℓ, i.e., if $n \in \Omega_{\ell r}$, $X_{\ell n}$ is still in simulation at stage r. Thus, $|\Omega_{\ell r}| = r - N_{\ell r}$. Notice that, for $\ell = 1$ or j,

$$S_\ell^2(N_{\ell r}) = \frac{1}{N_{\ell r} - 1} \left[\sum_{n=1}^{N_{\ell r}} Y_{\ell n}^2 - \frac{1}{N_{\ell r}} \left(\sum_{n=1}^{N_{\ell r}} Y_{\ell n} \right)^2 \right],$$

which can be further written as

$$\begin{aligned} S_\ell^2(N_{\ell r}) &= \frac{1}{N_{\ell r} - 1} \left[\sum_{n=1}^{r} X_{\ell n}^2 - \sum_{n \in \Omega_{\ell r}} X_{\ell n}^2 - \frac{1}{N_{\ell r}} \left(\sum_{n=1}^{r} X_{\ell n} - \sum_{n \in \Omega_{\ell r}} X_{\ell n} \right)^2 \right] \\ &= \frac{r}{N_{\ell r} - 1} \left[\frac{1}{r} \sum_{n=1}^{r} X_{\ell n}^2 - \left(\frac{1}{r} \sum_{n=1}^{r} X_{\ell n} \right)^2 \right] \\ &\quad + \frac{1}{N_{\ell r} - 1} \left[- \sum_{n \in \Omega_{\ell r}} X_{\ell n}^2 + \frac{(N_{\ell r} - r) r}{N_{\ell r}} \bar{X}_\ell(r)^2 + \frac{2r}{N_{\ell r}} \bar{X}_\ell(r) \sum_{n \in \Omega_{\ell r}} X_{\ell n} - \frac{1}{N_{\ell r}} \left(\sum_{n \in \Omega_{\ell r}} X_{\ell n} \right)^2 \right]. \end{aligned} \tag{3}$$

Similarly, as $\delta \to 0$ (which implies that $n_0 \to \infty$ and $r \to \infty$; for simplicity, we may state only one of them hereafter), then $r/(N_{\ell r}-1) \to 1$ w.p.1. By the strong law of large numbers (SLLN), we know that $r^{-1}\sum_{n=1}^{r} X_{\ell n}^2 \to \mathbb{E}[X_{\ell n}^2] = \sigma_\ell^2 + \mu_\ell^2$, and $\bar{X}_\ell(r) = r^{-1}\sum_{n=1}^{r} X_{\ell n} \to \mu_\ell$ w.p.1. By the continuous mapping theorem (Durrett (2004)), the first term of $S_\ell^2(N_{\ell r})$ in Equation (3) converges to σ_ℓ^2 w.p.1 as $r \to \infty$.

Notice that, for both $\ell = 1$ and j and all $n = 1, 2, \ldots, r$,

$$|X_{\ell n}| \le \max_{b=1,\ldots,r} |X_{\ell b}|.$$

Furthermore, we have $N_{\ell r} \ge r - m$ and $|\Omega_{\ell r}| \le m$. Then, when $n_0 > m+1$, implying that $r > m+1$, the second term of $S_\ell^2(N_{\ell r})$ in Equation (3) can be bounded as follows,

$$\begin{aligned}
&\frac{1}{N_{\ell r}-1}\left[-\sum_{n\in\Omega_{\ell r}} X_{\ell n}^2 + \frac{(N_{\ell r}-r)r}{N_{\ell r}}\bar{X}_\ell(r)^2 + \frac{2r}{N_{\ell r}}\bar{X}_\ell(r)\sum_{n\in\Omega_{\ell r}} X_{\ell n} - \frac{1}{N_{\ell r}}\left(\sum_{n\in\Omega_{\ell r}} X_{\ell n}\right)^2\right] \\
&\le \frac{mr}{r-m-1}\left[\frac{1}{r}\max_{n=1,\ldots,r} X_{\ell n}^2 + \frac{1}{r}\bar{X}_\ell(r)^2 + \frac{r}{r-m}\bar{X}_\ell(r)\cdot\frac{1}{r}\max_{n=1,\ldots,r} X_{\ell n} + \frac{mr}{r-m}\left(\frac{1}{r}\max_{n=1,\ldots,r} X_{\ell n}\right)^2\right]
\end{aligned}$$

Recall that $r^{-1}\sum_{n=1}^{r} X_{\ell n}^2 \to \sigma_\ell^2 + \mu_\ell^2$, and $r^{-1}\sum_{n=1}^{r} X_{\ell n} \to \mu_\ell$ w.p.1. Then, by Lemma EC.1 in the appendix, we have

$$\frac{1}{r}\max_{n=1,\ldots,r} X_{\ell n}^2 \to 0 \quad \text{and} \quad \frac{1}{r}\max_{n=1,\ldots,r} |X_{\ell n}| \to 0 \quad \text{w.p.1.}$$

By the continuous mapping theorem, the second term of $S_\ell^2(N_{\ell r})$ converges to 0 w.p.1 as $r \to \infty$. Therefore, as $\delta \to 0$, $S_\ell^2(N_{\ell r}) \to \sigma_\ell^2$ w.p.1. By continuous mapping theorem again, we obtain that

$$\frac{\sigma_1^2/r + \sigma_j^2/r}{S_1^2(N_{1r})/N_{1r} + S_j^2(N_{jr})/N_{jr}} \to 1 \quad \text{w.p.1} \quad \text{as } \delta \to 0. \tag{4}$$

We next analyze the second term on the RHS of Equation (1). Similarly to what we did above, we can write $\bar{Y}_\ell(N_{\ell r})$, $\ell = 1$ or j, in the following way,

$$\begin{aligned}
\bar{Y}_\ell(N_{\ell r}) = \frac{1}{N_{\ell r}}\sum_{n=1}^{N_{\ell r}} Y_{\ell n} &= \frac{1}{N_{\ell r}}\left[\sum_{n=1}^{r} X_{\ell n} - \sum_{n\in\Omega_{\ell r}} X_{\ell n}\right] \\
&= \frac{1}{N_{\ell r}}\sum_{n=1}^{r}(X_{\ell n} - \mu_\ell) + \frac{r}{N_{\ell r}}\mu_\ell - \frac{1}{N_{\ell r}}\sum_{n\in\Omega_{\ell r}} X_{\ell n} \\
&= \frac{\sigma_\ell\sqrt{N_{1j}^\delta}}{N_{\ell r}}\cdot H_\ell(s) + \frac{r}{N_{\ell r}}\mu_\ell - \frac{1}{N_{\ell r}}\sum_{n\in\Omega_{\ell r}} X_{\ell n},
\end{aligned}$$

where

$$H_\ell(s) = \frac{\sum_{n=1}^{r}(X_{\ell n}-\mu_\ell)}{\sigma_\ell\sqrt{N_{1j}^\delta}} = \frac{\sum_{n=1}^{\lfloor N_{1j}^\delta s\rfloor}(X_{\ell n}-\mu_\ell)}{\sigma_\ell\sqrt{N_{1j}^\delta}}, \quad s \in [\epsilon^\delta, 1],$$

and the last equality holds because $r = \lfloor N_{1j}^{\delta} s \rfloor$. Then, we have

$$\begin{aligned}
&s\sqrt{\frac{N_{1j}^{\delta}}{\sigma_1^2+\sigma_j^2}}\left[\bar{Y}_1\left(N_{1r}\right)-\bar{Y}_j(N_{jr})\right] \\
&= \left(\frac{N_{1j}^{\delta}s}{N_{1r}}\cdot\frac{\sigma_1}{\sqrt{\sigma_1^2+\sigma_j^2}}\cdot H_1(s)-\frac{N_{1j}^{\delta}s}{N_{jr}}\cdot\frac{\sigma_j}{\sqrt{\sigma_1^2+\sigma_j^2}}\cdot H_j(s)\right)+\ s\sqrt{\frac{N_{1j}^{\delta}}{\sigma_1^2+\sigma_j^2}}\cdot\left(\frac{r}{N_{1r}}\mu_1-\frac{r}{N_{jr}}\mu_j\right) \\
&\quad -\ s\sqrt{\frac{N_{1j}^{\delta}}{\sigma_1^2+\sigma_j^2}}\cdot\left(\frac{1}{N_{1r}}\sum_{n\in\Omega_{1r}}X_{1n}-\frac{1}{N_{jr}}\sum_{n\in\Omega_{jr}}X_{jn}\right). \qquad (5)
\end{aligned}$$

By Donsker's Theorem (Theorem 4.3.2 in Whitt (2002)), as $\delta\to 0$,

$$H_\ell(\cdot)\Rightarrow\mathcal{B}^\ell(\cdot),$$

where $\mathcal{B}^\ell(\cdot)$ is a standard Brownian motion process. Furthermore, because $H_1(\cdot)$ and $H_j(\cdot)$ are independent of each other, we have $\mathcal{B}^1(\cdot)$ and $\mathcal{B}^j(\cdot)$ are also independent of each other. Recall that $r=\lfloor N_{1j}^{\delta}s\rfloor$ and $r/N_{\ell r}\to 1$ w.p.1 as $\delta\to 0$. Then, $N_{1j}^{\delta}s/N_{\ell r}$, as a function of s, converges to the function that is identically equal to 1 w.p.1, for $\ell=1$ or j. By Theorem 11.4.5 in Whitt (2002), we have

$$\begin{aligned}
&\left(\frac{N_{1j}^{\delta}s}{N_{1r}}\cdot\frac{\sigma_1}{\sqrt{\sigma_1^2+\sigma_j^2}}\cdot H_1(s)-\frac{N_{1j}^{\delta}s}{N_{jr}}\cdot\frac{\sigma_j}{\sqrt{\sigma_1^2+\sigma_j^2}}\cdot H_j(s):\ 0<s\le 1\right) \\
&\Rightarrow\left(\frac{\sigma_1}{\sqrt{\sigma_1^2+\sigma_j^2}}\mathcal{B}^1(s)+\frac{\sigma_j}{\sqrt{\sigma_1^2+\sigma_j^2}}\mathcal{B}^j(s):\ 0<s\le 1\right)\ \stackrel{\mathcal{D}}{=}\ (\mathcal{B}(s):\ 0<s\le 1) \qquad (6)
\end{aligned}$$

where the last equation follows from the independence of $\mathcal{B}^1(\cdot)$ and $\mathcal{B}^j(\cdot)$ and $\stackrel{\mathcal{D}}{=}$ means "equal in distribution".

Because $r/N_{1r}\to 1$ and $r/N_{jr}\to 1$ w.p.1 as $\delta\to 0$, $\mu_1-\mu_j=\delta$ and $N_{1j}^{\delta}=\lceil 2a(\sigma_1^2+\sigma_j^2)/\delta^2\rceil$, we have as $\delta\to 0$,

$$s\sqrt{\frac{N_{1j}^{\delta}}{\sigma_1^2+\sigma_j^2}}\cdot\left(\frac{r}{N_{1r}}\mu_1-\frac{r}{N_{jr}}\mu_j\right)\to s\sqrt{2a}=s\Delta \qquad (7)$$

w.p.1 for all $s\in(0,1]$, where $\Delta=\sqrt{2a}$ by definition.

