



Operations Research

Publication details, including instructions for authors and subscription information:
<http://pubsonline.informs.org>

Knockout-Tournament Procedures for Large-Scale Ranking and Selection in Parallel Computing Environments

Ying Zhong, L. Jeff Hong

To cite this article:

Ying Zhong, L. Jeff Hong (2022) Knockout-Tournament Procedures for Large-Scale Ranking and Selection in Parallel Computing Environments. *Operations Research* 70(1):432-453. <https://doi.org/10.1287/opre.2020.2065>

Full terms and conditions of use: <https://pubsonline.informs.org/Publications/Librarians-Portal/PubsOnLine-Terms-and-Conditions>

This article may be used only for the purposes of research, teaching, and/or private study. Commercial use or systematic downloading (by robots or other automatic processes) is prohibited without explicit Publisher approval, unless otherwise noted. For more information, contact permissions@informs.org.

The Publisher does not warrant or guarantee the article's accuracy, completeness, merchantability, fitness for a particular purpose, or non-infringement. Descriptions of, or references to, products or publications, or inclusion of an advertisement in this article, neither constitutes nor implies a guarantee, endorsement, or support of claims made of that product, publication, or service.

Copyright © 2021, INFORMS

Please scroll down for article—it is on subsequent pages



With 12,500 members from nearly 90 countries, INFORMS is the largest international association of operations research (O.R.) and analytics professionals and students. INFORMS provides unique networking and learning opportunities for individual professionals, and organizations of all types and sizes, to better understand and use O.R. and analytics tools and methods to transform strategic visions and achieve better outcomes.

For more information on INFORMS, its publications, membership, or meetings visit <http://www.informs.org>



Methods

Knockout-Tournament Procedures for Large-Scale Ranking and Selection in Parallel Computing Environments

Ying Zhong,^a L. Jeff Hong^{b,c,*}

^aSchool of Management and Economics, University of Electronic Science and Technology of China, Chengdu, 611731 China; ^bDepartment of Management Science, School of Management, Fudan University, Shanghai, 200433 China; ^cSchool of Data Science, Fudan University, Shanghai, 200433 China

*Corresponding author

Contact: yzhong4@uestc.edu.cn,  <https://orcid.org/0000-0002-8242-6966> (YZ); hong_liu@fudan.edu.cn,  <https://orcid.org/0000-0001-7011-4001> (LJH)

Received: February 16, 2019

Revised: December 11, 2019; April 21, 2020

Accepted: June 16, 2020

Published Online in Articles in Advance:
June 4, 2021

Area of Review: Simulation

<https://doi.org/10.1287/opre.2020.2065>

Copyright: © 2021 INFORMS

Abstract. On one hand, large-scale ranking and selection (R&S) problems require a large amount of computation. On the other hand, parallel computing environments that provide a large capacity for computation are becoming prevalent today, and they are accessible by ordinary users. Therefore, solving large-scale R&S problems in parallel computing environments has emerged as an important research topic in recent years. However, directly implementing traditional stagewise procedures and fully sequential procedures in parallel computing environments may encounter problems because either the procedures require too many simulation observations or the procedures' selection structures induce too many comparisons and too frequent communications among the processors. In this paper, inspired by the knockout-tournament arrangement of tennis Grand Slam tournaments, we develop new R&S procedures to solve large-scale problems in parallel computing environments. We show that no matter whether the variances of the alternatives are known or not, our procedures can theoretically achieve the lowest growth rate on the expected total sample size with respect to the number of alternatives and thus, are optimal in rate. Moreover, common random numbers can be easily adopted in our procedures to further reduce the total sample size. Meanwhile, the comparison time in our procedures is negligible compared with the simulation time, and our procedures barely request for communications among the processors.

Funding: This research was supported in part by the National Natural Science Foundation of China [Grant NSFC 71991473].

Supplemental Material: The e-companion is available at <https://doi.org/10.1287/opre.2020.2065>.

Keywords: ranking and selection • parallel computing • optimal in rate

1. Introduction

Ranking and selection (R&S) aim to find the alternative with the largest mean performance among a finite set of alternatives, through conducting experiments and observing the random performance on all alternatives. The problem was first considered by Bechhofer (1954) and has since then been studied extensively in the literature: see Bechhofer et al. (1995), Chick (2006), and Kim and Nelson (2006) for various reviews. In this paper, we consider large-scale R&S problems where experiments are conducted by running computer simulation models in parallel computing environments, especially commercial clouds. Despite that there are already some procedures that are designed to handle this type of problems (see, for instance, Luo et al. 2015 and Ni et al. 2017), in this paper, we argue that we need a new mindset in order to break through the limit of the existing R&S theory and practice.

First, when solving large-scale R&S problems, we argue that the computational complexity with respect

to the number of alternatives should be considered in designing and comparing procedures. In the classical R&S literature, procedures are typically designed to handle problems with fewer than 500 alternatives, and they are typically measured and compared by their empirical performance (e.g., the total sample size) on a few test problems. When there are a large number of alternatives (for instance, Luo et al. (2015) and Ni et al. (2017) considered problems with 10^4 and 10^6 alternatives, respectively), it becomes theoretically important to understand the growth rate of the computational effort of a procedure with respect to the number of alternatives because a procedure with a lower growth rate tends to outperform a procedure with a higher rate as the number of alternatives becomes sufficiently large. Moreover, independent of any procedures, we can analyze the lower bound of the growth rate of all R&S procedures and ask whether we can design procedures whose rates are close to or even attain the lower bound.

This provides a theoretical challenge that matters especially for large-scale problems.

Second, when using parallel computing environments, we argue that the existing stagewise procedures and fully sequential procedures are both inefficient, and we need a new framework that takes a holistic view at the total computation time, including the time spent on simulation, comparison, and communication. Stagewise procedures were first proposed to handle physical experiments, and fully sequential procedures were more suitable to handle computer experiments generated in a single processor. Using them directly in parallel computing environments for large-scale problems will cause too many simulation observations, too many comparisons, or too many communications among the processors. In this paper, we propose new R&S procedures that consider the total computation time and work well in parallel computing environments, especially commercial clouds.

1.1. Literature Review

There is a large body of literature on R&S. Based on the statistical inference used, the existing procedures may be classified into two categories: Bayesian procedures (e.g., Chick and Inoue 2001, Chick and Frazier 2012) and frequentist's procedures (e.g., Paulson 1964, Rinott 1978, Kim and Nelson 2001). Bayesian procedures typically allocate a finite number of samples (i.e., experiments) to maximize some performance measures based on the posterior distributions (e.g., posterior probability of correct selection (PCS) (Chen et al. 2000, Chick and Inoue 2001) and expected value of information (Inoue and Chick 1998, Chick and Gans 2009)) or treat the problem as a dynamic program and solve it using some approximations (e.g., the knowledge gradient approach (Frazier et al. 2008, 2009) and the more recent approximations of Peng et al. (2018)). Frequentist's procedures typically aim to deliver a predefined *probability of correct selection* under the so-called indifference zone (IZ) formulation, first proposed by Bechhofer (1954), where the mean performance of the best alternative is at least $\delta > 0$ better than those of the other alternatives. Recently, there are some works on developing IZ-free frequentist's procedures (e.g., Fan et al. 2016, Zhong and Hong 2017). In this paper, we still adopt the IZ formulation.

Two types of procedures have been developed under the IZ formulation, stagewise procedures and fully sequential procedures. Stagewise procedures typically determine the total sample size of each alternative at the beginning of the selection process or after collecting some initial observations to estimate the variances and collect all observations at once to identify the best (e.g., Bechhofer 1954, Dudewicz and Dalal 1975, Rinott 1978).

They were mainly developed in early days of the R&S field, when problems were of small size (e.g., 100 or fewer alternatives) and observations were often collected from physical experiments (e.g., clinical trials and agricultural experiments). For these problems, it often takes a long time to generate an observation, but one can simultaneously generate a large number of them at a time. Stagewise procedures are naturally parallelizable (i.e., experiments may be conducted simultaneously), and they fit the problems very well. Fully sequential procedures collect one observation at a time from all surviving alternatives and eliminate alternatives when there is enough evidence (e.g., Paulson 1964, Kim and Nelson 2001, Hong 2006). They became popular with the fast development of computer technology when observations are increasingly collected from running computer simulation models typically on a single processor. In this situation, the time needed to generate an observation is significantly shortened, and the observations are collected one at a time. Therefore, fully sequential procedures fit the situation well. They reduce the total sample size and are capable of solving larger problems (e.g., up to 1000 alternatives). For both types of procedures, when observations are generated by computer simulations, common random numbers (CRNs) are often used to introduce positive correlations among the observations from different alternatives and to reduce the total sample size (Nelson and Matejcek 1995, Kim and Nelson 2001).

Recently, with the growing needs to solve large-scale R&S problems (with tens of thousands to even millions of alternatives), some works start to look into the possibilities of using parallel computing environments (e.g., commercial cloud services) with multiple processors to solve R&S problems. The first work in this line of research is done by Luo et al. (2015) (and their conference version: Luo and Hong 2011). By distributing the simulation experiments in a round-robin manner to all available processors, they implement Procedure \mathcal{KN} , a well-known fully sequential procedure developed by Kim and Nelson (2001), in a parallel computing environment and successfully solve problems with more than 20,000 alternatives. However, they also observe new issues. Procedure \mathcal{KN} requires all pairwise comparisons among all surviving alternatives after collecting each round of samples. The number of comparisons grows in a higher order than the total sample size does, and they cannot be easily parallelized. Therefore, the comparisons may become the bottleneck of the procedure, especially when the number of alternatives is large. Ni et al. (2017) (and their conference version: Ni et al. 2014) propose a "divide and conquer" approach to address this issue so that the number of comparisons in their procedure is negligible compared with the overall computation. By doing so, their procedure

dramatically increases the size of solvable problems. Their procedure is capable of solving problems with 10^6 alternatives. However, because of frequent communications among the processors, the performance of their procedure is not satisfactory on some popular parallel computing platforms (e.g., Apache Hadoop, where the communication cost is considerably high). It is also important to notice that both aforementioned procedures do not support the use of CRNs.

In general, designing parallel R&S procedures is non-trivial; see Hunter and Nelson (2017) for a comprehensive summary about the challenges of designing parallel procedures and the recent development of the field in this direction. Parallel computing environments are different from both physical experiments that are parallel in nature and single-processor computing environments that are sequential in nature. They support running multiple experiments at the same time, but the number of parallel experiments is limited by the number of processors, which is typically smaller than a few hundred; the experiments conducted on each processor are sequential in nature, and the communications among processors are typically time consuming and inefficient. Therefore, in this paper, instead of modifying the existing procedures that are inherently inefficient in parallel computing environments, we design new R&S procedures that are specifically fit for these environments.