Because $r=\lfloor N_{1j}^{\delta}s\rfloor$, we have $N_{1j}^{\delta}s\le r+1$. Recall that $N_{\ell r}\ge r-m$ and $|\Omega_{\ell r}|\le m$. Then, when $n_0\ge m+1$ implying that $r\ge m+1$,

$$\begin{aligned}
s\sqrt{\frac{N_{1j}^{\delta}}{\sigma_1^2+\sigma_j^2}}\cdot\frac{1}{N_{\ell r}}\sum_{n\in\Omega_{\ell r}}X_{\ell n} &\le \sqrt{\frac{(r+1)s}{\sigma_1^2+\sigma_j^2}}\cdot\frac{m}{r-m}\cdot\max_{n=1,\dots,r}|X_{\ell n}| \\
&= \sqrt{\frac{s\,m^2}{\sigma_1^2+\sigma_j^2}}\cdot\frac{\sqrt{r(r+1)}}{r-m}\cdot\frac{1}{\sqrt{r}}\max_{n=1,\dots,r}|X_{\ell n}|.
\end{aligned}$$

We have already shown that

$$\frac{1}{r}\max_{n=1,\ldots,r} X_{\ell n}^2 \to 0 \quad \text{w.p.1}$$

as $r\to\infty$, which implies that

$$\frac{1}{\sqrt{r}}\max_{n=1,\ldots,r} |X_{\ell n}| \to 0 \quad \text{w.p.1}$$

as $r\to\infty$. Notice also that $\sqrt{r(r+1)}/(r-m)\to 1$ as $r\to\infty$. Therefore, as $\delta\to 0$,

$$s\sqrt{\frac{N_{1j}^{\delta}}{\sigma_1^2+\sigma_j^2}}\cdot\frac{1}{N_{\ell r}}\sum_{n\in\Omega_{\ell r}} X_{\ell n}\to 0 \tag{8}$$

w.p.1 for both $\ell=1$ and j.

Notice that (6), (7) and (8) correspond to the limits of the three terms on the RHS of Equation (5), respectively. Therefore, by Theorem 11.4.5 in Whitt (2002), we have as $\delta\to 0$,

$$\left(s\sqrt{\frac{N_{1j}^{\delta}}{\sigma_1^2+\sigma_j^2}}\left[\bar{Y}_1\left(N_{1r}\right)-\bar{Y}_j(N_{jr})\right]:0<s\le 1\right)\Rightarrow\mathcal{B}(s)+s\Delta=\left(\mathcal{B}_{\Delta}(s):0<s\le 1\right). \tag{9}$$

Then, by (4) and (9), and by Theorem 11.4.5 in Whitt (2002) again, we have $Z_{1j}\left(\cdot\right)\Rightarrow\mathcal{B}_{\Delta}(\cdot)$ on $(0,1]$ as $\delta\to 0$. Combined with the result in Equation (2), we conclude the proof of the lemma.

Remark 4. Lemma 1 establishes the foundation for showing the statistical validity of the APS procedure. Notice that the proof of Lemma 1 does not require the condition that $\Gamma_{i\ell}$ and Γ_{jn} are independent for $i\neq j$ or $\ell\neq n$. This indicates that the APS procedure can be implemented in parallel computing environments where the multiple processors are not identical so that the replication times may be dependent on each other.

4.2. The Asymptotic Validity

Before proving the asymptotic validity of the APS procedure, we first define the continuation region that determines the elimination decisions in the procedure. Let

$$U_{1j}^{\delta}(s)=\max\left\{0,\frac{a\sqrt{\sigma_1^2+\sigma_j^2}}{\delta\sqrt{N_{1j}^{\delta}}}-\frac{\delta\sqrt{N_{1j}^{\delta}}}{2\sqrt{\sigma_1^2+\sigma_j^2}}\cdot\frac{\sigma_1^2/r+\sigma_j^2/r}{S_1^2(N_{1r})/N_{1r}+S_j^2(N_{jr})/N_{jr}}\cdot s\right\}.$$

The symmetric continuation region C_{1j}^{δ} for $Z_{1j}(\cdot)$ is formed by the upper boundary $U_{1j}^{\delta}(s)$ and lower boundary $-U_{1j}^{\delta}(s)$. Then either alternative 1 or j is eliminated depending on whether $Z_{1j}(\cdot)$ exits the continuation region C_{1j}^{δ} from above or below. By Equation (4), it is easy to show that

$$U_{1j}^{\delta}(s)\to U(s)=\max\left\{0,\frac{a}{\Delta}-\frac{\Delta}{2}\cdot s\right\},\quad \text{w.p.1 as } \delta\to 0,$$

where $\Delta=\sqrt{2a}$. Then, the asymptotic region C, formed by $U(s)$ and $-U(s)$, is a symmetric triangular region.

Let T_{1j}^{δ} denote the stopping time at which $Z_{1j}(\cdot)$ first exits the continuation region C_{1j}^{δ}, i.e.,

$$T_{1j}^{\delta} = \inf\left\{s : |Z_{1j}(s)| \geq U_{1j}^{\delta}(s)\right\}, \tag{10}$$

and let T_{1j} denote the stopping time at which $\mathcal{B}_{\Delta}(\cdot)$ first exits the triangular region C, i.e.,

$$T_{1j} = \inf\left\{s : |\mathcal{B}_{\Delta}(s)| \geq U(s)\right\}. \tag{11}$$

Lemma 1 establishes the weak convergence of $Z_{1j}(\cdot)$ to $\mathcal{B}_{\Delta}(\cdot)$ on $[0,1]$. However, elimination decisions are only made at these stopping times. To bound the probability of incorrect selection, we need a stronger result that ensures the value at the stopping time $Z_{1j}(T_{1j}^{\delta})$ can be approximated by $\mathcal{B}_{\Delta}(T_{1j})$, which can be guaranteed by the following lemma.

Lemma 2 **(Convergence to a Brownian Motion Process at the Stopping Time)**. *Suppose that the conditions in Theorem 1 are all satisfied. Then,*

$$Z_{1j}(T_{1j}^{\delta}) \Rightarrow \mathcal{B}_{\Delta}(T_{1j})$$

as $\delta \to 0$.

Remark 5. The key idea for proving Lemma 2 is exactly the same as that of proving Proposition 3.2 of Kim et al. (2005). We summarize the proof in Appendix EC.1.3 for completeness.

To prove the validity of the APS procedure, we also need the lemma of Fabian (1974), i.e., Lemma EC.2 in the appendix, on the probability of $\mathcal{B}_{\Delta}(\cdot)$ exiting the triangular continuation region C. This lemma is the foundation of many sequential R&S procedures, including those of Kim and Nelson (2001, 2006a) and Hong and Nelson (2005, 2007). Now we are ready to prove Theorem 1.

Proof of Theorem 1: We start from the slippage configuration where $\mu_1 - \delta = \mu_2 = \cdots = \mu_k$. Then, we have

$$\begin{aligned}\liminf_{\delta\to 0} \mathbb{P}\{\text{select alternative 1}\} &= \liminf_{\delta\to 0}\left[1 - \mathbb{P}\left\{\bigcup_{j=1}^{k-1}\{\text{alternative } j \text{ eliminates 1}\}\right\}\right] \\ &\geq 1 - \limsup_{\delta\to 0}\sum_{j=1}^{k-1}\mathbb{P}\{\text{alternative } j \text{ eliminates 1}\}, \end{aligned} \tag{12}$$

where (12) is due to Bonferroni inequality. Notice that

$$\begin{aligned}\limsup_{\delta\to 0} \mathbb{P}\{\text{alternative } j \text{ eliminates 1}\} &= \limsup_{\delta\to 0} \mathbb{P}\left\{Z_{1j}\left(T_{1j}^{\delta}\right) \leq 0\right\} && (13)\\ &= \mathbb{P}\left\{\mathcal{B}_{\Delta}\left(T_{1j}\right) \leq 0\right\} && (14)\\ &= \frac{1}{2}e^{-\frac{a}{\Delta}\Delta} = \frac{\alpha}{k-1}, && (15)\end{aligned}$$

where (13) denotes the probability that alternative j eliminates alternative 1 since $Z_{1j}(\cdot)$ exits the continuation region through the lower boundary, (14) follows from Lemma 2, and (15) follows from Lemma EC.2. Plugging (15) into (12) yields

$$\liminf_{\delta\to 0}\ \mathbb{P}\{\text{select alternative 1}\} \geq 1-\sum_{j=1}^{k-1}\frac{\alpha}{k-1}=1-\alpha.$$

For general cases under the IZ formulation, i.e., $\mu_1-\delta\geq\mu_2\geq\cdots\geq\mu_k$, $Z_{1j}(\cdot)$ defined in (1) no longer converges in distribution to $\mathcal{B}_\Delta(\cdot)$. However, we can define

$$V_{1j}(s)=\frac{\sigma_1^2/r+\sigma_j^2/r}{S_1^2(N_{1r})/N_{1r}+S_j^2(N_{jr})/N_{jr}}\cdot s\sqrt{\frac{N_{1j}^\delta}{\sigma_1^2+\sigma_j^2}}\left[\bar{Y}_1(N_{1r})-\bar{Y}_j(N_{jr})-(\mu_1-\mu_j-\delta)\right].$$

Then,

$$V_{1j}(s)\leq Z_{1j}(s),\quad a.s. \tag{16}$$

By Lemma 1, we know that $V_{1j}(\cdot)\Rightarrow\mathcal{B}_\Delta(\cdot)$ as $\delta\to 0$. Let

$$T_{1j}^{\delta,V}=\inf\left\{s:|V_{1j}(s)|\geq U_{1j}^\delta(s)\right\}.$$

Then,

$$\begin{aligned}
\limsup_{\delta\to 0}\ \mathbb{P}\{\text{alternative } j \text{ eliminates } 1\} &= \limsup_{\delta\to 0}\ \mathbb{P}\left\{Z_{1j}\left(T_{1j}^\delta\right)\leq 0\right\}\\
&\leq \limsup_{\delta\to 0}\ \mathbb{P}\left\{V_{1j}\left(T_{1j}^{\delta,V}\right)\leq 0\right\} \qquad (17)\\
&= \mathbb{P}\left\{\mathcal{B}_\Delta\left(T_{1j}\right)\leq 0\right\}\\
&= \frac{\alpha}{k-1}, \qquad (18)
\end{aligned}$$

where (17) follows from (16). Plugging (18) into (12) concludes the proof of the theorem.

5. Numerical Implementation

In this section, we report on an extensive numerical study to test the effectiveness and efficiency of both the VKN procedure and the APS procedure and their applicability to solve large-scale R&S problems in parallel computing environments.

5.1. Master/Slave Structure and a Parallel Computing Simulator

We design a parallel computing environment using the Master/Slave structure, a widely used structure for parallel computing, which contains two functions: a single master and multiple slaves. The master maintains data information for all alternatives and manipulates two daemon threads (daemon threads can be viewed as service providers for other threads running in the same program. When the only remaining threads are daemon threads, the program will exit automatically), called

"to-do" and "compare". The to-do thread manages the input sequence of all surviving alternatives and the compare thread conducts pairwise comparisons and elimination decisions based on the simulation observations collected from the slaves. Each slave, created as a daemon thread, works in a very simple cycle: taking an alternative from the to-do thread, generating an observation, and submitting the observation to the compare thread for comparison. In the procedures, we denote the master as processor 0 and the slaves as processors $1, 2, \ldots, m$.

This parallel structure is programmed in Java and can be easily implemented on various computer configurations, e.g., Windows operating systems on personal computers or Linux operating systems on local servers. Moreover, with a communication protocol (e.g., HTTP), it can be extended to computer farms or clouds. For more introduction about the Master/Slave structure, we refer to Silvay and Buyya (1999) for general details and Fujimoto et al. (2010) for an implementation in the cloud. We implement our procedures using the Master/Slave structure on a local server with 48 working cores and 64 GB memory. The server runs CentOS 6.2, a Linux-based operating system.

With respect to the simulation experiments that have been used to evaluate existing R&S procedures in the literature, the observations are often generated in a very simple way, e.g., Kim and Nelson (2001) simulate one observation by generating a normal random variable, so the replication time for generating one observation tends to be extremely short. This is a common and reasonable approach for testing procedures in a single-processor computing environment, where replication times do not affect the output sequence, and, therefore, do not affect the properties of the procedure, as evaluated by the probability of correct selection, or the efficiency of the procedure, as evaluated by the expected total number of observations. In a parallel computing environment, however, replication times affect the output sequence, affecting both the properties and efficiency of the procedure. Therefore, we need to take them into consideration in experimental designs.

There are two approaches to evaluating the impact of (random) replication times. The first one is to make the slaves sleep artificially for a certain amount of time. However, this approach may be very time-consuming in testing large-scale R&S problems with a large number of processors, because replication times cannot be set too small for two reasons. One is due to the limitation of Java, which may lose accuracy when the elapsed time is less than one millisecond. In addition, it takes time to wake up a sleeping thread and it is difficult to fully control the frequent sleep-active cycles in a punctual manner. The other reason is the potential overhead on the master caused by comparison work. No matter how observations are generated, either in parallel or sequentially, they must be recorded one-by-one in the master for pairwise comparison. Even if time for processing one observation is relatively small (within 0.1 milliseconds by a rough estimation), it could cause many observations to be queueing in front of the master when the unit comparison time exceeds

the ratio of the replication time to the number of slaves. Such cases are not representative of real situations.

These concerns motivate us to consider a second approach to test parallel computing R&S procedures: we build a simulator on a single processor to simulate the situations in a parallel computing environment. As discussed in Section 2.1, experiments in a parallel computing environment can be considered as a multi-server queue. Thus, they may be simulated using a typical discrete-event simulation mechanism, where replication times are simulated under the simulation clock instead of the real clock. By managing the simulation clock properly, we can guarantee correct logic of events happening on the simulator.

In Section 5.2 we conduct an extensive numerical study on the simulator to evaluate the effectiveness and sampling efficiency of both the VKN and the APS procedures. In Section 5.3 we apply the APS procedure to solve a practical R&S problem with more than 20,000 alternatives in an actual parallel computing environment and analyze its performance.

5.2. The Effectiveness and Sampling Efficiency Tests

We assume that $X_{i\ell}$ follows a normal distribution with mean μ_i and variance σ_i^2, and $\Gamma_{i\ell}$ follows an exponential distribution with mean $\mathbb{E}[\Gamma_{i\ell}] = \gamma$ for all $i = 1, 2, \ldots, k$ and $\ell = 1, 2, \ldots$. To study how dependence between $X_{i\ell}$ and $\Gamma_{i\ell}$ affects the performance of the procedure, we consider three scenarios in which $X_{i\ell}$ and $\Gamma_{i\ell}$ are independent, positively correlated and negatively correlated, respectively. For these three scenarios, we can use the NORTA method of Cario and Nelson (1998) to generate $\Gamma_{i\ell}$ and $X_{i\ell}$ as follows,

$$\begin{aligned}\Gamma_{i\ell} &= -\gamma \log\left(1 - \Phi(W_{i\ell}^1)\right)\\ X_{i\ell} &= \mu_i + \sigma_i\left(\rho W_{i\ell}^1 + \sqrt{1-\rho^2}W_{i\ell}^2\right)\end{aligned}$$

where $W_{i\ell} = (W_{i\ell}^1, W_{i\ell}^2)$ is a bivariate standard normal vector with correlation zero. When $\rho = 0$, $X_{i\ell}$ and $\Gamma_{i\ell}$ are independent, and when $\rho > 0$ (or < 0), $X_{i\ell}$ and $\Gamma_{i\ell}$ are positively (or negatively) correlated.

We first consider the slippage configuration (SC) of means where $\mu_1 = \delta, \mu_2 = \mu_3 = \cdots = \mu_k = 0$ and the equal-variance configuration where $\sigma_i = 1$ for all i. The main goal of the experiment is to demonstrate that our procedures can solve large-scale problems using multiple processors, so we vary the number of alternatives from $k = 10^3$ to $k = 10^4$ and the number of processors as $m = 4, 48, 96$. The first-stage sample size is fixed to $n_0 = 16$, and the IZ parameter is specified as $\delta = 1/\sqrt{n_0}$. The expected replication time is $\gamma = 100$ units (in simulation clock time), and the correlation is $\rho = 0$, $\rho = 0.8$ and $\rho = -0.8$ for independent, positively correlated and negatively

correlated cases, respectively. The targeted probability of correct selection (PCS) is set to 0.95, i.e., $1-\alpha = 0.95$. To achieve two-digit precision of the estimated PCS, we made 1,000 macroreplications in all configurations.

The SC is often considered as a difficult configuration since all inferior alternatives are close to the best. For many practical large-scale problems, a substantial number of the inferior alternatives may be significantly different from the best. Therefore, we consider another configuration of means, called grouped-decreasing-means (GDM) configuration, in which 10%, 20%, 30% and 40% of the alternatives are δ, 2δ, 3δ and 4δ different from the best. The means of GDM are defined as follows,

$$\mu_i = \begin{cases} \delta, & i = 1, \\ 0, & i = 2, \ldots, \lceil 0.1k \rceil + 1, \\ -\delta, & i = \lceil 0.1k \rceil + 2, \ldots, \lceil 0.3k \rceil + 1, \\ -2\delta, & i = \lceil 0.3k \rceil + 2, \ldots, \lceil 0.6k \rceil + 1, \\ -3\delta, & i = \lceil 0.6k \rceil + 2, \ldots, k. \end{cases}$$

To make the test problem more difficult to solve, we consider an increasing-variance configuration where $\sigma_i^2 = |\mu_i - \delta| + 2\delta$, and the IZ parameter is $\delta = 0.5$. The first-stage sample size for the GDM is set to $n_0 = 10$.

In Table 1 we summarize the simulation results for both the VKN and APS procedures in all scenarios under the SC settings when the number of alternatives is $k = 10^3$. We report the average total number of observations generated (Total Samples) with 95% confidence interval, the average simulation time for completing one macroreplication of either procedure (Makespan) with 95% confidence interval, and the estimated PCS, across 1,000 macroreplications.

From the table we have several findings. First, both the VKN and APS procedures can deliver the desired PCS, but the VKN procedure tends to be more conservative than the APS procedure, which may be because the APS procedure uses variance updating and it is valid only asymptotically. Second, it seems that the correlations between performance outputs and replication times do not play an important role. Third, the total sample sizes needed for different numbers of processors m are almost the same and the makespan reduces linearly as m increases, which implies multiple processors are attractive for R&S problems. However, it is worthwhile pointing out that the makespan is computed without considering the time for processing elimination decisions on the simulator, which results a linear speedup. A linear speedup is seldom achieved in an actual parallel computing environment because the processing capacity of the master and the overhead of I/O (Input/Output) between the master and the slaves could affect the speedup as the number of processors increases. Intuitively, if there are too many slaves and the replication times are too short, then it is inevitable that many observations are ready to be sent back to the master for elimination while the master is not able to finish pairwise comparisons immediately, which, as a consequence, may accumulate a queue in front of the master (see Appendix EC.2 for an example).

Table 1 Summary under the SC settings when $k = 10^3$.

Configuration		$m = 4$		
		Independent	Positive Corr.	Negative Corr.
VKN	Total Samples	3.528×10^5	3.554×10^5	3.469×10^5
		$\pm 0.032 \times 10^5$	$\pm 0.027 \times 10^5$	$\pm 0.030 \times 10^5$
	Makespan	8.821×10^6	8.884×10^6	8.673×10^6
		$\pm 0.079 \times 10^6$	$\pm 0.069 \times 10^6$	$\pm 0.075 \times 10^6$
	PCS	0.999	0.998	0.999
APS	Total Samples	1.788×10^5	1.788×10^5	1.791×10^5
		$\pm 0.013 \times 10^5$	$\pm 0.013 \times 10^5$	$\pm 0.013 \times 10^5$
	Makespan	4.470×10^6	4.469×10^6	4.478×10^6
		$\pm 0.033 \times 10^6$	$\pm 0.032 \times 10^6$	$\pm 0.033 \times 10^6$
	PCS	0.986	0.987	0.984
Configuration		$m = 48$		
		Independent	Positive Corr.	Negative Corr.
VKN	Total Samples	3.529×10^5	3.553×10^5	3.545×10^5
		$\pm 0.032 \times 10^5$	$\pm 0.032 \times 10^5$	$\pm 0.032 \times 10^5$
	Makespan	7.358×10^5	7.401×10^5	7.385×10^5
		$\pm 0.067 \times 10^5$	$\pm 0.066 \times 10^5$	$\pm 0.066 \times 10^5$
	PCS	1.000	0.999	0.996
APS	Total Samples	1.792×10^5	1.787×10^5	1.792×10^5
		$\pm 0.013 \times 10^5$	$\pm 0.013 \times 10^5$	$\pm 0.013 \times 10^5$
	Makespan	3.733×10^5	3.722×10^5	3.733×10^5
		$\pm 0.028 \times 10^5$	$\pm 0.028 \times 10^5$	$\pm 0.027 \times 10^5$
	PCS	0.981	0.988	0.986
Configuration		$m = 96$		
		Independent	Positive Corr.	Negative Corr.
VKN	Total Samples	3.578×10^5	3.596×10^5	3.512×10^5
		$\pm 0.031 \times 10^5$	$\pm 0.029 \times 10^5$	$\pm 0.034 \times 10^5$
	Makespan	3.732×10^5	3.760×10^5	3.671×10^5
		$\pm 0.033 \times 10^5$	$\pm 0.031 \times 10^5$	$\pm 0.036 \times 10^5$
	PCS	0.998	0.998	0.997
APS	Total Samples	1.792×10^5	1.792×10^5	1.786×10^5
		$\pm 0.013 \times 10^5$	$\pm 0.013 \times 10^5$	$\pm 0.013 \times 10^5$
	Makespan	1.867×10^5	1.867×10^5	1.861×10^5
		$\pm 0.014 \times 10^5$	$\pm 0.014 \times 10^5$	$\pm 0.014 \times 10^5$
	PCS	0.982	0.984	0.978

For the GDM configuration, to avoid reporting similar results, we consider only the independent case (correlation $\rho = 0$) using the APS procedure when the number of processors is $m = 48$ and the number of alternatives varies from $k = 1\times10^3, 2\times10^3, \ldots, 10^4$. The estimated PCS for each k is always greater than the desired level 0.95. Figure 6 plots the average total sample size for different k's, from which we see that the total number of samples appears to increase almost linearly.

5.3. The Three Stage Buffer Allocation Problem

We consider a three-stage flowline with a finite number of buffer storage locations in front of stations 2 and 3 (including the one in service at each station, denoted as x_4 and x_5) and an infinite number of jobs in front of station 1 (see Buzacott and Shanthikumar (1993), Pichitlamken et al.