1.2. Knockout Tournament

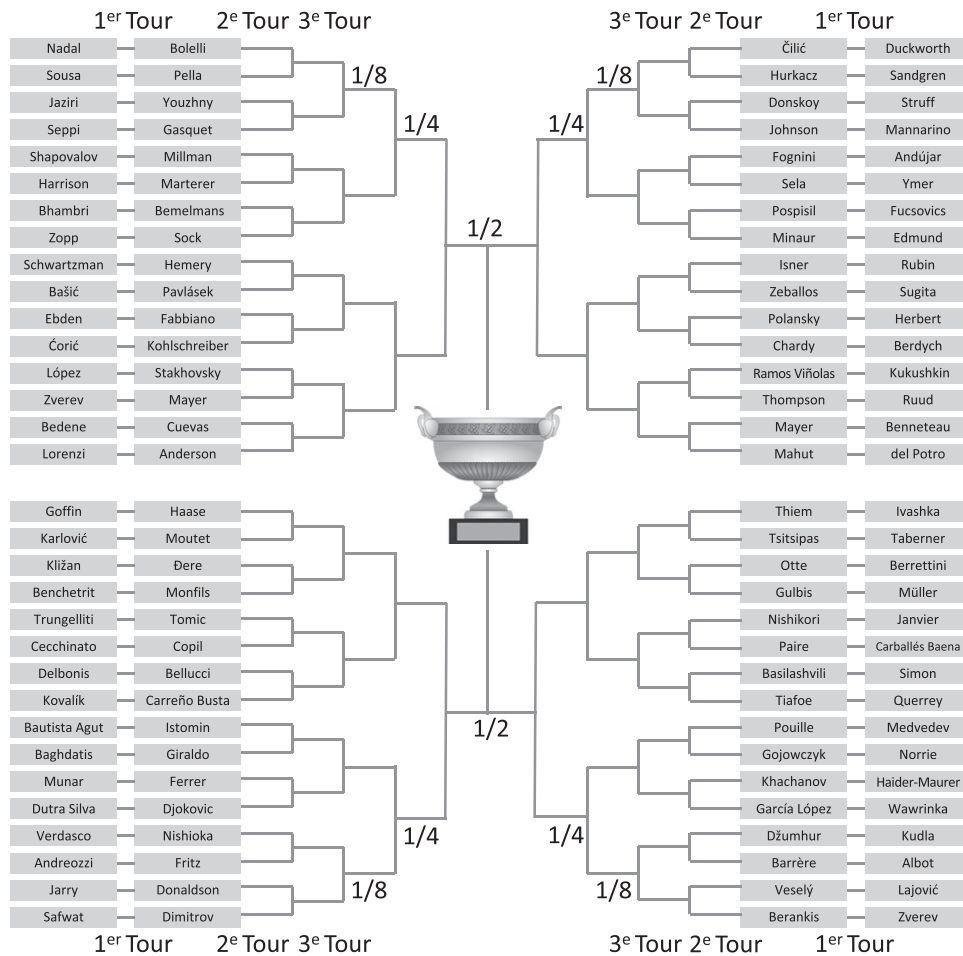
In this paper, we develop new R&S procedures inspired by the knockout-tournament arrangement of tennis Grand Slam tournaments. A knockout tournament, also known as a single-elimination tournament, is a type of elimination tournament where the loser of each matchup is immediately eliminated from the tournament, and the winner will move on to the next round until the final matchup, whose winner becomes the tournament champion (Kim et al. 2017). In Figure 1, we show the bracket of the 2018 French Open men's singles. There are several interesting observations of the bracket. First, there are in total 128 players in the bracket. To become the champion, a player does not need to beat all other 127 players. Instead, he only needs to win seven (i.e., $\log_2 128$) matches. Furthermore, there are in total 127 matches, and each player plays an average of fewer than 2 matches. Second, the players may be divided into several groups (e.g., four quarters as shown in Figure 1). Technically, we can assign each group a tennis court, and the matches within the group may be conducted only on the court until the winner of the group is identified. The matches of all groups may be conducted simultaneously on all the courts, and no coordination across groups is necessary.

Following the knockout-tournament scheme, we develop R&S procedures that have a “match” between two alternatives at a time. The winner goes on, and the loser is knocked out. The procedures use Procedure \mathcal{KN} to conduct the matches to achieve the finite-time statistical guarantee and to gain efficiency on the total sample size. The procedures eliminate about half of the alternatives at each round of the selection and theoretically achieve the lowest growth rate on the expected total sample size no matter whether the variances of the alternatives are known or not. Moreover, because comparisons are only made within matches and no all-pairwise comparisons are needed, one can trivially show that the comparison time in the procedures is negligible compared with the simulation time, and CRNs can be easily applied in this situation without any global synchronizations. To implement our procedures in parallel computing environments, the set of alternatives can be divided into small subsets, and the selections in different subsets can be conducted on different processor simultaneously. Therefore, no communications are necessary among the processors until each processor produces a local best alternative.

To further speed up the selection process in parallel computing environments, we also investigate some extensions on the procedures. First, if only two alternatives are allowed in a match, when both alternatives are clearly inferior and have similar means, the procedures tend to spend a significant amount of observations to distinguish them. This can cause efficiency loss on the sample size and slow down the selection process. To overcome this issue, we suggest properly expanding the number of alternatives in a match. By doing so, we lower the chance that all alternatives in a match are equally “bad”. Second, when every processor produces a local best alternative, keeping the knockout-tournament selection structure can cause processor idling because during the rest of the selection process, there are fewer alternatives than the processors. To maximize the utilizations of the processors, after a processor decides a local best alternative, we let it additionally simulate enough observations as suggested by a stagewise procedure for the alternative. Then, after every processor finds a local best alternative, we only need to compare the sample means of these local best alternatives to select the final best alternative. As we shall see, these modifications can significantly improve the empirical performance of our procedures. In the meantime, they may not affect the rates of theoretical upper bounds of the procedures' running times but will change the constants on the upper bounds.

We close this section with a remark that, in the multi-armed bandit field, there is a stream of research works

Figure 1. The 2018 French Open Men’s Singles Bracket



known as the best arm identification. It is very similar to the context of R&S problems, except that they typically consider the cases where the samples are drawn from sub-Gaussian distributions with known variance proxies (e.g., distributions with bounded supports). Many novel procedures have been proposed in the literature (e.g., the successive elimination procedure (Even-Dar et al. 2002), the median elimination procedure (Even-Dar et al. 2002), and the lil’UCB procedure where UCB stands for upper confidence bound (Jamieson et al. 2014)). It is worth noting that, similar to the knockout-tournament scheme, the median elimination procedure also eliminates half of the alternatives at each round of the selection by eliminating the alternatives whose sample means are below the median. However, like all pairwise comparisons, finding the median of the sample means has a high order of computational complexity, can only be done on a single processor, and requires a large number of communications among the processors. Thus, the procedure is not suitable for parallel computing environments as well.

The rest of this paper is organized as follows. In Section 2, we introduce the problem formulation and conduct lower-bound analysis on the growth rate of the expected total sample size for a general R&S procedure. In Section 3, inspired by the tennis knockout-tournament scheme, we design a procedure under the setting where the observations are normally distributed with a common known variance, and we extend the procedure to handle problems with unknown and unequal variances in Section 4. In Section 5, we further propose several modifications on the procedure to make it more suitable for solving large-scale R&S problems in parallel computing environments. In Section 6, we conduct a comprehensive numerical study to understand the performance of our procedures and compare them with existing procedures. We conclude in Section 7.

2. Lower-Bound Analysis on Expected Total Sample Size

Traditionally, when the problem size (i.e., the number of alternatives) is relatively small, numerical experiments

are often conducted to compare the total sample size of different procedures. Meanwhile, with an increasing number of alternatives, the growth rate of the expected total sample size with respect to the number of alternatives may become an important factor affecting the total sample size of a procedure. Therefore, analyzing the growth rate provides a theoretical framework to compare various large-scale R&S procedures.

In the following subsections, we first give a mathematical formulation of the R&S problem. We then establish a lower bound on the growth rate of the expected total sample size of a general R&S procedure, as long as it can guarantee to select the best alternative with the desired probability. In the later sections, we use this lower bound as the benchmark to compare the growth rates of different R&S procedures, including ours.

2.1. Problem Formulation

Suppose that there are in total k alternatives in contention at the beginning of the selection process, and we use $\mathcal{K} = \{1, 2, \dots, k\}$ to index all the alternatives. For each alternative $i \in \mathcal{K}$, it generates independent and identically distributed (i.i.d.) simulation observations $X_{i,1}, X_{i,2}, \dots$, from the normal distribution with mean μ_i and variance σ_i^2 . Furthermore, we use $\bar{X}_i(t) = \sum_{\ell=1}^t X_{i,\ell}/t$ to denote the sample average of alternative i based on the first t observations. In this paper, we allow the use of CRNs (i.e., the observations generated by different alternatives may be correlated). Without loss of generality, we assume that the means are in an ascending order throughout this paper (i.e., $\mu_1 \leq \mu_2 \leq \dots \leq \mu_{k-1} \leq \mu_k$) and alternative k is the best. Following the convention in the ranking and selection literature, while developing the procedures, we adopt the indifference zone formulation. We assume that there exists a minimal gap $\delta > 0$, which is also called the IZ parameter and prespecified by the user, between the means of the best alternative and the second-best alternative (i.e., $\mu_k - \mu_{k-1} \geq \delta$). Our objective is to design procedures that can achieve, if $\mu_k - \mu_{k-1} \geq \delta$,

$$\mathbb{P}(\text{select alternative } k) \geq 1 - \alpha, \quad (1)$$

where $1 - \alpha$ is called the *probability of correct selection*. In this paper, we call the procedure satisfying Equation (1) the procedure with the PCS guarantee.

As pointed out by Ni et al. (2017), there is a theoretical issue related to using a procedure with the PCS guarantee to solve large-scale problems. When the number of alternatives gets large, it becomes increasingly difficult to ensure that the mean of the best alternative is at least δ better than those of others. There may exist multiple alternatives lying within δ to the best. As a result, the IZ assumption no longer holds in this situation. To address this issue, Ni et al. (2017) adopt a stronger

selection guarantee, which requires a procedure to select an alternative within δ to the best with probability at least $1 - \alpha$ even when the IZ assumption is violated. It is called the *probably approximately correct* (PAC) guarantee. Let \mathfrak{J} denote the index of the alternative that a procedure terminates with. The PAC guarantee essentially requires the procedure to achieve that

$$\mathbb{P}(\mu_k - \mu_{\mathfrak{J}} \leq \delta) \geq 1 - \alpha.$$

The PAC guarantee does not make any assumptions on the configuration of the means. In this paper, even though we focus primarily on developing procedures with the PCS guarantee, we show that, with some slight modifications, our procedures can satisfy the PAC guarantee as well, but with a significant efficiency loss observed empirically.

2.2. The Lower Bound

Let \mathbf{N} denote the total sample size of a general R&S procedure, and \mathbf{N} is a random variable. In this paper, we study the growth rate of $\mathbb{E}[\mathbf{N}]$ as a function of k . Following the definitions in Cormen et al. (2001), we first introduce the Big- Ω notation and Big- \mathcal{O} notation as follows.

Definition 1. For a set of constants y_q , where $q = 1, 2, \dots$, and a corresponding set of constants b_q where there exists a constant $Q > 0$ such that $b_q > 0$ for all $q > Q$, the notation

$$y_q = \Omega(b_q),$$

means that y_q grows at least in the order of b_q . That is, there exist finite constants $M_1 > 0$ and $Q_1 > 0$ such that $|y_q| \geq M_1 b_q$, for all $q > Q_1$.

Definition 2. For a set of constants y_q , where $q = 1, 2, \dots$, and a corresponding set of constants b'_q where there exists a constant $Q' > 0$ such that $b'_q > 0$ for all $q > Q'$, the notation

$$y_q = \mathcal{O}(b'_q)$$

means that y_q grows at most in the order of b'_q . That is, there exist finite constants $M'_1 > 0$ and $Q'_1 > 0$ such that $|y_q| \leq M'_1 b'_q$, for all $q > Q'_1$.

When $y_q = \Omega(b_q)$, we can view b_q as the lower bound of the growth rate of y_q . Similarly, if $y_q = \mathcal{O}(b'_q)$, we can view b'_q as the upper bound of the growth rate of y_q . If sets b_q and b'_q are identical (i.e., $b_q = b'_q$ for all $q \geq 1$), then we can conclude that y_q grows in the exact same rate as that of b_q .