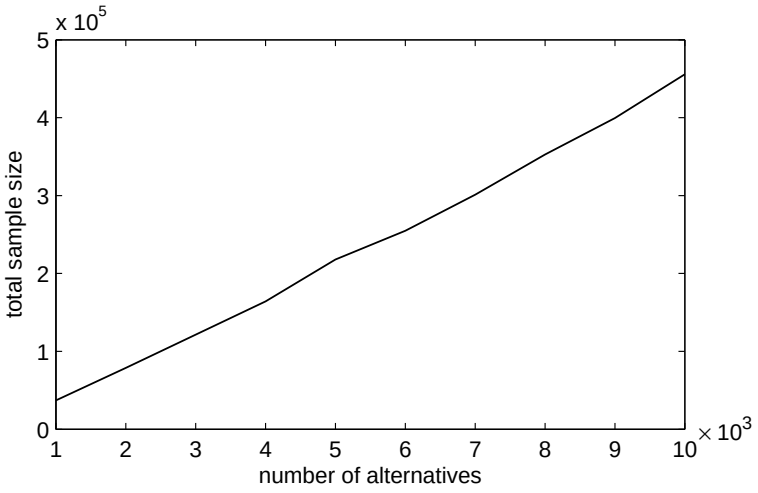


Figure 6 Total sample size vs. number of systems when $m = 48$.

(2006) and Xu et al. (2010)). There is a single server at each station, and the service time at station i is exponentially distributed with service rate x_i, $i = 1, 2, 3$. If the buffer of station i is full, then station $i-1$ is blocked (i.e., production blocking) and a finished job cannot be released from station $i-1$. The total number of buffer locations and the total service rates are limited. The goal is to find an allocation of buffer locations and service rates such that the steady-state throughput of the flowline is maximized. The constraints of this problem are $x_1 + x_2 + x_3 \leq 20$, $x_4 + x_5 = 20$, $1 \leq x_i \leq 20$ and $x_i \in \mathbb{Z}_+$ for $i = 1, 2, \ldots, 5$. The problem has totally $k = 21{,}660$ feasible solutions. For any feasible solution, the throughput is estimated from running a simulation experiment with total simulation time being 1000 units and the warm-up period being 500 units (in simulation clock time). This problem size with 21,660 alternatives was often considered too large to be solved by R&S procedures. In the simulation literature, it is often solved by optimization via simulation algorithms, as in Pichitlamken et al. (2006) and Xu et al. (2010). With parallel computing environments, however, we may solve this problem as a R&S problem.

By solving the balance equations for the underlying Markov chain from Buzacott and Shanthikumar (1993), we obtain that the optimal solutions are $(6, 7, 7, 12, 8)$ and $(7, 7, 6, 8, 12)$ (denoted as best alternatives) with steady-state throughput 5.776. We set the IZ parameter as $\delta = 0.01$ and define the feasible solutions with steady-state throughput within δ from the best as good alternatives. The event of selecting one from either the best or the good alternatives is defined as a "correct selection". Table 2 provides the information for all best and good alternatives.

Unlike the experiments reported in Section 5.2, which are implemented on a simulator of parallel computing environments, we solve this problem on a (real) local server with 48 processors and 64 GB memory and running CentOS 6.2, a Linux-based operating system. To understand how the number of processors (i.e., slaves) affect the performances of the APS procedure, we test this

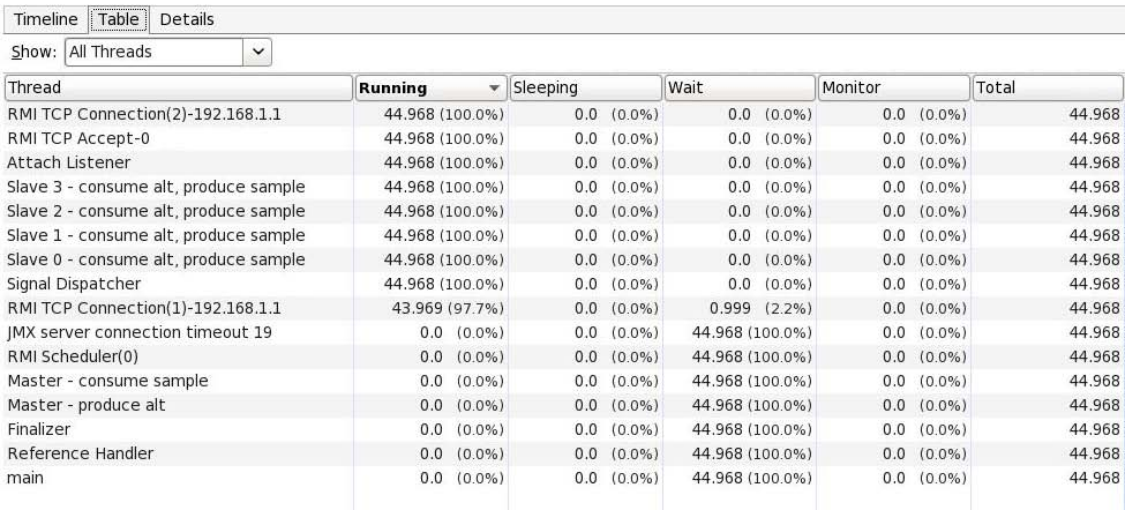
Table 2 Best and good alternatives for the buffer allocation problem.

Alternative	Throughput	Status
$(6,7,7,12,8)$	5.776	Best
$(7,7,6,8,12)$	5.776	Best
$(6,7,7,13,7)$	5.772	Good
$(7,7,6,7,13)$	5.772	Good
$(6,7,7,11,9)$	5.771	Good
$(7,7,6,9,11)$	5.771	Good

problem with different numbers of processors, $m = 1, 4, 8, 16, 32, 48$. In these experiments, we set the first-stage sample size $n_0 = 10$ and the desired PCS as 0.95.

Figure 7 captures a snapshot of the status of the threads for the Master/Slave structure after starting the APS procedure with 4 slaves. From the figure, we observe that all slaves (denoted as slaves 0 to 3) are working in parallel to generate samples while the two threads in the master, named "consume sample" thread (i.e., the compare thread) and "produce alt" thread (i.e., the to-do thread) are idling at that time since the elimination has not been conducted and the input sequence has already been prepared.

In Table 3 we report the average total sample size with 95% confidence interval, the average makespan with 95% confidence interval, and the estimated PCS, based on 100 macroreplications. We find that the total sample sizes are almost the same for various numbers of slaves and the APS procedure can always deliver a correct selection. However, we also notice that the makespan (i.e., total time to complete the procedure) seems not to reduce in proportion to the number of slaves. This is because the R&S procedure is not completely parallel. To estimate what percentage of the procedure is executed in parallel, we fit the average makespan based on Amdahl's law (Amdahl (1967)), which states that the speedup of parallelism can be defined as $\frac{1}{(1-P)+P/m}$, where P the


Timeline | Table | Details

Show: All Threads

Thread	Running	Sleeping	Wait	Monitor	Total
RMI TCP Connection(2)-192.168.1.1	44.968 (100.0%)	0.0 (0.0%)	0.0 (0.0%)	0.0 (0.0%)	44.968
RMI TCP Accept-0	44.968 (100.0%)	0.0 (0.0%)	0.0 (0.0%)	0.0 (0.0%)	44.968
Attach Listener	44.968 (100.0%)	0.0 (0.0%)	0.0 (0.0%)	0.0 (0.0%)	44.968
Slave 3 - consume alt, produce sample	44.968 (100.0%)	0.0 (0.0%)	0.0 (0.0%)	0.0 (0.0%)	44.968
Slave 2 - consume alt, produce sample	44.968 (100.0%)	0.0 (0.0%)	0.0 (0.0%)	0.0 (0.0%)	44.968
Slave 1 - consume alt, produce sample	44.968 (100.0%)	0.0 (0.0%)	0.0 (0.0%)	0.0 (0.0%)	44.968
Slave 0 - consume alt, produce sample	44.968 (100.0%)	0.0 (0.0%)	0.0 (0.0%)	0.0 (0.0%)	44.968
Signal Dispatcher	44.968 (100.0%)	0.0 (0.0%)	0.0 (0.0%)	0.0 (0.0%)	44.968
RMI TCP Connection(1)-192.168.1.1	43.969 (97.7%)	0.0 (0.0%)	0.999 (2.2%)	0.0 (0.0%)	44.968
JMX server connection timeout 19	0.0 (0.0%)	0.0 (0.0%)	44.968 (100.0%)	0.0 (0.0%)	44.968
RMI Scheduler(0)	0.0 (0.0%)	0.0 (0.0%)	44.968 (100.0%)	0.0 (0.0%)	44.968
Master - consume sample	0.0 (0.0%)	0.0 (0.0%)	44.968 (100.0%)	0.0 (0.0%)	44.968
Master - produce alt	0.0 (0.0%)	0.0 (0.0%)	44.968 (100.0%)	0.0 (0.0%)	44.968
Finalizer	0.0 (0.0%)	0.0 (0.0%)	44.968 (100.0%)	0.0 (0.0%)	44.968
Reference Handler	0.0 (0.0%)	0.0 (0.0%)	44.968 (100.0%)	0.0 (0.0%)	44.968
main	0.0 (0.0%)	0.0 (0.0%)	44.968 (100.0%)	0.0 (0.0%)	44.968

Figure 7 A screenshot of the Master/Slave with the number of processors $m = 4$.

Table 3 Summary of three-stage-buffer-allocation example with different m's.

Number of Slaves	$m=1$	$m=4$	$m=8$	$m=16$	$m=32$	$m=48$
Total samples ($\times 10^5$)	2.426	2.434	2.442	2.442	2.433	2.436
	± 0.004	± 0.004	± 0.004	± 0.004	± 0.003	± 0.004
Makespan (minutes)	370.5	129.4	94.4	68.3	41.7	34.2
	± 1.9	± 2.3	± 3.1	± 4.1	± 3.2	± 1.9
PCS	1.00	1.00	1.00	1.00	1.00	1.00

proportion of a program that can be made parallel, $1-P$ is the remaining proportion that cannot be parallelized, and m is the number of processors. Suppose the makespan, T, can be modeled as follows,

$$T = \beta\left[(1-P)+\frac{P}{m}\right]+\epsilon = (\beta-\beta P)+\beta P\cdot\frac{1}{m}+\epsilon = c_0 + c_1\frac{1}{m}+\epsilon$$

where β is the constant coefficient and ϵ is random noise. By a linear regression, we obtain that $c_0 = 40.4$ and $c_1 = 332.9$ (with $R^2 \geq 0.994$), which implies that $P = 0.892$, indicating that 89.2% of the program can be made parallel, a very high compatibility. Notice that this result is what we expected because the vast majority of the task are independent simulation runs that can be easily parallelized and it suggests that large-scale R&S problems may be effectively solved using a parallel computing environment when it is available.

6. Conclusions and Future Work

In this paper, we show that it is very attractive to solve large-scale R&S problems using parallel computing environments, which may reduce total computational time by an order of magnitude and greatly enlarge the set of R&S problems that are considered solvable. However, we also find that a direct implementation of sequential R&S procedures in a parallel computing environment may lead to unexpected statistical issues and affect the statistical validity and efficiency of procedures. In this paper, we design two different approaches to solve R&S problems in parallel computing environments.

To further improve the efficiency of the procedures, there are a few issues that are worth future investigation. First, we adopted a straightforward round-robin rule in the input sequence in this paper. However, this may not be necessary. Indeed, a higher level of efficiency may be achievable if we use more carefully chosen input sequences. Second, the current Master/Slave structure requires a large amount of communication between the master and slaves. When simulation experiments are computationally fast or there are a very large number of slaves, the master may become a bottleneck. More effective ways of handling the operations on the master are also worth studying.