In order to proceed our lower-bound analysis, we further make the following two assumptions on the general R&S procedure that we consider. They basically state that, when the procedure allocates the observations, the alternatives with observations do not

provide any information on the alternatives without any observations, and the procedure can only randomly pick an alternative as the best if it decides to select the best from a set of alternatives with no observations.

Assumption 1. Let \mathcal{E} denote the set of alternatives without any observations when the procedure decides to stop. If the procedure chooses to select the best from \mathcal{E} , then, for any alternative $i \in \mathcal{E}$, the probability that the alternative is selected is $1/|\mathcal{E}|$, where $|\mathcal{E}|$ denotes the cardinality of the set \mathcal{E} .

Assumption 2. Let \mathcal{E}_s denote the set of alternatives without any observations when the procedure allocates the s th observation. If the procedure chooses to allocate the s th observation to an alternative in \mathcal{E}_s , then for any alternative $i \in \mathcal{E}_s$, the probability that the observation is allocated to i is $1/|\mathcal{E}_s|$.

With the conditions stated in Assumptions 1 and 2, we can prove that $\mathbb{E}[\mathbf{N}] = \Omega(k)$, summarized by the following theorem.

Theorem 1. If Assumptions 1 and 2 hold, for any procedure that can select the best alternative with probability at least $1 - \alpha$, where $1 - \alpha$ is the predefined PCS and $0 < \alpha < 1/k$, we have

$$\mathbb{E}[\mathbf{N}] = \Omega(k).$$

Proof. We prove Theorem 1 by contradiction. Let $K_1 = 16/(3 + \alpha)(1 - \alpha)$, $C_1 = (1 - \alpha)^2/8$, and $C_2 = 1 - \alpha/4$. Consider the case of $k \geq K_1$. If $\mathbb{E}[\mathbf{N}] < C_1 k$, by Markov's Inequality, we have

$$\mathbb{P}(\mathbf{N} \geq C_2 k) \times C_2 k \leq \mathbb{E}[\mathbf{N}] < C_1 k.$$

It further yields

$$\mathbb{P}(\mathbf{N} \geq C_2 k) < \frac{C_1}{C_2}. \quad (2)$$

Then, for the procedure, we can decompose the probability of correctly selecting alternative k as

$$\begin{aligned} \mathbb{P}(\text{select alternative (alt.) } k) &= \mathbb{P}(\{\text{select alt. } k\} \cap \{\mathbf{N} \geq C_2 k\}) \\ &\quad + \mathbb{P}(\{\text{select alt. } k\} \cap \{\mathbf{N} < C_2 k\}) \\ &< \frac{C_1}{C_2} + \mathbb{P}(\{\text{select alt. } k\} \cap \{\mathbf{N} < C_2 k\} \\ &\quad \cap \{\text{alt. } k \text{ has an observation (obs.)}\}) \\ &\quad + \mathbb{P}(\{\text{select alt. } k\} \cap \{\mathbf{N} < C_2 k\} \\ &\quad \cap \{\text{alt. } k \text{ does not have an obs.}\}), \end{aligned} \quad (3)$$

where the first inequality follows directly from Equation (2). Next, we show that the second term in Equation (3) is upper bounded by C_2 . Let \mathcal{Q}_s denote the event that the procedure does not allocate the s th observation to alternative k . Then, we have

$$\begin{aligned} &\mathbb{P}(\{\text{select alt. } k\} \cap \{\mathbf{N} < C_2 k\} \cap \{\text{alt. } k \text{ has an obs.}\}) \\ &\leq \mathbb{P}(\{\mathbf{N} < C_2 k\} \cap \{\text{alt. } k \text{ has an obs.}\}) \\ &\leq 1 - \mathbb{P}\left(\bigcap_{s=1}^{C_2 k} \{\mathcal{Q}_s\}\right) \\ &= 1 - \mathbb{P}(\mathcal{Q}_1) \times \prod_{s=2}^{C_2 k} \mathbb{P}(\mathcal{Q}_s | \mathcal{Q}_1, \dots, \mathcal{Q}_{s-1}) \\ &\leq 1 - \frac{k-1}{k} \times \frac{k-2}{k-1} \times \dots \times \frac{k-C_2 k}{k-C_2 k+1} \\ &= C_2. \end{aligned} \quad (4)$$

For the third inequality, if events $\{\mathcal{Q}_1, \dots, \mathcal{Q}_{s-1}\}$ hold, when the procedure allocates the s th observation, including alternative k , there are at least $k - s + 1$ alternatives having no observations (i.e., $|\mathcal{E}_s| \geq k - s + 1$). Then, by Assumption 2, $\mathbb{P}(\mathcal{Q}_s | \mathcal{Q}_1, \dots, \mathcal{Q}_{s-1}) \geq (k - s) / (k - s + 1)$.

For the third term in Equation (3), if the total sample size $\mathbf{N} < C_2 k$, then among all k alternatives, there are at least $k - C_2 k$ alternatives having no observations. Then, by Assumption 1, we can deduce that

$$\begin{aligned} &\mathbb{P}(\{\text{select alt. } k\} \cap \{\mathbf{N} < C_2 k\} \\ &\quad \cap \{\text{alt. } k \text{ does not have an obs.}\}) \\ &\leq \mathbb{P}(\{\text{select alt. } k\} \\ &\quad \cap \{\text{at least } k - C_2 k \text{ alt. } s. \text{ do not have an obs.}\} \\ &\quad \cap \{\text{alt. } k \text{ does not have an obs.}\}) \\ &\leq \frac{1}{(1 - C_2)k}. \end{aligned} \quad (5)$$

Plugging the results in (4) and (5) to Equation (3), we have

$$\begin{aligned} \mathbb{P}(\text{select alt. } k) &< \frac{C_1}{C_2} + \mathbb{P}(\{\text{select alt. } k\} \\ &\quad \cap \{\mathbf{N} < C_2 k\} \cap \{\text{alt. } k \text{ has an obs.}\}) \\ &\quad + \mathbb{P}(\{\text{select alt. } k\} \cap \{\mathbf{N} < C_2 k\} \\ &\quad \cap \{\text{alt. } k \text{ does not have an obs.}\}) \\ &\leq \frac{C_1}{C_2} + C_2 + \frac{1}{(1 - C_2)k} \\ &< 1 - \alpha, \end{aligned} \quad (6)$$

The last inequality holds by the definitions of C_1 , C_2 and the assumption that $k \geq K_1$. Equation (6) contradicts the fact that the procedure can guarantee to select the best alternative with probability at least $1 - \alpha$. Thus, we have

$$\mathbb{E}[\mathbf{N}] \geq C_1 k, \quad \text{for all } k \geq K_1.$$

Therefore, by Definition 1, we have $\mathbb{E}[\mathbf{N}] = \Omega(k)$. \square

Theorem 1 applies to the procedures no matter whether CRNs are used or not. Also, the result in Theorem 1 is developed without employing the IZ assumption. It is

a universal lower bound for all procedures that can guarantee to select the best alternative with probability at least $1 - \alpha$, including IZ procedures as well as IZ-free procedures where $\delta = 0$. In the rest of this paper, we call an R&S procedure with the PCS guarantee *optimal in rate* if the growth rate of $\mathbb{E}[N]$ for the procedure is upper bounded by the universal lower-bound rate (i.e., $\mathbb{E}[N] = \mathcal{O}(k)$).

3. Procedure for Common Known Variance Case

In this section, we first introduce the traditional stage-wise and fully sequential procedures, as they are the building blocks of our procedures. We then consider the simplest case, where the variances of all alternatives are known and equal, to illustrate our main idea on how to develop a procedure that is suitable for solving large-scale R&S problems in parallel computing environments and is optimal in rate. We extend the idea to the more realistic case with unknown and unequal variances in Section 4.

Throughout this section, we make the following assumption on the observations of all alternatives. It basically states that the observations generated from all alternatives are normally distributed with a common known variance and are independent.

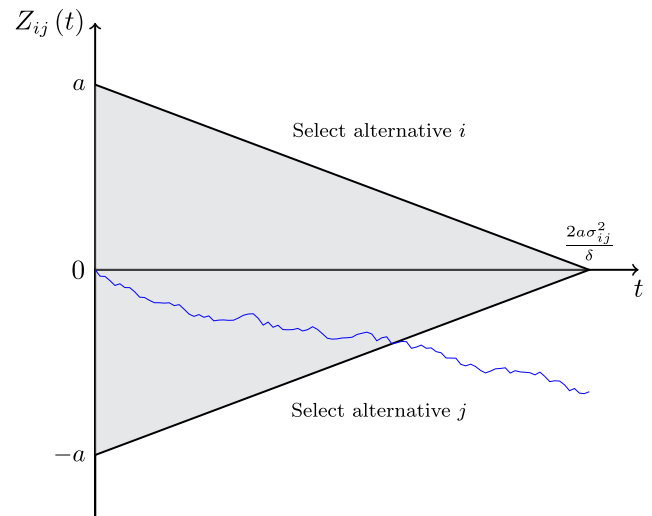
Assumption 3. For each alternative $i \in \mathcal{K}$, it generates i.i.d. observations $X_{i,\ell}$, for $\ell = 1, 2, \dots$, from the normal distribution with unknown mean μ_i and known variance σ^2 . The observations generated by different alternatives are independent.

Remark 1. In this section, for illustration purposes, we consider a simple case where the observations generated by different alternatives are independent. All analyses apply to the case where the observations generated by different alternatives are correlated. However, the latter case requires one’s foreknowledge on the variance of the difference between any two alternatives instead of the variance of each alternative.

3.1. Traditional Stagewise and Fully Sequential Procedures

In the R&S literature, two types of procedures with the PCS guarantee have been developed. One is the stagewise procedures (e.g., Bechhofer’s procedure and Rinott’s procedure). A stagewise procedure typically determines the total sample size of each alternative at the beginning of the selection process (or after the first stage that estimates the variances of all alternatives) and collects all observations at once. Then, the procedure calculates the sample means of all alternatives and selects the one with the largest sample mean as the best. The other one is the fully sequential procedures (e.g., Paulson’s procedure (Paulson 1964) and Procedure

Figure 2. (Color online) Continuation Region (Gray Area) for $Z_{ij}(t)$ in Procedure \mathcal{KN} , Where $a = \log[(k - 1)/(2\alpha)]/\delta$



\mathcal{KN}). A fully sequential procedure typically models the partial sum difference process $\{Z_{ij}(t) = t[\bar{X}_i(t) - \bar{X}_j(t)]/\sigma_{ij}^2 : t = 1, 2, \dots\}$ between any two alternatives i and j as a Brownian motion (BM) process, where σ_{ij}^2 is the variance of $X_{i,\ell} - X_{j,\ell}$.¹ As shown in Figure 2, after the partial sum difference process exits the predetermined triangular continuation region, one alternative can be eliminated accordingly. The procedure stops when only a single alternative is left and selects it as the best. Because fully sequential procedures allow early eliminations of the clearly inferior alternatives, they typically have smaller total sample size than stagewise procedures.

In order to satisfy the PCS guarantee, both types of procedures decompose the original problem of selecting the best alternative into $k - 1$ subproblems in which alternative k competes with all the alternatives i , for $i = 1, 2, \dots, k - 1$. Alternative k needs to eliminate all other $k - 1$ alternatives to be the best. These procedures are typically designed to ensure that for each competition between alternative k and any alternative i , for $i = 1, 2, \dots, k - 1$, false elimination of alternative k happens with probability at most $\alpha/(k - 1)$, so that

$$\begin{aligned} \mathbb{P}(\text{alt. } k \text{ is eliminated}) &\leq \sum_{i=1}^{k-1} \mathbb{P}(\text{alt. } i \text{ eliminates alt. } k) \\ &\leq (k - 1) \times \frac{\alpha}{k - 1} = \alpha. \end{aligned} \tag{7}$$

Equation (7) implies that, for the procedures designed under this framework, as the number of alternatives k increases, the desired false elimination probability $\alpha/(k - 1)$ decreases. As a result, more observations are needed for alternative k and every individual alternative i , for $i = 1, 2, \dots, k - 1$, that alternative k competes with. Because the overall total sample size sums over

all alternatives, the expected total sample size of these procedures grows faster than the order of k . Therefore, it is quite difficult to achieve the rate optimality under the current R&S framework.²

Furthermore, these procedures were originally developed with the purpose of solving relatively small-size problems. Directly implementing these procedures in parallel computing environments to solve large-scale R&S problems may not work well. For instance, even though stagewise procedures are easy to parallelize (i.e., no communications among processors until the sample mean comparisons at the last step), they are typically not efficient in total sample size. Fully sequential procedures are efficient in sample size. However, they make comparisons between approximately $k(k-1)/2$ possible pairs of alternatives (i.e., all pairwise comparisons) after each alternative in contention adds an observation. It requires frequent communications among the processors. Even worse, such comparisons can only be done on a single processor and often require an expected computational complexity of $\mathcal{O}(k^2)$ each time that is significantly larger than $\mathcal{O}(k)$, which is the expected computational complexity needed to generate an additional observation for each alternative in contention. When the number of alternatives is large, all pairwise comparisons may become the bottleneck of the procedures and delay the time to solve the problems.

Recently, some efforts have been made to make fully sequential procedures more suitable for parallel computing environments. As discussed in Hunter and Nelson (2017) and Ni et al. (2017), to reduce the number of comparisons in the procedures, avoiding global comparisons is critical. To achieve this, the authors propose two methods. One can partition the alternatives into small groups and only compare the alternatives that belong to the same group (i.e., the “divide and conquer” approach). In this case, by assigning different processors different groups of alternatives, the comparisons become parallelizable, and the expected computational complexity of comparisons in each processor only relates to the number of alternatives assigned to it. Alternatively, one can find high-precision estimates of an apparently good alternative and let it screen out inferior alternatives. In this case, the expected computational complexity of comparisons can be significantly reduced from $\mathcal{O}(k^2)$ to $\mathcal{O}(k)$ each time.

3.2. Procedure \mathcal{KN}_0

In this subsection, motivated by the knockout tournament, we propose a new framework of developing procedures to solve large-scale R&S problems in parallel computing environments. Conceptually, the procedure proceeds in rounds. At the beginning of each round $r \geq 1$, we pair up the alternatives that are still in contention. For each pair of alternatives, we conduct a match between the two alternatives using an existing R&S

procedure. The winner advances to the next round of comparisons. Following such a selection scheme, about half of the alternatives can be eliminated at each round, and the procedure can stop in $\lceil \log_2 k \rceil$ rounds, where $\lceil \cdot \rceil$ is the ceiling function. Therefore, instead of competing with all other $k-1$ alternatives as in existing R&S procedures, the best alternative only needs to win at each round or in total $\lceil \log_2 k \rceil$ alternatives in the procedure.

$$\mathcal{KN}_0(\{i, j\}, \alpha_r, \delta, \sigma^2)$$

Step 1 (Initialization). Compute

$$a_r = \frac{\log\lfloor 1/(2\alpha_r) \rfloor}{\delta} \text{ and } N_r = \left\lfloor \frac{4\sigma^2 a_r}{\delta} \right\rfloor + 1,$$

where $\lfloor \cdot \rfloor$ is the floor function. Simulate one observation $X_{i,1}$ and $X_{j,1}$ for alternatives i and j , respectively, and set $t = 1$.

Step 2 (Screening). Let

$$\begin{aligned} \text{eli Condition 1} &= Z_{ij}(t) - a_r + \frac{\delta t}{4\sigma^2} \text{ and} \\ \text{eli Condition 2} &= Z_{ji}(t) - a_r + \frac{\delta t}{4\sigma^2}. \end{aligned}$$

Step 3 (Stopping Rule).

- If *eli Condition 1* > 0 , let the procedure return alternative i . If *eli Condition 2* > 0 , let the procedure return alternative j .
- Else if $t = N_r$, stop and return the alternative that has the larger $\bar{X}_i(t)$.
- Otherwise, take an additional observation $X_{i,t+1}$ and $X_{j,t+1}$ from alternatives i and j , respectively; set $t = t + 1$; and go to Step 2.

To satisfy the PCS guarantee, we allocate the probability of incorrect selection (PICS) $\alpha_r = \alpha/2^r$ to each round r . The adopted procedure should ensure that for the match between alternative k and its opponent at round r , false elimination of alternative k happens with the probability less than the allocated PICS α_r . The reason that we allocate more PICS to earlier rounds is because there are more matches in the earlier rounds, and a larger PICS can bring more benefit in total sample size.

In this paper, we use Procedure \mathcal{KN} to conduct the matches because it typically provides the tightest continuation region among the fully sequential procedures and allows the early termination if one alternative is clearly better than the other. Let $\mathcal{KN}_0(\{i, j\}, \alpha_r, \delta, \sigma^2)$ denote the output of Procedure \mathcal{KN} , which can correctly select the best alternative between two alternatives i and j with probability at least $1 - \alpha_r$ if the difference in means is at least δ , and the variances of the alternatives are known and equal to σ^2 . We summarize the property of $\mathcal{KN}_0(\{i, j\}, \alpha_r, \delta, \sigma^2)$ as follows.

Lemma 1. For any two alternatives i and j , where $\mu_i - \mu_j \geq \delta$, we have

$$\mathbb{P}\left(\mathcal{KN}_0(\{i, j\}, \alpha_r, \delta, \sigma^2) \neq i\right) \leq \alpha_r.$$

Remark 2. The original Procedure \mathcal{KN} of Kim and Nelson (2001) is designed for the case of unknown and unequal variances. However, its statistical validity (i.e., Lemma 1) can be extended easily to the case of a common known variance. Therefore, we omit the proof of Lemma 1.

The detailed descriptions of the procedure are listed, and we call it Procedure \mathcal{KT}_0 , where \mathcal{KT} stands for knockout tournament.

Procedure \mathcal{KT}_0 (a common known variance)

Step 1 (Initialization). Select the PICS α ($0 < \alpha < 1/k$). Set parameter $\delta > 0$. Let \mathcal{I}_r be the set of alternatives in contention at the beginning of round r . Set $r=1$ and $\mathcal{I}_r = \{1, 2, \dots, k\}$.

Step 2 (Pair up). Let $\mathcal{I}_{r+1} = \emptyset$. Pair alternatives in \mathcal{I}_r . If there is an odd number of alternatives in \mathcal{I}_r , directly advance the leftover alternative to the next round of comparisons (i.e., include the index of the leftover alternative in \mathcal{I}_{r+1}).

Step 3 (Screening). For each pair of alternatives, namely, i and j , let $\alpha_r = \alpha/2^r$. Then, compute

$$\mathcal{I}_{r+1} = \mathcal{I}_{r+1} \cup \left\{ \mathcal{KN}_0(\{i, j\}, \alpha_r, \delta, \sigma^2) \right\}. \quad (8)$$

Step 4 (Stopping Rule). Set $r = r + 1$. If $|\mathcal{I}_r| = 1$, stop and select the alternative whose index is in \mathcal{I}_r as the best. Otherwise, go to Step 2.

Remark 3. In this paper, when pairing alternatives at each round, we randomly draw two alternatives at a time without replacement either from the set \mathcal{I}_r if the procedure is run on a single processor or as we will show later, from a subset of \mathcal{I}_r if the procedure is run in a parallel computing environment.

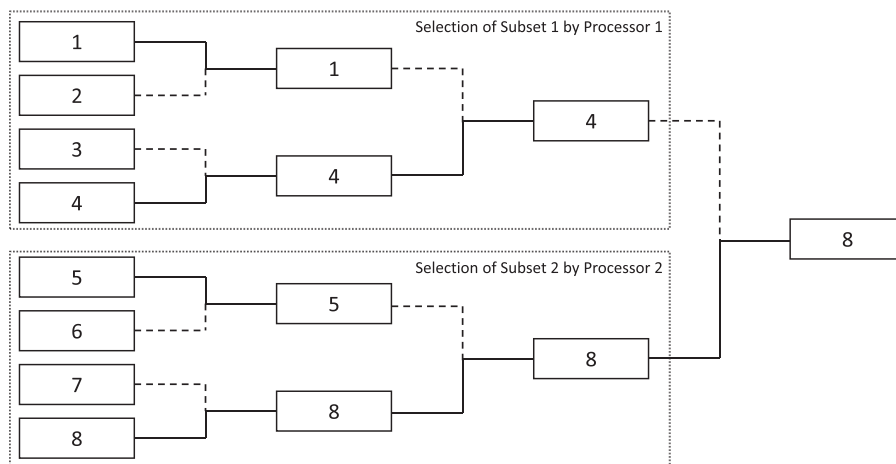
Procedure \mathcal{KT}_0 provides a general framework for conducting the selection. To practically run Procedure \mathcal{KT}_0 in a parallel computing environment, the implementation

details may vary from one parallel computing platform to another. In what follows, we give an intuitive explanation on how Procedure \mathcal{KT}_0 can be implemented in a parallel computing environment.

In Figure 3, we give an example of the selection of eight alternatives by Procedure \mathcal{KT}_0 in a parallel computing environment with two processors. The underlying mechanism works as follows. If there are m available processors in the parallel computing environment, one can equally divide the set of alternatives into m subsets at the very beginning of the selection process. Without any prior knowledge on the mean configuration, this division can be random. During the selection, we only (randomly) pair the alternatives that belong to the same subset. Then, similar to the tennis knockout-tournament example, the selections in different subsets can be independently conducted. Therefore, if each processor handles the selection of one subset, no communications and synchronizations are needed among the processors, and at any time, different processors can execute the matches at different rounds. After each subset decides a local best alternative, one can either continue the roundwise selection in a single processor or use a more efficient way, which will be discussed in Section 5, to finish the selection of these best alternatives. Therefore, like stagewise procedures, Procedure \mathcal{KT}_0 only requests for a minimal number of communications among the processors (i.e., a one-to-all broadcast at the very beginning of the selection process and an all-to-one reduction after every processor finds its own best alternative). At the same time, Procedure \mathcal{KT}_0 largely maintains the fully sequential feature, which allows the eliminations of the inferior alternatives at any time during the selection process.

However, there is a potential issue associated with this implementation scheme. As we assign the matches to the processors all at once at the beginning of the

Figure 3. An Illustration of Procedure \mathcal{KT}_0 in a Parallel Computing Environment



selection process, because of the random completion times of different matches, it can be the case that the processors that find their local best alternatives earlier need to wait for the last one to find. This can cause processor idling and thus, efficiency loss. For the common known variance case, to address this issue, one may have some initial sampling on the alternatives to estimate their means, rank and seed them based on the initial sample mean information (like typical tennis Grand Slam tournaments), and schedule the matches accordingly. By doing so, the procedure can better balance the workloads of different processors and may even achieve better practical performance on the total sample size. However, when the variances of the alternatives are unknown and unequal, and CRNs are used, not only the means but also, the variance of the difference between two alternatives can affect the time needed to conduct a match. To schedule the matches, one additionally needs to estimate the variance of the difference between any two alternatives. When the number of alternatives is very large, estimating these sample variances can take a very long time because its computational complexity is $\mathcal{O}(k^2)$, and storing these sample variances is challenging as well. Given these reasons, in this paper, we still choose to randomly pair alternatives at each round. We also want to note that for this implementation scheme, processor idling only occurs when a processor finds its local best alternative. It suggests that after a processor finishes its jobs, it will no longer be occupied by the program and can be used by other programs.