Acknowledgments

The authors would like to thank the associate editor and two anonymous referees for their insightful and detailed comments that have significantly improved this paper. A preliminary version of this paper (Luo and Hong (2011)) was published in the *Proceedings of the 2011 Winter Simulation Conference*. This research was supported in part by the Hong Kong Research Grants Council [GRF XXX], the National Science Foundation [Grant XXX] and the Natural Science Foundation of China [Grants 71401104 and 71421002].

Jun Luo is an assistant professor of Antai College of Economics and Management at Shanghai Jiao Tong University. His research interests include stochastic modeling and simulation, with their applications in service operations management and healthcare management.

L. Jeff Hong is a Chair Professor of Management Sciences in the College of Business at the City University of Hong Kong. His research interests include stochastic simulation, stochastic optimization, and financial engineering and risk management.

Barry L. Nelson is the Walter P. Murphy Professor and Chair of the Department of Industrial Engineering and Management Sciences at Northwestern University. His research addresses statistical issues in the design and analysis of stochastic computer simulation experiments, including metamodeling, multivariate input modeling, simulation optimization, input uncertainty quantification, and variance reduction. He is a Fellow of INFORMS and IIE.

Yang Wu received his bachelor degree from Software Institute at Nanjing University and his master degree from Industrial Engineering and Logistics Management at the Hong Kong University of Science and Technology. Currently he is a software development engineer at the Alibaba Group in China.

References

Amdahl, Gene. 1967. Validity of the single processor approach to achieving large-scale computing capabilities. *AFIPS Conference Proceedings* **30** 483–485.

Bechhofer, Robert E. 1954. A single-sample multiple decision procedure for ranking means of normal populations with known variances. *The Annals of Mathematical Statistics* **25** 16–39.

Bechhofer, Robert E., Thomas J. Santner, David M. Goldsman. 1995. *Design and Analysis of Experiments for Statistical Selection, Screening, and Multiple Comparisons*. John Wiley & Sons, New York.

Branke, Jürgen, Stephen E. Chick, Christian Schmidt. 2007. Selecting a selection procedure. *Management Science* **53** 1916–1932.

Buzacott, John A., J. George Shanthikumar. 1993. *Stochastic Models of Manufacturing Systems*. Prentice Hall.

Cario, M. C., B. L. Nelson. 1998. Numerical methods for fitting and simulating autoregressive-to-anything processes. *INFORMS Journal on Computing* **10** 72–81.

Chen, Chun-Hung, Jianwu Lin, Enver Yücesan, Stephen E. Chick. 2000. Simulation budget allocation for further enhancing the efficiency of ordinal optimization. *Discrete Event Dynamic Systems* **10** 251–270.

Chen, E. Jack. 2005. Using parallel and distributed computing to increase the capability of selection procedures. *Proceedings of the 2005 Winter Simulation Conference*. 723–731.

Chick, S. E., P. I. Frazier. 2012. Sequential sampling with economics of selection procedures. *Management Science* **58(3)** 550–569.

Chick, Stephen E., Noah Gans. 2009. Economic analysis of simulation selection problems. *Management Science* **55** 421–437.

Chick, Stephen E., Koichiro Inoue. 2001a. New procedures to select the best simulated system using common random numbers. *Management Science* **47** 1133–1149.

Chick, Stephen E., Koichiro Inoue. 2001b. New two-stage and sequential procedures for selecting the best simulated system. *Operations Research* **49** 732–743.

Durrett, Richard. 2004. *Theory and Examples, Third Edition*. Duxbury Press.

Fabian, Vaclav. 1974. Note on anderson's sequential procedures with triangular boundary. *The Annals of Statistics* **2** 170–176.

Foster, Ian. 1995. *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering*. Parallel programming / scientific computing, Addison-Wesley.

Fujimoto, Richard M. 1990. Parallel discrete event simulation. *Communications of the ACM* **33** 30–53.

Fujimoto, Richard M., Asad Waqar Malik, Alfred J. Park. 2010. Parallel and distributed simulation in the cloud. *SCS Modeling and Simulation Magazine* **1** Society for Modeling and Simulation, Intl.

Glynn, Peter W., Philip Heidelberger. 1991. Analysis of parallel replicated simulations under a completion time constraint. *ACM Transactions on Modeling and Computer Simulation* **1** 3–23.

Heidelberger, Philip. 1988. Discrete event simulation and parallel processing: statistical properties. *SIAM Journal on Scientific and Statistical Computing* **6** 1114–1132.

Hong, L. Jeff. 2006. Fully sequential indifference-zone selection procedures with variance-dependent sampling. *Naval Research Logistics* **53** 464–476.

Hong, L. Jeff, Barry L. Nelson. 2005. The tradeoff between sampling and switching : new sequential procedures for indifference-zone selection. *IIE Transactions* **37** 623–634.

Hong, L. Jeff, Barry L. Nelson. 2007. Selecting the best system when systems are revealed sequentially. *IIE Transactions* **39** 723–734.

Hong, L. Jeff, Barry L. Nelson. 2009. A brief introduction to optimization via simulation. *Proceedings of the 2009 Winter Simulation Conference* 75–85.

Hsieh, M., P. W. Glynn. 2009. New estimators for parallel steady-state simulations. *Proceedings of the 2009 Winter Simulation Conference* 469–474.

Kim, S.-H., B. L. Nelson, James R. Wilson. 2005. Some almost-sure convergence properties useful in sequential analysis. *Sequential Anal.* **24(4)** 411–419.

Kim, Seong-Hee, Barry L. Nelson. 2001. A fully sequential procedure for indifference-zone selection in simulation. *ACM Transactions on Modeling and Computer Simulation* **11** 251–273.

Kim, Seong-Hee, Barry L. Nelson. 2006a. On the asymptotic validity of fully sequential selection procedures for steady-state simulation. *Operations Research* **54** 475–488.

Kim, Seong-Hee, Barry L. Nelson. 2006b. Selecting the best system. Shane G. Henderson, Barry L. Nelson, eds., *Elsevier Handbooks in Operations Research and Management Science: Simulation*. Elsevier, 501–534.

Luo, Jun, L. Jeff Hong. 2011. Large-scale ranking and selection using cloud computing. *Proceedings of the 2011 Winter Simulation Conference* 4051–4061.

Misra, Jayadev. 1986. Distributed discrete-event simulation. *ACM Computing Surveys* **18** 39–65.

Nelson, Barry L., Julie Swann, David Goldsman, Wheyming Song. 2001. Simple procedures for selecting the best simulated system when the number of alternatives is large. *Operations Research* **49** 950–963.

Ni, Eric Cao, Susan R. Hunter, Shane G. Henderson. 2013. Ranking and selection in a high performance computing environment. *Proceedings of the 2013 Winter Simulation Conference* 833–845.

Pichitlamken, Juta, Barry L. Nelson, L. Jeff Hong. 2006. A sequential procedure for neighborhood selection-of-the-best in optimization via simulation. *European Journal of Operational Research* **173** 283–298.

Pinedo, Michael L. 2008. *Scheduling: Theory, Algorithms, and Systems*. Third edition ed. Springer.

Rinott, Yosef. 1978. On two-stage selection procedures and related probability-inequalities. *Communications in Statistics - Theory and Methods* **A7** 799–811.

Silvay, Luís Moura E, Rajkumar Buyya. 1999. *High Performance Cluster Computing: Programming and Applications*, chap. 1. Parallel programming models and paradigms. Prentice Hall, 4–27.

Stein, Charles. 1945. A two-sample test for a linear hypothesis whose power is independent of the variance. *The Annals of Mathematical Statistics* **16** 243–258.

Whitt, Ward. 2002. *Stochastic-process limits*. Springer Series in Operations Research, Springer-Verlag, New York.

Xu, Jie, Barry L. Nelson, L. Jeff Hong. 2010. Industrial strength COMPASS: A comprehensive algorithm and software for optinization via simulation. *ACM Transactions on Modeling and Computer Simulation* **20** 1–29.

Yücesan, Enver, Yuh-Chuyn Luo, Chun-Hung Chen, Insup Lee. 2001. Distributed web-based simulation experiments for optimization. *Simulation Practice and Theory* **9** 73–90.

Technical Notes and Additional Numerical Results

EC.1. Technical Notes

EC.1.1. Derivation of the Asymptotic Independence in Section 2.3

Because there is only alternative 1, for notational simplicity we omit the 1 (denoting alternative 1) in the subscript of $Y_{1\ell}$ and replace $Y_{1\ell}$ by Y_ℓ in this section. We first derive closed-form expressions for Y_ℓ, $\ell = 1, 2, \ldots$, and their properties.

Since the m processors are identical and the service time of each customer (replication time) is exponentially distributed with mean μ_1, by the Markovian property, we know that the inter-departure time between the $(\ell-1)$th and ℓth customer, denoted by A_ℓ, $\ell = 1, 2, \ldots$, are i.i.d. with mean $\mu = \mu_1/m$ (note that A_1 is the departure time for the first customer, which is Y_1). At time $t_0 = 0$, there are m customers assigned to the server pool; at time t_ℓ, $\ell = 1, 2, \ldots$, the ℓth customer leaves the system and the next customer in queue is immediately admitted to the empty server (see Figure EC.1).

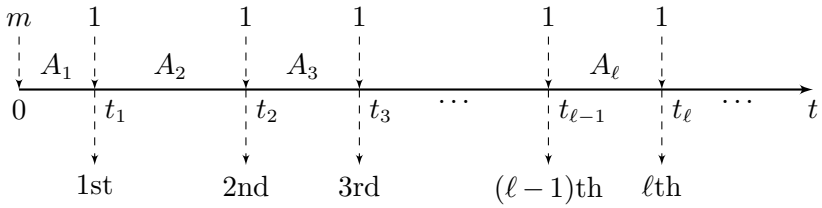


Figure EC.1 The inter-departure time.

Notice that Y_ℓ is the service time of the ℓth departing customer in this example, for $\ell = 1, 2, \ldots$. Moreover, the service time of the ℓth departing customer depends on the time point at which it enters the system. For instance, the first departure can only enter the system at time t_0, then $Y_1 = A_1$. The second departure can enter the system at times t_1 or t_0, with probabilities $\frac{m\cdot 1}{m^2}$ or $\frac{m\cdot(m-1)}{m^2}$, respectively. Then,

$$Y_2 = \begin{cases} A_2, & \text{w.p. } \frac{1}{m}, \\ A_1 + A_2, & \text{w.p. } \frac{m-1}{m}. \end{cases}$$

The third departure can enter the system at time t_2, t_1 or t_0, with probabilities $\frac{m\cdot 1\cdot m}{m^3}$, $\frac{m\cdot(m-1)\cdot 1}{m^3}$, or $\frac{m\cdot(m-1)\cdot(m-1)}{m^3}$, respectively. Then,

$$Y_3 = \begin{cases} A_3, & \text{w.p. } \frac{1}{m}, \\ A_2 + A_3, & \text{w.p. } \frac{m-1}{m^2}, \\ A_1 + A_2 + A_3, & \text{w.p. } \frac{(m-1)^2}{m^2}. \end{cases}$$

Similarly, like counting the paths in an m-ary tree, we can obtain

$$Y_\ell = \begin{cases} A_\ell, & \text{w.p. } \frac{1}{m}, \\ A_{\ell-1} + A_\ell, & \text{w.p. } \frac{m-1}{m^2}, \\ \vdots & \\ \sum_{d=2}^{\ell} A_d, & \text{w.p. } \frac{(m-1)^{\ell-2}}{m^{\ell-1}}, \\ \sum_{d=1}^{\ell} A_d, & \text{w.p. } \frac{(m-1)^{\ell-1}}{m^{\ell-1}}. \end{cases}$$

Then, based on the closed-form expression for the distribution of Y_ℓ, we can further derive the mean of Y_ℓ and expectation of the sample mean estimator $\bar{Y}_\ell(n)$, respectively, as follows,

$$\begin{aligned} \mathbb{E}[Y_\ell] &= \sum_{d=1}^{\ell} \left(\frac{m-1}{m}\right)^{\ell-d} \mathbb{E}[A_d] \;=\; \mu_1 \left[1 - \left(1 - \frac{1}{m}\right)^{\ell}\right], \\ \mathbb{E}\left[\bar{Y}_\ell(n)\right] &= \mu_1 \left\{1 - \frac{m-1}{n}\left[1 - \left(1 - \frac{1}{m}\right)^{n}\right]\right\}, \end{aligned}$$

We next use the moment generating function (MGF) to show the asymptotic independence between Y_ℓ and $Y_{\ell+n}$ as $n \to \infty$, that is,

$$\lim_{n\to\infty} \left(M_{Y_\ell}(t) \cdot M_{Y_{\ell+n}}(t) - M_{Y_\ell + Y_{\ell+n}}(t)\right) = 0. \tag{EC.1}$$

Define Y_ℓ in a rigorous way. Let $D_\ell = s$ denote the event that the ℓth departing customer enters the system at time t_s, $s = 0, 1, \ldots, \ell - 1$. Conditioning on $D_\ell = s$, Y_ℓ can be written as follows,

$$Y_\ell | \{D_\ell = s\} = \sum_{d=s+1}^{\ell} A_d.$$

Then, the MGF of Y_ℓ is $M_{Y_\ell}(t) = \mathbb{E}\left[\mathbb{E}\left[e^{tY_\ell} | D_\ell\right]\right] = \sum_{s=0}^{\ell-1} \mathbb{P}\left\{D_\ell = s\right\} \mathbb{E}\left[e^{t\sum_{d=s+1}^{\ell} A_d}\right]$.

Recall that A_ℓ are independent exponential random variables with mean $\mu = \frac{\mu_1}{m}$, so the MGF of A_d is $M_{A_d}(t) = (1 - \mu t)^{-1}$. Furthermore, we derive the probability distribution function of D_ℓ,

$$\mathbb{P}\left\{D_\ell = s\right\} = \begin{cases} \dfrac{m(m-1)^{\ell-1}}{m^\ell}, & s = 0, \\ \dfrac{m^s(m-1)^{\ell-s-1}}{m^\ell}, & s = 1, 2, \ldots, \ell - 1. \end{cases}$$

Plugging the expressions for $\mathbb{P}\left\{D_\ell = s\right\}$ and $M_{A_d}(t)$ into $M_{Y_\ell}(t)$, and with some algebra, we obtain that

$$M_{Y_\ell}(t) = \frac{1}{1 - m\mu t} - \frac{m\mu t}{1 - m\mu t}\left(\frac{m-1}{m} \cdot \frac{1}{1 - \mu t}\right)^{\ell}.$$

Notice that the MGF $M_{Y_\ell}(t)$ is well-defined in the neighborhood of zero. Furthermore, the non-negativity of $M_{Y_\ell}(t)$ implies that $1 - m\mu t > 0$, which further indicates that $\frac{m-1}{m} \cdot \frac{1}{1-\mu t} < 1$. Then, $\lim_{\ell\to\infty} M_{Y_\ell}(t) = \frac{1}{1-m\mu t}$, which is the MGF of an exponential random variable with mean $\mu_1 = m\mu$.

By calculating the first order derivative of $M_{Y_\ell}(t)$ at $t=0$, we also arrive that the conclusion that $\lim_{\ell\to\infty}\mathbb{E}[Y_\ell]=\mu_1$ as in Section 2.3.

Similarly, we can derive the closed-form expression of the MGF of $Y_{\ell+n}$ as

$$M_{Y_{\ell+n}}(t)=\frac{1}{1-m\mu t}-\frac{m\mu t}{1-m\mu t}\left(\frac{m-1}{m}\cdot\frac{1}{1-\mu t}\right)^{\ell+n}.$$

Then,

$$\lim_{n\to\infty}M_{Y_\ell}(t)\cdot M_{Y_{\ell+n}}(t)=\frac{1}{(1-m\mu t)^2}-\frac{m\mu t}{(1-m\mu t)^2}\left(\frac{m-1}{m}\cdot\frac{1}{1-\mu t}\right)^{\ell}. \tag{EC.2}$$

The MGF of $Y_\ell+Y_{\ell+n}$ is

$$\begin{aligned}M_{Y_\ell+Y_{\ell+n}}(t)&=\mathbb{E}\left[\mathbb{E}\left[e^{t(Y_\ell+Y_{\ell+n})}|D_\ell,D_{\ell+n}\right]\right]\\&=\sum_{j=0}^{\ell-1}\sum_{s=0}^{\ell+n-1}\mathbb{P}\left\{D_\ell=j,D_{\ell+n}=s\right\}\mathbb{E}\left[e^{t(\sum_{d=j+1}^{\ell}A_d+\sum_{d=s+1}^{\ell+n}A_d)}\right].\end{aligned} \tag{EC.3}$$

The joint distribution of D_ℓ and $D_{\ell+n}$ seems complicated because it depends on the relations between j and s, as well as ℓ and s. Since we are interested in the situation that $n\to\infty$, without loss of generality, we assume that $n\geq 2$. We derive the results for the case that $\ell\geq 3$ (in fact, the case that $\ell=1$ or $\ell=2$ can be handled in a similar but simpler way as $\ell\geq 3$). By enumerating all possibilities, we obtain the the joint distribution as follows.

$$\mathbb{P}\left\{D_\ell=j,D_{\ell+n}=s\right\}=\begin{cases}\dfrac{m(m-1)^n(m-2)^{\ell-1}}{m^{\ell+n}}, & j=0,\ s=0,\\ \dfrac{m(m-1)^{n+s-1}(m-2)^{\ell-s-1}}{m^{\ell+n}}, & j=0,\ s=1,\ldots,\ell-1,\\ \dfrac{m^{s-\ell+1}(m-1)^{n+2\ell-s-2}}{m^{\ell+n}}, & j=0,\ s=\ell,\ldots,\ell+n-1,\\ \dfrac{m(m-1)^{n+j-1}(m-2)^{\ell-j-1}}{m^{\ell+n}}, & j=1,\ldots,\ell-1,\ s=0,\\ \dfrac{m^s(m-1)^{n+j-s-1}(m-2)^{\ell-j-1}}{m^{\ell+n}}, & j=1,\ldots,\ell-1,\ s=1,\ldots,j-1,\\ 0, & j=1,\ldots,\ell-1,\ s=j,\\ \dfrac{m^j(m-1)^{n+s-j-1}(m-2)^{\ell-s-1}}{m^{\ell+n}}, & j=1,\ldots,\ell-1,\ s=j+1,\ldots,\ell-1,\\ \dfrac{m^{s+j-\ell}(m-1)^{n+2\ell-s-j-2}}{m^{\ell+n}}, & j=1,\ldots,\ell-1,\ s=\ell,\ldots,\ell+n-1.\end{cases}$$

Then, Equation (EC.3) can be written as the sum of eight parts,

$$M_{Y_\ell+Y_{\ell+n}}(t)=\mathbb{P}\left\{D_\ell=0,D_{\ell+n}=0\right\}\mathbb{E}\left[e^{t(\sum_{d=1}^{\ell}A_d+\sum_{d=1}^{\ell+n}A_d)}\right] \tag{EC.4}$$

$$+\sum_{s=0}^{\ell-1}\mathbb{P}\left\{D_\ell=0,D_{\ell+n}=s\right\}\mathbb{E}\left[e^{t(\sum_{d=1}^{\ell}A_d+\sum_{d=s+1}^{\ell+n}A_d)}\right] \tag{EC.5}$$

$$+\sum_{s=\ell}^{\ell+n-1}\mathbb{P}\{D_\ell=0,D_{\ell+n}=s\}\,\mathbb{E}\left[e^{t(\sum_{d=1}^{\ell}A_d+\sum_{d=s+1}^{\ell+n}A_d)}\right] \tag{EC.6}$$

$$+\sum_{j=1}^{\ell-1}\mathbb{P}\{D_\ell=j,D_{\ell+n}=0\}\,\mathbb{E}\left[e^{t(\sum_{d=j+1}^{\ell}A_d+\sum_{d=1}^{\ell+n}A_d)}\right] \tag{EC.7}$$

$$+\sum_{j=1}^{\ell-1}\sum_{s=1}^{j-1}\mathbb{P}\{D_\ell=j,D_{\ell+n}=s\}\,\mathbb{E}\left[e^{t(\sum_{d=j+1}^{\ell}A_d+\sum_{d=s+1}^{\ell+n}A_d)}\right] \tag{EC.8}$$

$$+\sum_{j=1}^{\ell-1}\mathbb{P}\{D_\ell=j,D_{\ell+n}=j\}\,\mathbb{E}\left[e^{t(\sum_{d=j+1}^{\ell}A_d+\sum_{d=j+1}^{\ell+n}A_d)}\right] \tag{EC.9}$$

$$+\sum_{j=1}^{\ell-1}\sum_{s=j+1}^{\ell-1}\mathbb{P}\{D_\ell=j,D_{\ell+n}=s\}\,\mathbb{E}\left[e^{t(\sum_{d=j+1}^{\ell}A_d+\sum_{d=s+1}^{\ell+n}A_d)}\right] \tag{EC.10}$$

$$+\sum_{j=1}^{\ell-1}\sum_{s=\ell}^{\ell+n-1}\mathbb{P}\{D_\ell=j,D_{\ell+n}=s\}\,\mathbb{E}\left[e^{t(\sum_{d=j+1}^{\ell}A_d+\sum_{d=s+1}^{\ell+n}A_d)}\right]. \tag{EC.11}$$

We now deal with the eight parts in Equation (EC.4)–(EC.11) one-by-one.

The first part is

$$\begin{aligned}(\text{EC.4}) &= \mathbb{P}\{D_\ell=0,D_{\ell+n}=0\}\,\mathbb{E}\left[e^{2t\sum_{d=1}^{\ell}A_d+t\sum_{d=\ell+1}^{\ell+n}A_d}\right]\\ &= \frac{m(m-1)^n(m-2)^{\ell-1}}{m^{\ell+n}}\cdot\frac{1}{(1-2\mu t)^\ell}\cdot\frac{1}{(1-\mu t)^n},\end{aligned}$$

which converges to zero as $n\to\infty$ because of the condition $\frac{m-1}{m}\cdot\frac{1}{1-\mu t}<1$ verified above.

The second part is

$$\begin{aligned}(\text{EC.5}) &= \sum_{s=0}^{\ell-1}\mathbb{P}\{D_\ell=0,D_{\ell+n}=s\}\,\mathbb{E}\left[e^{t\sum_{d=1}^{s}A_d+2t\sum_{d=s+1}^{\ell}A_d+t\sum_{d=\ell+1}^{\ell+n}A_d}\right]\\ &= \sum_{s=0}^{\ell-1}\frac{m(m-1)^{n+s-1}(m-2)^{\ell-s-1}}{m^{\ell+n}}\cdot\frac{1}{(1-\mu t)^s}\cdot\frac{1}{(1-2\mu t)^{\ell-s}}\cdot\frac{1}{(1-\mu t)^n}\end{aligned}$$

which converges to zero as $n\to\infty$ because of the condition $\frac{m-1}{m}\cdot\frac{1}{1-\mu t}<1$ verified above.