One may view the implementation scheme of Procedure \mathcal{KT}_0 as a variant of the “divide and conquer” approach because each processor handles the selection of one subset of alternatives. However, one distinct feature of Procedure \mathcal{KT}_0 is that, within each subset, the procedure still keeps the roundwise selection structure to gain further reductions on the computational complexity of comparisons and achieve a lower growth rate on the expected total sample size, instead of using Procedure \mathcal{KN} directly on all alternatives in the subset. In the next subsection, we show that Procedure \mathcal{KT}_0 is statistically valid and optimal in rate.

3.3. Statistical Validity and Upper-Bound Analysis

In this subsection, we first show that Procedure \mathcal{KT}_0 satisfies the PCS guarantee, summarized by the following theorem.

Theorem 2. *If Assumption 3 holds and $\mu_k - \mu_{k-1} \geq \delta$, with probability at least $1 - \alpha$, Procedure \mathcal{KT}_0 can select alternative k as the best.*

Proof. The proof of Theorem 2 is straightforward as the overall PICS of Procedure \mathcal{KT}_0 is simply a summation of the allocated PICS α_r at each round r . Let k_r ,

where $k_r \neq k$, denote the index of the alternative that competes with alternative k at round r .

With the result in Lemma 1, if $\mu_k - \mu_{k-1} \geq \delta$, we have

$$\begin{aligned} \mathbb{P}(\text{select alt. } k) &= \mathbb{P}\left(\bigcap_{r=1}^{\lceil \log_2 k \rceil} \{\text{alt. } k \text{ is selected at round } r\}\right) \\ &\geq 1 - \sum_{r=1}^{\lceil \log_2 k \rceil} \mathbb{P}(\text{alt. } k \text{ is eliminated at round } r) \\ &= 1 - \sum_{r=1}^{\lceil \log_2 k \rceil} \mathbb{P}(\mathcal{KN}_0(\{k, k_r\}, \alpha_r, \delta, \sigma^2) \neq k) \\ &> 1 - \sum_{r=1}^{\infty} \frac{\alpha}{2^r} = 1 - \alpha. \end{aligned}$$

It concludes the proof. \square

We prove in the following theorem that Procedure \mathcal{KT}_0 is optimal in rate.

Theorem 3. *Let \mathbf{N}_1 denote the total sample size that Procedure \mathcal{KT}_0 requires to identify the best alternative. Then, for any $k > 0$,*

$$\mathbf{N}_1 \leq \left(\frac{16\sigma^2 \log(2/\alpha)}{\delta^2} + 4 \right) k,$$

and therefore, $\mathbb{E}[\mathbf{N}_1] = \mathcal{O}(k)$.

Proof. Because at each round, about half of the alternatives are eliminated, it can be easily verified that the number of alternatives in contention at the beginning of round r i.e., $|\mathcal{I}_r|$, can be bounded as

$$2^{\lceil \log_2 k \rceil - r} \leq |\mathcal{I}_r| \leq 2^{\lceil \log_2 k \rceil - r + 1}.$$

The equation also implies that the total number of rounds is upper bounded by $\lceil \log_2 k \rceil$ i.e., $r \leq \lceil \log_2 k \rceil$, because as long as $|\mathcal{I}_r| \leq 2$, the procedure can produce the best alternative at round r . As $\mathcal{KN}_0(\{i, j\}, \alpha_r, \delta, \sigma^2)$ is applied at each round to compare alternatives, then at round r , for each alternative $i \in \mathcal{I}_r$, it can take at most

$$N_r = \left\lfloor \frac{4\sigma^2 a_r}{\delta} \right\rfloor + 1 = \left\lfloor \frac{4\sigma^2 [(r-1) \log 2 - \log \alpha]}{\delta^2} \right\rfloor + 1,$$

observations, where N_r is the ending point of the continuation region of Procedure \mathcal{KN} . Therefore, we can upper bound \mathbf{N}_1 as follows:

$$\begin{aligned} \mathbf{N}_1 &\leq \sum_{r=1}^{\lceil \log_2 k \rceil} |\mathcal{I}_r| \times N_r \\ &\leq \sum_{r=1}^{\lceil \log_2 k \rceil} 2^{\lceil \log_2 k \rceil - r + 1} \\ &\quad \times \left(\frac{4\sigma^2 [(r-1) \log 2 - \log \alpha]}{\delta^2} + 1 \right) \\ &\leq \left(\frac{8\sigma^2 \log 2}{\delta^2} \sum_{r=1}^{\infty} \frac{r-1}{2^r} - \frac{8\sigma^2 \log \alpha}{\delta^2} \sum_{r=1}^{\infty} \frac{1}{2^r} + 2 \sum_{r=1}^{\infty} \frac{1}{2^r} \right) \times 2^{\lceil \log_2 k \rceil} \\ &\leq \left(\frac{8\sigma^2 \log(2/\alpha)}{\delta^2} + 2 \right) \times 2^{\lceil \log_2 k \rceil} \\ &\leq \left(\frac{16\sigma^2 \log(2/\alpha)}{\delta^2} + 4 \right) k. \end{aligned}$$

Therefore, $\mathbb{E}[\mathbf{N}_1] = \mathcal{O}(k)$. This concludes the proof. \square

The results in Theorems 2 and 3 show that Procedure \mathcal{KT}_0 is statistically valid and optimal in rate. It suggests that in terms of the total sample size, the procedure is competitive for solving large-scale R&S problems, or at least, the total sample size of the procedure will not differ by orders of magnitude from that of the best procedure.

In addition to the number of communications and the total sample size, the number of comparisons is also an important aspect on evaluating a procedure’s performance in parallel computing environments. In Procedure \mathcal{KT}_0 , for each match, generating two observations (i.e., one for each alternative) is always coupled with just a single comparison. Therefore, the number of comparisons is $N_1/2$. Because the time to make a comparison is typically significantly less than that of generating two observations, the total comparison time is negligible compared with the simulation time. Moreover, the comparison is done right after collecting two observations on the same processor. Therefore, the comparisons are also distributed to different processors, which is different from that of Luo et al. (2015) where all comparisons are done in a single processor.

3.4. Procedure \mathcal{KT}_0 with the PAC Guarantee

As previously mentioned, the PCS guarantee may become inappropriate when the number of alternatives becomes large. In this subsection, inspired by the techniques used to design the median elimination procedure (Even-Dar et al. 2002) in the best arm identification literature, we show that with a slight modification on Procedure \mathcal{KT}_0 , it can satisfy the PAC guarantee, and the procedure’s expected total sample size remains to grow linearly in k . Let $\{\delta_r > 0 : r = 1, 2, \dots, \lceil \log_2 k \rceil\}$ denote a sequence of IZ parameters, which satisfy that $\delta_r \geq \frac{1}{3}(\frac{3}{4})^r \delta$, for $r = 1, 2, \dots, \lceil \log_2 k \rceil$, and $\sum_{r=1}^{\lceil \log_2 k \rceil} \delta_r = \delta$. We further define N'_1 as the total sample size of Procedure \mathcal{KT}_0 after the modification. We summarize the modification in the following proposition, and its proof is included in the e-companion.

Proposition 1. *If one replaces the match in (8) by*

$$\mathcal{I}_{r+1} = \mathcal{I}_{r+1} \cup \left\{ \mathcal{KN}_0 \left(\{i, j\}, \alpha_r, \delta_r, \sigma^2 \right) \right\} \quad (9)$$

and Assumption 3 holds, then, with probability at least $1 - \alpha$, Procedure \mathcal{KT}_0 can guarantee to select an alternative within δ to the best, and N'_1 satisfies that

$$N'_1 \leq \left(\frac{1152\sigma^2 \log(256/\alpha)}{\delta^2} + 4 \right) k.$$

Therefore, $N'_1 = \mathcal{O}(k)$.

Proposition 1 suggests that in order to alter Procedure \mathcal{KT}_0 to satisfy the PAC guarantee, other than the PICS α , we should allocate the tolerable error δ to

each round. By properly designing the allocation rule for δ , we can ensure that the procedure’s total sample size remains to grow linearly in k . However, to satisfy the more strict PAC guarantee, smaller IZ parameters are assigned to the matches. The procedure loses efficiency on the total sample size. One can observe that the constant on the upper bound of N'_1 is much larger than that of N_1 . In this paper, while allocating the tolerable error, we let $\delta_r = \frac{1}{3}(\frac{3}{4})^r$, for $r = 1, 2, \dots, \lceil \log_2 k \rceil - 1$, and $\delta_{\lceil \log_2 k \rceil} = \delta - \sum_{r=1}^{\lceil \log_2 k \rceil - 1} \frac{1}{3}(\frac{3}{4})^r \delta$.

4. Procedure for Unknown and Unequal Variances Case

In practice, the variances of all alternatives are typically unknown in advance. In this section, we extend our procedure to the case of unknown and unequal variances and make the following assumption on the observations of all alternatives.

Assumption 4. *For each alternative $i \in \mathcal{C}$, it generates i.i.d. observations $X_{i,\ell}$, for $\ell = 1, 2, \dots$, from the normal distribution with unknown mean μ_i and unknown variance σ_i^2 .*

To allow the use the CRNs, in Assumption 4, we do not particularly emphasize the independence between the observations generated from different alternatives. To revise Procedure \mathcal{KT}_0 to handle the case of unknown and unequal variances, we only need to adopt the unknown variance version of Procedure \mathcal{KN} to conduct matches in (8).

Let $\mathcal{KN}(\mathcal{C}, \alpha_r, \delta, n_0)$ denote the output of Procedure \mathcal{KN} with the first-stage sample size n_0 , which can correctly select the best alternative from the ones in set \mathcal{C} with probability at least $1 - \alpha_r$ if the difference in means between the best and the second-best alternatives in \mathcal{C} is at least δ , and the variances of the alternatives are unknown. We summarize the property of $\mathcal{KN}(\mathcal{C}, \alpha_r, \delta, n_0)$ as follows.

Lemma 2 (Kim and Nelson (2001)). *For a set of alternatives \mathcal{C} , if $k \in \mathcal{C}$ and the IZ assumption holds (i.e., $\mu_k - \max_{i \in \mathcal{C}, i \neq k} \mu_i \geq \delta$), then*

$$\mathbb{P}(\mathcal{KN}(\mathcal{C}, \alpha_r, \delta, n_0) \neq k) \leq \alpha_r.$$

The detailed descriptions of the procedure for the case of unknown and unequal variances are listed, and we call it Procedure \mathcal{KT} .

$\mathcal{KN}(\mathcal{C}, \alpha_r, \delta, n_0)$

Step 1 (Initialization). *For each alternative $i \in \mathcal{C}$, simulate n_0 observations $X_{i,1}, X_{i,2}, \dots, X_{i,n_0}$ from alternative i . Let*

$$h_r^2 = (n_0 - 1) \times \left[\left(\frac{2\alpha_r}{|\mathcal{C}| - 1} \right)^{-\frac{2}{n_0 - 1}} - 1 \right], \quad (10)$$

and for all $j \neq i$, compute

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

$$N_{r,i,j} = \frac{h_r^2 S_{ij}^2}{\delta^2}. \quad (11)$$

Let $N_{r,i} = \max_{j \neq i} N_{r,i,j}$ and $N_{r,\max} = \max_{i \in C} N_{r,i}$. If $n_0 > \lceil N_{r,\max} \rceil$, then stop and select the alternative with the largest $\bar{X}_i(t)$. Otherwise, set $t = n_0$ and go to Step 2.