The third part is

$$\begin{aligned}(\text{EC.6}) &= \sum_{s=\ell}^{\ell+n-1}\mathbb{P}\{D_\ell=0,D_{\ell+n}=s\}\,\mathbb{E}\left[e^{t\sum_{d=1}^{\ell}A_d+t\sum_{d=s+1}^{\ell+n}A_d}\right]\\ &= \sum_{s=\ell}^{\ell+n-1}\frac{m^{s-\ell+1}(m-1)^{n+2\ell-s-2}}{m^{\ell+n}}\cdot\frac{1}{(1-\mu t)^{2\ell+n-s}}\\ &= \frac{m(m-1)^{\ell+n-1}}{m^{\ell+n}}\cdot\frac{1}{(1-\mu t)^{\ell+n}}\cdot\frac{-1}{1-m\mu t}+\frac{m^{n+1}(m-1)^{\ell-1}}{m^{\ell+n}}\cdot\frac{1}{(1-\mu t)^\ell}\cdot\frac{1}{1-m\mu t}\\ &\to 0+\frac{(m-1)^{\ell-1}}{m^{\ell-1}}\cdot\frac{1}{(1-\mu t)^\ell}\cdot\frac{1}{1-m\mu t},\quad\text{as}\quad n\to\infty.\end{aligned}$$

The fourth part is the same as the second part with only exchanging the positions of s and j, so (EC.7) also converges to zero as $n\to\infty$.

The fifth part is

$$
\begin{aligned}
\text{(EC.8)} &= \sum_{j=1}^{\ell-1}\sum_{s=1}^{j-1} \mathbb{P}\{D_\ell = j, D_{\ell+n} = s\}\, \mathbb{E}\left[e^{t\sum_{d=s+1}^{j} A_d + 2t\sum_{d=j+1}^{\ell} A_d + t\sum_{d=\ell+1}^{\ell+n} A_d}\right] \\
&= \sum_{j=1}^{\ell-1}\sum_{s=1}^{j-1} \frac{m^s(m-1)^{n+j-s-1}(m-2)^{\ell-j-1}}{m^{\ell+n}} \cdot \frac{1}{(1-2\mu t)^{\ell-j}} \cdot \frac{1}{(1-\mu t)^{n+j-s}} \\
&= \frac{m(m-1)^n(m-2)^{\ell-1}}{m^{\ell+n}} \cdot \frac{1}{(1-2\mu t)^{\ell-1}} \cdot \frac{1}{(1-\mu t)^{n-1}} \cdot \frac{1}{(1-m\mu t)^2} \\
&\quad - \frac{m(m-1)^{\ell+n-1}}{m^{\ell+n}} \cdot \frac{1}{(1-\mu t)^{\ell+n-2}} \cdot \frac{1}{(1-m\mu t)^2} \\
&\quad - \frac{m(m-1)^n(m-2)^{\ell-1}}{m^{\ell+n}} \cdot \frac{1}{(1-2\mu t)^{\ell-1}} \cdot \frac{1}{(1-\mu t)^{n}} \cdot \frac{1}{2(1-m\mu t)^2} \\
&\quad + \frac{m^\ell(m-1)^n}{m^{\ell+n}} \cdot \frac{1}{(1-\mu t)^{n}} \cdot \frac{1}{2(1-m\mu t)^2} \\
&\to 0, \quad \text{as} \quad n \to \infty.
\end{aligned}
$$

The sixth part (EC.9) equals zero because the joint probability equals zero.

The seventh part is

$$
\begin{aligned}
\text{(EC.10)} &= \sum_{j=1}^{\ell-1}\sum_{s=j+1}^{\ell-1} \mathbb{P}\{D_\ell = j, D_{\ell+n} = s\}\, \mathbb{E}\left[e^{t\sum_{d=j+1}^{s} A_d + 2t\sum_{d=s+1}^{\ell} A_d + t\sum_{d=\ell+1}^{\ell+n} A_d}\right] \\
&= \sum_{j=1}^{\ell-1}\sum_{s=j+1}^{\ell-1} \frac{m^j(m-1)^{n+s-j-1}(m-2)^{\ell-s-1}}{m^{\ell+n}} \cdot \frac{1}{(1-2\mu t)^{\ell-s}} \cdot \frac{1}{(1-\mu t)^{n+s-j}} \\
&= \frac{m(m-1)^n(m-2)^{\ell-1}}{m^{\ell+n}} \cdot \frac{1}{(1-2\mu t)^{\ell-2}} \cdot \frac{1}{(1-\mu t)^{n}} \cdot \frac{1}{2(1-m\mu t)^2} \\
&\quad - \frac{m^\ell(m-1)^n}{m^{\ell+n}} \cdot (1-2\mu t) \cdot \frac{1}{(1-\mu t)^{n}} \cdot \frac{1}{2(1-m\mu t)^2} \\
&\quad - \frac{m(m-1)^{\ell+n-1}}{m^{\ell+n}} \cdot \frac{1}{(1-\mu t)^{\ell+n-2}} \cdot \frac{1}{(1-m\mu t)^2} + \frac{m^\ell(m-1)^n}{m^{\ell+n}} \cdot \frac{1}{(1-\mu t)^{n-1}} \cdot \frac{1}{(1-m\mu t)^2} \\
&\to 0, \quad \text{as} \quad n \to \infty.
\end{aligned}
$$

The eighth part is

$$
\begin{aligned}
\text{(EC.11)} &= \sum_{j=1}^{\ell-1}\sum_{s=\ell}^{\ell+n-1} \mathbb{P}\{D_\ell = j, D_{\ell+n} = s\}\, \mathbb{E}\left[e^{t\sum_{d=j+1}^{\ell} A_d + t\sum_{d=s+1}^{\ell+n} A_d}\right] \\
&= \sum_{j=1}^{\ell-1}\sum_{s=\ell}^{\ell+n-1} \frac{m^{s+j-\ell}(m-1)^{n+2\ell-s-j-2}}{m^{\ell+n}} \cdot \frac{1}{(1-\mu t)^{\ell-j}} \cdot \frac{1}{(1-\mu t)^{\ell+n-s}} \\
&= \frac{m(m-1)^{\ell+n-1}}{m^{\ell+n}} \cdot \frac{1}{(1-\mu t)^{\ell+n-1}} \cdot \frac{1}{(1-m\mu t)^2} - \frac{m^{n+1}(m-1)^{\ell-1}}{m^{\ell+n}} \cdot \frac{1}{(1-\mu t)^{\ell-1}} \cdot \frac{1}{(1-m\mu t)^2} \\
&\quad - \frac{m^\ell(m-1)^n}{m^{\ell+n}} \cdot \frac{1}{(1-\mu t)^{n}} \cdot \frac{1}{(1-m\mu t)^2} + \frac{m^{\ell+n}}{m^{\ell+n}} \cdot \frac{1}{(1-m\mu t)^2} \\
&\to 0 - \frac{(m-1)^{\ell-1}}{m^{\ell-1}} \cdot \frac{1}{(1-\mu t)^{\ell-1}} \cdot \frac{1}{(1-m\mu t)^2} - 0 + \frac{1}{(1-m\mu t)^2}, \quad \text{as} \quad n \to \infty.
\end{aligned}
$$

Combining the limits of the eight parts for (EC.4)–(EC.11) and (EC.2) yields the equality in Equation (EC.1), which concludes the desired result.

EC.1.2. Useful Lemmas

LEMMA EC.1. *Let* $(x_n : n \geq 1)$ *be a real-valued sequence such that* $n^{-1}\sum_{i=1}^{n} x_i \to \mu$ *as* $n \to \infty$, *where* μ *is finite. Then* $n^{-1}\max_{i=1,\ldots,n}|x_i| \to 0$ *as* $n \to \infty$.

Proof: Let $s_n = \sum_{i=1}^{n} x_i$. Then,

$$\frac{x_n}{n} = \frac{s_n}{n} - \frac{n-1}{n}\frac{s_{n-1}}{n-1} \to \mu - \mu = 0$$

as $n \to \infty$. Since $x_n/n \to 0$, it follows that for all $\epsilon > 0$ there exists $n^* = n^*(\epsilon)$ such that for $n > n^*$, $|x_n|/n \leq \epsilon$. For $n \geq n^*$,

$$\begin{aligned}\frac{1}{n}\max_{i=1,\ldots,n}|x_i| &\leq \frac{1}{n}\max_{i=1,\ldots,n^*}|x_i| + \frac{1}{n}\max_{i=n^*+1,\ldots,n}|x_i| \\ &\leq \frac{1}{n}\max_{i=1,\ldots,n^*}|x_i| + \max_{i=n^*+1,\ldots,n}\frac{|x_i|}{i} \\ &\leq \frac{1}{n}\max_{i=1,\ldots,n^*}|x_i| + \epsilon.\end{aligned}$$

Then, the limsup of the LHS as $n \to \infty$ is at most $0 + \epsilon$. Since ϵ is arbitrary, the result follows.

LEMMA EC.2 **(Fabian (1974))**. *For a fixed triangular continuation region* C *defined by* $U(s) = \max\{0, A - Bs\}$ *and* $-U(s)$ *, if* $B = \Delta/2$ *and* $\Delta > 0$, *then*

$$\mathbb{P}[\mathcal{B}_\Delta(T) < 0] = \frac{1}{2}e^{-A\Delta},$$

where $T = \inf\{s > 0, \mathcal{B}_\Delta(s) \notin C\}$, *the random stopping time that* $\mathcal{B}_\Delta(\cdot)$ *first exits* C.

EC.1.3. Sketch of the Proof of Lemma 2

The detailed proof of Lemma 2 follows exactly the same steps as in Kim et al. (2005), so we only provide a sketch of the idea behind it.

Recall that $\mathbb{D}[0,1]$ is the Skorohod space of all right-continuous functions with left limits. Let Λ be the set of strictly increasing functions λ mapping the domain $[0,1]$ onto itself, such that both λ and its inverse λ^{-1} are continuous. Then, the Skorohod metric ρ on $\mathbb{D}[0,1]$ can be defined by

$$\rho(X,Y) = \inf_{\lambda\in\Lambda}\left\{d : \sup_{t\in[0,1]}|\lambda(t) - t| \leq d, \text{ and } \sup_{t\in[0,1]}|X(t) - Y(\lambda(t))| \leq d\right\}.$$

Besides the definition of the Skorohod metric ρ, we also need to define the following two mapping functions (as Definitions 2.1 and 2.2 in Kim et al. (2005)):

DEFINITION EC.1. On the Skorohod space $\mathbb{D}[0,1]$,

(a) For $Y \in \mathbb{D}[0,1]$, Let $T_Y(U^\delta) = \inf\left\{t : |Y(t)| \geq U^\delta(t)\right\}$, and define the function $p^\delta : Y \in \mathbb{D}[0,1] \to p^\delta(Y) \in \mathbb{R}$ by $p^\delta(Y) = Y\left(T_Y(U^\delta)\right)$.

(b) For $Y \in \mathbb{D}[0,1]$, Let $T_Y(U) = \inf\{t : |Y(t)| \geq U(t)\}$, and define the function $p : Y \in \mathbb{D}[0,1] \to p(Y) \in \mathbb{R}$ by $p(Y) = Y\left(T_Y(U)\right)$.