Step 2 (Screening). Set $C^{\text{old}} = C$. Let

$$C = \left\{ i : i \in C^{\text{old}} \text{ and } t[\bar{X}_i(t) - \bar{X}_j(t)] \geq -\max \left\{ \frac{h_r^2 S_{ij}^2}{2\delta} - \frac{\delta t}{2}, 0 \right\}, \forall j \in C^{\text{old}}, j \neq i \right\}.$$

Step 3 (Stopping Rule).

- If $|C| = 1$, stop and select the alternative whose index is in C as the best.
- Else if $t = \lceil N_{r,\max} \rceil + 1$, stop and select the alternative whose index is in C having the largest $\bar{X}_i(t)$.
- Otherwise, take an additional observation $X_{i,t+1}$ from each alternative $i \in C$, set $t = t + 1$, and go to Step 2.

Procedure \mathcal{KT} (unknown and unequal variances)

Step 1 (Initialization). Select the PICS α ($0 < \alpha < 1/k$). Set parameter $\delta > 0$ and first-stage sample size n_0 . Let \mathcal{I}_r be the set of alternatives in contention at the beginning of round r . Set $r = 1$ and $\mathcal{I}_r = \{1, 2, \dots, k\}$.

Step 2 (Pair up). Let $\mathcal{I}_{r+1} = \emptyset$. Pair alternatives in \mathcal{I}_r . If there is an odd number of alternatives in \mathcal{I}_r , directly advance the leftover alternative to the next round of comparisons (i.e., include the index of the leftover alternative in \mathcal{I}_{r+1}).

Step 3 (Screening). For each pair of alternatives, namely, i and j , let $\alpha_r = \alpha/2^r$, $C = \{i, j\}$. Then, compute

$$\mathcal{I}_{r+1} = \mathcal{I}_{r+1} \cup \{\mathcal{KN}(C, \alpha_r, \delta, n_0)\}. \quad (12)$$

Step 4 (Stopping Rule). Set $r = r + 1$. If $|\mathcal{I}_r| = 1$, stop and select the alternative whose index is in \mathcal{I}_r as the best. Otherwise, go to Step 2.

In the R&S literature, the use of CRNs is for the purpose of generating positively correlated observations for the alternatives that are compared. It requires one to use one identical (pseudo-)random number stream to generate these observations. There are two major reasons that restrict existing parallel procedures from using CRNs to solve large-scale problems. First, to use CRNs, one has to calculate and store the sample variance information of the difference between any two alternatives that are compared. If the comparisons involve a large number of alternatives, it can incur a significant amount of computation and is a burden to the system’s memory as well. For example, if all pairwise comparisons are conducted and $k = 10^6$, then the

variance matrix of the size $10^6 \times 10^6$ is needed. It is difficult to compute and difficult to store in memory. Second, because of random completion times of generating different observations, in parallel computing environments, it may be difficult to guarantee that every alternative has the same number of observations whenever comparisons are conducted. The use of CRNs may fail in this situation. Because of these reasons, the parallel procedures proposed by Luo et al. (2015) and Ni et al. (2017) do not support the use of CRNs.

Compared with existing parallel procedures, one advantage of Procedure \mathcal{KT} is that it allows the practical use of CRNs in parallel computing environments to solve large-scale problems. In Procedure \mathcal{KT} , only the alternatives in a match are compared. Because we use Procedure \mathcal{KN} to conduct the matches’ and each match is executed by one processor, the use of CRNs in Procedure \mathcal{KT} is essentially equivalent to the use of CRNs in Procedure \mathcal{KN} in single-processor computing environments to solve small-scale problems. In Procedure \mathcal{KT} , to use CRNs, one can simply initiate a new random number stream for each match and use the random number stream to generate observations for the alternatives in the match. It is easy to implement.

To satisfy the PCS guarantee, in Procedure \mathcal{KT} , each match is a new match (i.e., no previous sample information can be used), and a first-stage sampling is always required in each match. In total, there are about $k - 1$ matches before the procedure selects the best alternative. Therefore, on average, each alternative only engages in approximately two matches.³ In Theorem 4, we show that Procedure \mathcal{KT} remains optimal in rate. The proof of Theorem 4 is included in the e-companion.

Theorem 4. If Assumption 4 holds and $\mu_k - \mu_{k-1} \geq \delta$, with probability at least $1 - \alpha$, Procedure \mathcal{KT} can select alternative k as the best. Moreover, let \mathbf{N}_2 denote the total sample size of Procedure \mathcal{KT} . If σ_{ij}^2 is upper bounded by a constant $\sigma_{\text{upper}}^2 > 0$ (i.e., $\sigma_{ij}^2 \leq \sigma_{\text{upper}}^2$) for all $i \neq j \in \mathcal{K}$ and $n_0 \geq 4$, then there exists a constant

$$\kappa_1 = \frac{16\sigma_{\text{upper}}^2(n_0 - 1)}{\alpha^{\frac{2}{n_0-1}}\delta^2} + 4(n_0 + 1),$$

such that for any $k > 0$, $\mathbb{E}[\mathbf{N}_2] \leq \kappa_1 k$.

Remark 4. Similar to Procedure \mathcal{KT}_0 , the communications in Procedure \mathcal{KT} occur at the beginning of the selection process and after every processor finds its own best alternative. The number of comparisons in the procedure is $\mathbf{N}_2/2$.

It is interesting to point out that our procedures are optimal in rate no matter whether the variances of the alternatives are known or not. As a comparison, for some traditional procedures, the expected total sample size grows faster for the case of unknown variances

than that of known variances. For instance, in Procedure \mathcal{KN} , the expected total sample size of each alternative grows proportionally to the ending point of the continuation region, and the ending point grows in the order of $\log k$ when the variances are known and in the order of k^{2/n_0-1} when the variances are unknown.

In Proposition 2, we show that with the same modification as that of Procedure \mathcal{KT}_0 , we can alter Procedure \mathcal{KT} to satisfy the PAC guarantee and keep the total sample size growing linearly in k . The proof of Proposition 2 is included in the e-companion.

Proposition 2. *Let N'_2 denote the total sample size of Procedure \mathcal{KT} after the modification. If Assumption 4 holds and one replaces the match in (12) by*

$$\mathcal{I}_{r+1} = \mathcal{I}_{r+1} \bigcup \{\mathcal{KN}(C, \alpha_r, \delta_r, n_0)\},$$

where δ_r is the IZ parameter assigned to the matches at round r and satisfies that $\delta_r \geq \frac{1}{3}(\frac{3}{4})^r \delta$ for $r = 1, 2, \dots, \lceil \log_2 k \rceil$, and $\sum_{r=1}^{\lceil \log_2 k \rceil} \delta_r = \delta$, then, with probability at least $1 - \alpha$, Procedure \mathcal{KT} can select an alternative within δ to the best. Moreover, if σ_{ij}^2 is upper bounded by a constant $\sigma_{upper}^2 > 0$, for all $i \neq j \in \mathcal{K}$, and $n_0 \geq 13$, then there exists a constant κ'_1 such that for any $k > 0$, $\mathbb{E}[N'_2] \leq \kappa'_1 k$.

5. Speeding up the Selection in Parallel Computing Environments

In Section 3.2, we have demonstrated that the selection structures of our knockout-tournament procedures fit the parallel computing environments well. However, to efficiently run the procedures in parallel computing environments, some additional modifications are needed. In this section, we propose two major modifications on Procedure \mathcal{KT} to make it more suitable for parallel computing environments.

First, in Procedure \mathcal{KT} , each match involves the competition between two alternatives. As the selection proceeds, it is possible that the two alternatives assigned to the same match are clearly inferior alternatives that have similar or even equal means. In this situation, the procedure needs to take a very long time and a large number of observations to distinguish them. It can significantly slow down the selection process. To resolve this issue, we suggest properly enlarging the size of the match. By allowing more alternatives to participate in a match, we lower the chance that all these alternatives are equally bad. As long as there is a “good” alternative, all other clearly inferior alternatives can be quickly eliminated. However, with more alternatives in a match, the ending point of the continuation region in Procedure \mathcal{KN} becomes larger. Procedure \mathcal{KN} needs more observations to find the best alternative in the worst case scenario (i.e., worse performance in the worst case scenario). Second, as we have

mentioned in Section 3.2, while implementing our procedures in parallel computing environments, we always face the problem of selecting the best from m alternatives in the final step. When the number of processors m is relatively small, a single processor can quickly find the final best. However, when the number of processors m is very large, using a single processor to conduct the selection can cause other processors to stay idle for a long time. It could potentially make the procedure inefficient. In the following Procedure \mathcal{KT}^+ , we offer a way to address this issue so that one can make full use of all the processors in the final step selection. Basically, after each processor finds a local best alternative, we use a stagewise procedure (without CRNs) so that each local best alternative simulates enough observations. We then select the alternative with the largest sample mean from this set of local best alternatives. In the following procedure, we suggest using Rinott’s procedure for the task.

5.1. The Procedure

In this subsection, we consider the case where there are m processors in the parallel computing environments and present the detailed descriptions of the procedure. We call it Procedure \mathcal{KT}^+ .

Procedure \mathcal{KT}^+ (parallel implementation)

Step 1 (Initialization). *Select the PICS α ($0 < \alpha < 1/k$). Set parameter $\delta > 0$, first-stage sample size n_0 , number of processors m , and number of alternatives $g \geq 2$ within a match. Let \mathcal{I}_r^s be the set of alternatives in contention at the beginning of round r in processor s for $s = 1, 2, \dots, m$.*

Step 2 (Assigning Alternatives). *Equally allocate k alternatives to m processors so that each processor handles the selection of approximately k/m alternatives: for example, for $i = 1, 2, \dots, k$, let*

$$\mathcal{I}_1^{(i \bmod m)+1} = \mathcal{I}_1^{(i \bmod m)+1} \bigcup \{i\}.$$

Step 3 (Parallel Selection). *Execute the selection in processor $s = 1, 2, \dots, m$: set $r = 1$,*

a. *Let $\mathcal{I}_{r+1}^s = \emptyset$. Group alternatives in \mathcal{I}_r^s with the size of g . In case of leftover ones, let them form a group. After grouping, there are in total $\lceil |\mathcal{I}_r^s| / g \rceil$ groups. Let $\mathcal{I}_{r,q}^s$ denote the set of the alternatives in group q for $q = 1, 2, \dots, \lceil |\mathcal{I}_r^s| / g \rceil$ of processor s at round r .*

b. *Let $\alpha_r = \alpha/2^r$. For each group $q = 1, 2, \dots, \lceil |\mathcal{I}_r^s| / g \rceil$, set $\mathcal{C} = \mathcal{I}_{r,q}^s$, and compute*

$$\mathcal{I}_{r+1}^s = \mathcal{I}_{r+1}^s \bigcup \{\mathcal{KN}(C, \alpha_r, \delta, n_0)\}. \quad (13)$$

c. *Set $r = r + 1$. If $|\mathcal{I}_r^s| = 1$, let I_s denote the index of the alternative in \mathcal{I}_r^s . Otherwise, go to Step 3(a).*

d. *Take n_0 observations from alternative I_s . Calculate its sample variance $S_{I_s}^2$ based on these n_0 observations. Set $r = \lceil \log_{\frac{k}{m}} \frac{k}{m} \rceil + 1$, $\alpha_r = \alpha/2^r$, and $h(\alpha_r, m, n_0)$, where $h(\alpha_r, m, n_0)$ is the Rinott’s constant determined by α_r , m , and n_0 .*

Then, take additional

$$\max\left\{0, \left[\left(\frac{h(\alpha_r, m, n_0)S_{I_s}}{\delta}\right)^2\right] - n_0\right\} \quad (14)$$

observations from alternative I_s .

Step 4 (Selecting the Best). Let \mathcal{I}_{final} denote the set of alternatives containing all the best alternatives produced by m processors. Select the alternative with the largest sample mean based on all the observations we take in Step 3(d) as our final output.

It can be seen that, in Procedure \mathcal{KT}^+ , each processor performs the selection for a subset of alternatives. After every processor produces a local best alternative, only m alternatives are left. As in the final step selection, Rinott's procedure is applied; in Step 3(d), the procedure generates enough observations for each local best alternative as Rinott's procedure requires, and in Step 4, one can simply pick up the alternative with the largest sample mean as the best. In Procedure \mathcal{KT}^+ , we still define α_r as $\alpha/2^r$ instead of α/g^r because if $g > 2$, a relatively large amount of PICS is left unused.

5.2. Statistical Validity and Upper-Bound Analysis

To prove that Procedure \mathcal{KT}^+ is statistically valid, we first summarize the property of the Rinott's procedure as follows.

Lemma 3 (Rinott 1978). Suppose that $k \in \mathcal{I}_{final}$ and $\mu_k - \mu_j \geq \delta$ for all $j \in \mathcal{I}_{final}$ and $j \neq k$. If each alternative $i \in \mathcal{I}_{final}$ has more than

$$\max\left\{n_0, \left[\left(\frac{h(\alpha_r, m, n_0)S_i}{\delta}\right)^2\right]\right\}$$

observations, then with probability at least $1 - \alpha_r$, alternative k has the largest sample mean.

Then, we can show that Procedure \mathcal{KT}^+ inherits the statistical validity from Procedure \mathcal{KT} .

Analyzing the growth rate of the expected total sample size of Procedure \mathcal{KT}^+ is not straightforward because the constant $h(\alpha_r, m, n_0)$ used in Rinott's procedure is a solution to a function that can only be numerically solved. It may be hard to quantify the behavior of $h(\alpha_r, m, n_0)$ with respect to the changes of α_r , m , and n_0 . In Lemma 4, we find an upper bound on the constant $h(\alpha_r, m, n_0)$. The upper bound is of the same order as that of the constant h_r used in Procedure \mathcal{KN} .

Lemma 4. Let $h(\alpha_r, m, n_0)$ be the constant in Rinott's procedure determined by α_r , m , and n_0 . Then,

$$h(\alpha_r, m, n_0) \leq \left\{2(n_0 - 1) \left[\left(\frac{2(m-1)^{\frac{2}{n_0-1}}}{\alpha_r} - 1 \right) \right] \right\}^{1/2}.$$

With the results in Lemma 4, we can proceed our analysis on the growth rate of the expected total

sample size of Procedure \mathcal{KT}^+ . We summarize the analysis results by the following Theorem 5, and its proof is included in the e-companion.

Theorem 5. If Assumption 4 holds and $\mu_k - \mu_{k-1} \geq \delta$, with probability at least $1 - \alpha$, Procedure \mathcal{KT}^+ can select alternative k as the best. Moreover, let \mathbf{N}_3 denote the total sample size of Procedure \mathcal{KT}^+ . If σ_i^2 is upper bounded by the constant $\sigma^2 > 0$ for all $i \in \mathcal{K}$, σ_{ij}^2 is upper bounded by the constant $\sigma_{upper}^2 > 0$, for all $i \neq j \in \mathcal{K}$, $g \geq 2$, $m \geq 1$, and $n_0 \geq 4$, then, $\mathbb{E}[\mathbf{N}_3] = \mathcal{O}(k)$.

Even though Procedure \mathcal{KT}^+ remains optimal in rate, the constant on the growth rate of the expected total sample size is now additionally depending on the number of alternatives g in a match and the number of processors m .

In general, the constant on the upper bound increases as one sets a larger value for g because of the worse performance of Procedure \mathcal{KN} in the worst case scenario in each match. When $g = 2$, the procedure has the tightest upper bound and performs the best asymptotically, but it may be the case that the procedure will never show its superior performance until k is much larger than the size of the problem we can encounter in practice. Therefore, to balance the procedure's finite-time and asymptotic performances, we suggest setting $g > 2$. Based on the results of our numerical experiments, we suggest having fewer than 30 alternatives in a match when the size of the problem is relatively small (e.g., $k \leq 10^5$) and 50 alternatives when $k \geq 10^5$.

In Theorem 5, we consider the situation where the number of processors in the parallel computing environment does not change (i.e., m is a constant). In practice, the number of processors used to solve a problem may vary. Taking this into consideration, the growth rate of Procedure \mathcal{KT}^+ may also increase with m . In the extreme case where $m = k$, the procedure degenerates to Rinott's procedure, and the growth rate of the procedure is the same as that of Rinott's procedure. However, it is arguable that the impact of such loss on the wall clock time is limited. As we look at the number of observations simulated by each processor, it decreases whenever m increases because fewer alternatives are assigned to each processor. It suggests that for Procedure \mathcal{KT}^+ , from the aspect of reducing the simulation time in each processor, using more processors is always beneficial. Also, when the number of processors is very large (compared with the size of the problem), in each processor, the computation cost for simulations accounts for only a small fraction of total computation cost. In this situation, the difference in the total sample size may only slightly affect the wall clock time.

In Procedure \mathcal{KT}^+ , it can be checked that approximately $g(g-1)/2$ comparisons are associated with generating g observations within each match. Thus, the

number of comparisons for Procedure \mathcal{KT}^+ is roughly $(g-1)\mathbf{N}_3/2$. Because g is small compared with k , one can treat the growth rates of the number of comparisons and the total sample size in Procedure \mathcal{KT}^+ the same. Also, CRNs still can be used in each match in Procedure \mathcal{KT}^+ without much difficulty.

5.3. Procedure \mathcal{KT}^+ with the PAC Guarantee

Different from Procedure \mathcal{KT}_0 and Procedure \mathcal{KT} , to convert Procedure \mathcal{KT}^+ to satisfy the PAC guarantee, Procedure \mathcal{KN} cannot be used in each match. When $g = 2$, at round r , Procedure \mathcal{KN} conducts the selection either in the situation where the IZ assumption holds or in the situation where the difference in means between the two alternatives is less than δ_r , in which case selecting any one of them is acceptable. When $g > 2$, however, it can be the case that within a match, some alternatives are within δ_r to the best, and the others are not. In this situation, we need to use a procedure that can satisfy the PAC guarantee to conduct the selection. As shown by Kao and Lai (1980), with a slight modification on Paulson's procedure (Paulson 1964) (another well-known fully sequential procedure), it can satisfy the PAC guarantee. We let $\mathcal{P}\text{-}\mathcal{PAC}(\mathcal{C}, \alpha, \delta, n_0)$ denote the output of the modified Paulson's procedure with the first-stage sample size n_0 , which can select an alternative within δ to the best alternative in \mathcal{C} with probability at least $1 - \alpha$. In the e-companion, we provide detailed descriptions for the selection in $\mathcal{P}\text{-}\mathcal{PAC}(\mathcal{C}, \alpha, \delta, n_0)$. We summarize the property of $\mathcal{P}\text{-}\mathcal{PAC}(\mathcal{C}, \alpha, \delta, n_0)$ as follows:

Lemma 5 (Kao and Lai (1980)). *For a set of alternatives \mathcal{C} , if $\mathcal{P}\text{-}\mathcal{PAC}(\mathcal{C}, \alpha, \delta, n_0) = \mathfrak{I}_P$, we have*

$$\mathbb{P}\left(\max_{i \in \mathcal{C}} \mu_i - \mu_{\mathfrak{I}_P} \leq \delta\right) \geq 1 - \alpha.$$

In the following proposition, we show the modifications we have made on Procedure \mathcal{KT}^+ to alter it to satisfy the PAC guarantee, and its proof is included in the e-companion. Because Rinott's procedure can automatically satisfy the PAC guarantee, we keep using it in the final step selection.

Proposition 3. *Let \mathbf{N}'_3 denote the total sample size of Procedure \mathcal{KT}^+ after modifications. If Assumption 4 holds and one replaces (13) and (14) in Procedure \mathcal{KT}^+ by*

$$\mathcal{I}_{r+1}^s = \mathcal{I}_{r+1}^s \bigcup \left\{ \mathcal{P}\text{-}\mathcal{PAC}(\mathcal{C}, \alpha_r, \delta_r, n_0) \right\} \text{ and} \\ \max\left\{ 0, \left\lceil \left(\frac{h(\alpha_r, m, n_0) S_{I_s}}{\delta_r} \right)^2 \right\rceil - n_0 \right\},$$

respectively, where δ_r is the IZ parameter assigned to the matches at round r and satisfies that $\delta_r \geq \frac{1}{3} \left(\frac{2}{3}\right)^r$ for $r = 1, 2, \dots, \lceil \log_g \frac{k}{m} \rceil + 1$, and $\sum_{r=1}^{\lceil \log_g \frac{k}{m} \rceil + 1} \delta_r = \delta$, then, with probability at least $1 - \alpha$, the procedure can select an alternative

within δ to the best. Moreover, if σ_{ij}^2 is upper bounded by a constant $\sigma_{upper}^2 > 0$, for all $i \neq j \in \mathcal{K}$, and $n_0 > 2 \log 2 (2 \log 3 + \log g - 4 \log 2) + 1$, then $\mathbb{E}[\mathbf{N}'_3] = \mathcal{O}(k)$.

In the rest of this paper, we call the procedure after modifications Procedure $\mathcal{KT}^+\text{-PAC}$. To efficiently run Procedure $\mathcal{KT}^+\text{-PAC}$ in parallel computing environments, we recommend setting g between 50 and 100, which is larger than that of Procedure \mathcal{KT}^+ .

6. Numerical Experiments

In this section, we examine the performance of the proposed procedures. The objectives of the numerical experiments are threefold: (1) to illustrate that the expected total sample size of our procedures indeed grows linearly in k ; (2) to demonstrate that CRNs can be easily applied to our procedures and our procedures can benefit from them significantly; and (3) to compare Procedure \mathcal{KT}^+ with the good selection procedure (GSP) proposed by Ni et al. (2017), aiming to shed some light on the performance of the procedures in parallel computing environments.

For all the numerical experiments reported in this section, the desired PICS α and the IZ parameter δ are set to be 0.05 and 0.1, respectively. In the cases of unknown variances, 20 observations are simulated for every alternative to estimate the sample variances (i.e., $n_0 = 20$). For the experiments in the first two subsections, we consider a simple setting where the observations are generated from normal distributions, and for the experiments in the last subsection, the observations are obtained by conducting real simulation tasks. All codes used in this section can be retrieved from <https://github.com/biazhong/KT>.