Notice that the forms of the upper boundaries $U^\delta(t)$ and $U(t)$ can be generally specified as in Kim et al. (2005), however, for simplicity, we may directly regard $U^\delta(t) = U_{1j}^\delta(t)$ and $U(t)$ as defined in Section 4.2.

To show that $Z_{1j}(T_{1j}^\delta) \Rightarrow \mathcal{B}_\Delta(T_{1j})$ in Lemma 2, a key step is to show that $p^\delta\left(Z_{1j}(\cdot)\right) \Rightarrow p\left(\mathcal{B}_\Delta(\cdot)\right)$ as $\delta \to 0$, which involves the functional central limit theorem and the generalized continuous mapping theorem (Theorem 3.4.4 in Whitt (2002)). The following proposition justifies that the conditions of Theorem 3.4.4 in Whitt (2002) can be satisfied.

PROPOSITION EC.1. *If $p^\delta(\cdot)$ and $p(\cdot)$ are as in Definition EC.1, then*

$$\mathbb{P}[\mathcal{B}_\Delta \in \mathbb{D}_p] = 0, \tag{EC.12}$$

where $\mathbb{D}_p$ is the set of $x \in \mathbb{D}[0,1]$ such that $p^\delta(x^\delta) \to p(x)$ fails for some sequence $\{x^\delta\}$ with $\rho(x^\delta, x) \to 0$ in $\mathbb{D}[0,1]$ as $\delta \to 0$.

Proposition EC.1 presents the same result as Proposition 3.1 in Kim et al. (2005), and interested readers may refer to the paper for the detailed proof.

Then with the results in Equation (1) and (EC.12), we can apply the generalized continuous-mapping theorem (Theorem 3.4.4 in Whitt (2002)) to show that

$$p^\delta\left(Z_{1j}(\cdot)\right) \Rightarrow p\left(\mathcal{B}_\Delta(\cdot)\right), \quad \text{as } \delta \to 0,$$

which concludes the proof.

EC.2. Numerical Results of Test Experiments using Master/Slave

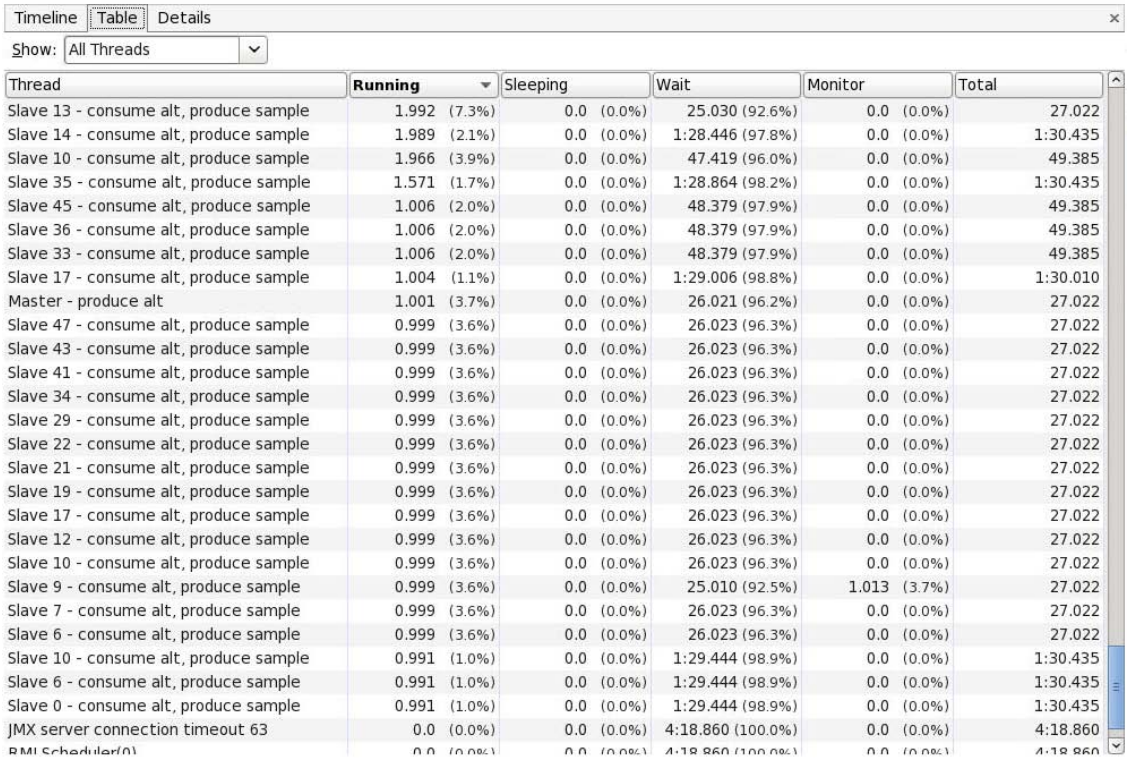
When the time to generate one observation is extremely small, then the master may not be able to process available observations immediately after they are ready to be sent back to the master. We conducted some numerical experiments on our local server to help better understand this issue.

Table EC.1 Summary under the SC settings when $m = 48$.

Number of alternatives	$k = 10^3$	$k = 10^4$	$k = 10^5$
Total samples	1.881×10^5	2.257×10^6	2.710×10^7
Makespan (seconds)	9.01	104.2	1320.6
PCS	0.99	0.99	1.00

We consider the SC with $k = 10^3, 10^4, 10^5$ and implement the APS procedure using Master/Slave structure with $m = 48$ slaves. The replication time is simply the time for generating a normal random variable ($\approx 5 \times 10^{-3}$ ms). We report the numerical result in Table EC.1. Compared with the result in Table 1 for the same $k = 10^3$ and $m = 48$, the samples simulated on the server (i.e., 1.881×10^5) is larger than that generated on the simulator (i.e., 1.792×10^5). That is because the replication time is relatively short on the server which leads to the situation that the master is not fast enough to process all available samples immediately. Thus, the available observations are filling the queueing buffer in front of the master, waiting for the comparison and elimination decisions. This is consistent with the monitoring of the buffer in front of the master during the simulation. Secondly, we find that the total sample size increases in proportion to the makespan as k increases. However, most of the time is not devoted to simulation (i.e., generating samples) but consumed by the master for computing work. For instance, it only takes about 0.02 ($\approx 1.881 \times 10^5 \times 5 \times 10^{-6}/48$) seconds to generate 1.881×10^5 samples with $m = 48$ slaves, but the procedure spends about 9.01 seconds to select the best. This means 97.8% of the time the slaves are idling either for submitting the result or waiting for the next task, which is very close to what we observe during the simulation as shown in Figure EC.2.

Figure EC.2 captures a snapshot of the status of the threads for the Master/Slave structure after starting the APS procedure with 48 slaves. From that figure, we find that all slaves (denoted as slaves 0 to 47) are working in parallel to generate samples using only a small proportion of the total time.



Timeline | Table | Details

Show: All Threads

Thread	Running	Sleeping	Wait	Monitor	Total
Slave 13 - consume alt, produce sample	1.992 (7.3%)	0.0 (0.0%)	25.030 (92.6%)	0.0 (0.0%)	27.022
Slave 14 - consume alt, produce sample	1.989 (2.1%)	0.0 (0.0%)	1:28.446 (97.8%)	0.0 (0.0%)	1:30.435
Slave 10 - consume alt, produce sample	1.966 (3.9%)	0.0 (0.0%)	47.419 (96.0%)	0.0 (0.0%)	49.385
Slave 35 - consume alt, produce sample	1.571 (1.7%)	0.0 (0.0%)	1:28.864 (98.2%)	0.0 (0.0%)	1:30.435
Slave 45 - consume alt, produce sample	1.006 (2.0%)	0.0 (0.0%)	48.379 (97.9%)	0.0 (0.0%)	49.385
Slave 36 - consume alt, produce sample	1.006 (2.0%)	0.0 (0.0%)	48.379 (97.9%)	0.0 (0.0%)	49.385
Slave 33 - consume alt, produce sample	1.006 (2.0%)	0.0 (0.0%)	48.379 (97.9%)	0.0 (0.0%)	49.385
Slave 17 - consume alt, produce sample	1.004 (1.1%)	0.0 (0.0%)	1:29.006 (98.8%)	0.0 (0.0%)	1:30.010
Master - produce alt	1.001 (3.7%)	0.0 (0.0%)	26.021 (96.2%)	0.0 (0.0%)	27.022
Slave 47 - consume alt, produce sample	0.999 (3.6%)	0.0 (0.0%)	26.023 (96.3%)	0.0 (0.0%)	27.022
Slave 43 - consume alt, produce sample	0.999 (3.6%)	0.0 (0.0%)	26.023 (96.3%)	0.0 (0.0%)	27.022
Slave 41 - consume alt, produce sample	0.999 (3.6%)	0.0 (0.0%)	26.023 (96.3%)	0.0 (0.0%)	27.022
Slave 34 - consume alt, produce sample	0.999 (3.6%)	0.0 (0.0%)	26.023 (96.3%)	0.0 (0.0%)	27.022
Slave 29 - consume alt, produce sample	0.999 (3.6%)	0.0 (0.0%)	26.023 (96.3%)	0.0 (0.0%)	27.022
Slave 22 - consume alt, produce sample	0.999 (3.6%)	0.0 (0.0%)	26.023 (96.3%)	0.0 (0.0%)	27.022
Slave 21 - consume alt, produce sample	0.999 (3.6%)	0.0 (0.0%)	26.023 (96.3%)	0.0 (0.0%)	27.022
Slave 19 - consume alt, produce sample	0.999 (3.6%)	0.0 (0.0%)	26.023 (96.3%)	0.0 (0.0%)	27.022
Slave 17 - consume alt, produce sample	0.999 (3.6%)	0.0 (0.0%)	26.023 (96.3%)	0.0 (0.0%)	27.022
Slave 12 - consume alt, produce sample	0.999 (3.6%)	0.0 (0.0%)	26.023 (96.3%)	0.0 (0.0%)	27.022
Slave 10 - consume alt, produce sample	0.999 (3.6%)	0.0 (0.0%)	26.023 (96.3%)	0.0 (0.0%)	27.022
Slave 9 - consume alt, produce sample	0.999 (3.6%)	0.0 (0.0%)	25.010 (92.5%)	1.013 (3.7%)	27.022
Slave 7 - consume alt, produce sample	0.999 (3.6%)	0.0 (0.0%)	26.023 (96.3%)	0.0 (0.0%)	27.022
Slave 6 - consume alt, produce sample	0.999 (3.6%)	0.0 (0.0%)	26.023 (96.3%)	0.0 (0.0%)	27.022
Slave 10 - consume alt, produce sample	0.991 (1.0%)	0.0 (0.0%)	1:29.444 (98.9%)	0.0 (0.0%)	1:30.435
Slave 6 - consume alt, produce sample	0.991 (1.0%)	0.0 (0.0%)	1:29.444 (98.9%)	0.0 (0.0%)	1:30.435
Slave 0 - consume alt, produce sample	0.991 (1.0%)	0.0 (0.0%)	1:29.444 (98.9%)	0.0 (0.0%)	1:30.435
JMX server connection timeout 63	0.0 (0.0%)	0.0 (0.0%)	4:18.860 (100.0%)	0.0 (0.0%)	4:18.860
RMI Scheduler(0)	0.0 (0.0%)	0.0 (0.0%)	4:18.860 (100.0%)	0.0 (0.0%)	4:18.860

Figure EC.2 A screenshot of the Master/Slave with the number of processors $m = 48$.