6.1. The Case with a Common Known Variance

In this subsection, we use a simple example to test growth rates of the expected total sample size of Procedure \mathcal{KT}_0 and Procedure \mathcal{KN} . Specifically, we consider a problem where all alternatives share a common known variance 1 (i.e., $\sigma_1^2 = \sigma_2^2 = \dots = \sigma_k^2 = \sigma^2 = 1$), and the means of the alternatives follow the slippage configuration (i.e., $\mu_1 = \mu_2 = \dots = \mu_k - \delta$). The observations simulated by different alternatives are independent. We conduct this experiment on a single processor. When the variances of all alternatives are known and equal, conducting all pairwise comparisons in Procedure \mathcal{KN} at each round is equivalent to letting every alternative only compare with the one with the largest sample mean. Therefore, the number of comparisons for Procedure \mathcal{KN} can be significantly reduced in this particular example. Furthermore, simulating a normal observation takes almost no time. Because of these reasons, both procedures can handle a relatively large number of alternatives even on a single processor. We

Table 1. Comparisons of Procedure \mathcal{KT}_0 and Procedure \mathcal{KN} with Different Numbers of Alternatives

k	Procedure \mathcal{KT}_0		Procedure \mathcal{KN}	
	PCS	Average total sample size	PCS	Average total sample size
10^2	0.963	1.462×10^5 $\pm 0.003 \times 10^5$	0.978	0.884×10^5 $\pm 0.008 \times 10^5$
10^3	0.965	1.512×10^6 $\pm 0.001 \times 10^6$	0.989	1.165×10^6 $\pm 0.008 \times 10^6$
10^4	0.956	1.527×10^7 $\pm 0.001 \times 10^7$	0.994	1.419×10^7 $\pm 0.009 \times 10^7$
10^5	0.955	1.528×10^8 $\pm 0.001 \times 10^8$	0.994	1.686×10^8 $\pm 0.008 \times 10^8$
10^6	0.956	1.528×10^9 $\pm 0.001 \times 10^9$	0.993	1.990×10^9 $\pm 0.008 \times 10^9$

test the performance of Procedure \mathcal{KT}_0 and Procedure \mathcal{KN} with different numbers of alternatives k . We summarize the estimated PCS and the average total sample size with 95% confidence interval based on 1,000 independent macro replications in Table 1. To better demonstrate our results, we also plot the average total sample size per alternative (average total sample size/ k) against the number of alternatives k in Figure 4.

From the results in Figure 4 and Table 1, we have the following conclusions. First, Procedure \mathcal{KN} requires a smaller average total sample size than Procedure \mathcal{KT}_0 does when the number of alternatives is small (i.e., $k \leq 10^4$). However, as the number of alternatives increases, the average total sample size of Procedure \mathcal{KN} grows faster than that of Procedure \mathcal{KT}_0 and eventually surpasses that of Procedure \mathcal{KT}_0 when $k \geq 10^5$. It indicates that our procedure is suitable for solving large-scale R&S problems in terms of the total sample size. Second, Procedure \mathcal{KN} delivers a higher PCS than

Procedure \mathcal{KT}_0 does in this example. This is because Procedure \mathcal{KN} makes full use of the sample information on the alternatives during the whole selection process. Meanwhile, Procedure \mathcal{KT}_0 completely abandons previous sample information. Therefore, if the total sample sizes of both procedures are close, Procedure \mathcal{KN} has a higher PCS. However, Procedure \mathcal{KT}_0 can still provide the PCS guarantee even under the least favorable configuration of the means (i.e., the slippage configuration). Third, from Figure 4, we can see that as the number of alternatives increases, the average total sample size per alternative of Procedure \mathcal{KT}_0 remains constant. It suggests that the linear growth rate on the expected total sample size is indeed attainable for our procedure.

6.2. The Use of CRNs

In this numerical experiment, we focus on investigating how our procedure may benefit from using CRNs. We assume that the variances of all alternatives are equal (i.e., $\sigma_1^2 = \sigma_2^2 = \dots = \sigma_k^2 = 1$) but unknown. The observations simulated by any two alternatives are correlated, and the correlation is ρ . Procedure \mathcal{KT} is adopted to solve this problem. Because, in practice, when the number of alternatives is large, the means of alternatives are often spread out over a wide range; in this example, for each macro replication, we consider a mean configuration such that

$$\mu_i = \begin{cases} [\text{unif}(0, 15)]\delta & \text{for } i = 1, 2, \dots, k-1 \\ 16\delta & \text{for } i = k \end{cases}$$

where $\text{unif}(0, 15)$ is the uniform distribution with the support $[0, 15]$. We test the performance of Procedure \mathcal{KT} with different numbers of alternatives k and different values of ρ . Similar to the previous experiment, we implement the procedure on a single processor and report the estimated PCS and average total sample size with 95% confidence interval based on 1,000 independent macro replications in Table 2.

We highlight the main findings from this experiment as follows. First, as the correlation ρ increases, we observe a steady decrease on the average total sample size (see each row). Particularly, in this example, every 0.25 increase on ρ leads to about 25% decrease on the average total sample size. When $\rho = 0.75$, the procedure only requires an average total sample size, which is only about 25% as that of $\rho = 0$. It suggests that even though our procedure may experience the conservativeness caused by having several equally bad alternatives in a match and abandoning previous sample information, the use of CRNs can help us greatly offset such efficiency loss. Second, comparing the estimated PCS at each row, we may conclude that the use of CRNs does not affect the PCS. Third, in this case, we observe a slight increase on the total sample size per alternative (see each column). However, as k increases,

Figure 4. (Color online) Growth Rate of Total Sample Size per Alternative

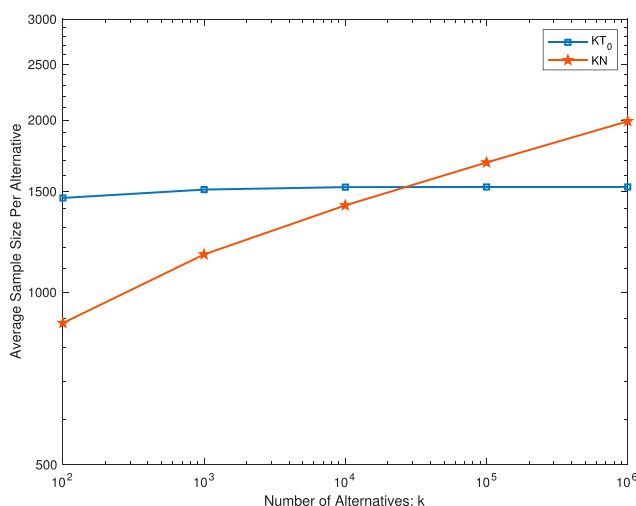


Table 2. Performance Test for Procedure \mathcal{KT} with Different Numbers of Alternatives and Different Values of ρ

k	Procedure \mathcal{KT}							
	$\rho = 0$		$\rho = 0.25$		$\rho = 0.50$		$\rho = 0.75$	
	PCS	Average total sample size	PCS	Average total sample size	PCS	Average total sample size	PCS	Average total sample size
10^2	0.998	8.156×10^4 $\pm 0.046 \times 10^4$	0.996	6.142×10^4 $\pm 0.033 \times 10^4$	0.996	4.111×10^4 $\pm 0.023 \times 10^4$	0.995	2.091×10^4 $\pm 0.011 \times 10^4$
10^3	0.994	8.885×10^5 $\pm 0.016 \times 10^5$	0.995	6.673×10^5 $\pm 0.012 \times 10^5$	0.992	4.459×10^5 $\pm 0.007 \times 10^5$	0.994	2.260×10^5 $\pm 0.004 \times 10^5$
10^4	0.989	9.124×10^6 $\pm 0.005 \times 10^6$	0.990	6.854×10^6 $\pm 0.004 \times 10^6$	0.995	4.584×10^6 $\pm 0.003 \times 10^6$	0.994	2.321×10^6 $\pm 0.001 \times 10^6$
10^5	0.991	9.155×10^7 $\pm 0.002 \times 10^7$	0.992	6.877×10^7 $\pm 0.001 \times 10^7$	0.992	4.597×10^7 $\pm 0.001 \times 10^7$	0.987	2.328×10^7 $\pm 0.001 \times 10^7$
10^6	0.990	9.160×10^8 $\pm 0.001 \times 10^8$	0.993	6.882×10^8 $\pm 0.001 \times 10^8$	0.992	4.609×10^8 $\pm 0.001 \times 10^8$	0.991	2.328×10^8 $\pm 0.001 \times 10^8$

the rate of increase approaches to zero. It implies that the total sample size per alternative may be upper bounded as k increases, and the linear growth rate on the expected total sample size remains intact.

6.3. The Three-Stage Buffer Allocation Problem

In this subsection, we test the performance of Procedure \mathcal{KT}^+ and an existing parallel procedure GSP proposed by Ni et al. (2017) with the three-stage buffer allocation problem in parallel computing environments.

GSP is by far the most efficient parallel procedure used to solve large-scale problems in the R&S literature and satisfies the PAC guarantee. GSP is a fully sequential procedure. GSP uses the “divide and conquer” approach that we discussed in Section 3.1 to reduce the comparison time. Also, GSP requires each alternative to simulate a batch of observations to reduce the frequency of communications among the processors, and carefully determines batch sizes of different alternatives to balance the workloads of different processors. These strategies used in GSP make it suitable for parallel computing environments. Codes for GSP used in this subsection are retrieved from <https://bitbucket.org/ericni/mapredms> and <https://bitbucket.org/ericni/sparkrns>.

The three-stage buffer allocation problem is a practical simulation testing problem and is often used to compare the practical performance of different procedures. The problem considers a flow line with three stations 1, 2, and 3. An infinite number of jobs wait in front of station 1. Each job is sequentially processed by the three stations. The service times at stations 1, 2, and 3 are independently drawn from exponential distributions with service rates s_1 , s_2 , and s_3 , respectively. In front of stations 2 and 3, there are finite numbers of buffer storage locations, denoted as b_2 and b_3 . If the buffer of station i , for $i = 2, 3$, is fully occupied, then station $i - 1$ is blocked, and it must hold the finished job until the job at station i is finished and released.

Our objective is to find an allocation of buffer and service rates with certain constraints to maximize the steady-state throughput of the flow line. The mathematical formulation of this problem is listed as follows:

$$\begin{aligned} & \max_x \mathbb{E}[f(x; \xi)] \\ & \text{s.t. } s_1 + s_2 + s_3 = \mathcal{L}_1 \\ & \quad b_2 + b_3 = \mathcal{L}_2 \\ & \quad x = (s_1, s_2, s_3, b_2, b_3) \in \mathcal{Z}_+^5, \end{aligned}$$

where \mathcal{L}_1 and \mathcal{L}_2 are the problem parameters that specify the set of feasible solutions and $f(x; \xi)$ is the random throughput of the flow line. For every feasible solution, we obtain observations of $f(x; \xi)$ by running simulation experiments. For each simulation experiment, we warm up the system with 2,000 jobs. After 2,000 jobs are processed, we observe the throughput of the subsequent 50 jobs.

We consider three problems with different sizes of feasible solutions by setting different values for \mathcal{L}_1 and \mathcal{L}_2 . Because the service times are exponentially distributed, we can analytically calculate $\mathbb{E}[f(x; \xi)]$ for each feasible solution by solving the balance equations for the underlying Markov chain from Buzacott and Shanthikumar (1993). Hence, we are able to provide some detail information on these problems in Table 3. One can observe that, for all three problems, there are multiple alternatives whose means lie within δ from the best. The IZ assumption is violated in these problems. Therefore, in the rest of the experiments, instead of PCS, we report the estimated PAC probability (i.e., the probability of

Table 3. Summarized Information on Three-Stage Buffer Allocation Problems

$(\mathcal{L}_1, \mathcal{L}_2)$	Number of alternatives: k	Highest mean: μ_k	Number of alternatives within: $[\mu_k - \delta, \mu_k]$
(20, 20)	3,249	5.78	21
(50, 50)	57,624	15.70	43
(128, 128)	1,016,127	41.66	97

selecting an alternative within δ to the best) of different procedures.

In this subsection, we focus on implementing the procedures on Apache Hadoop and Apache Spark. They are two parallel computing platforms that are widely supported by popular commercial clouds including Amazon EC2/S3, Microsoft Azure, Google Cloud Platform (GCP), etc. To make a fair comparison, while running the procedures, the configurations of the parallel computing environments are the same. For GSP, as its performance additionally depends on the number of simulation observations each alternative can take at a time, we set the batch-size parameter $\beta = 100$ as recommended by the authors. Except for the first-stage sample size n_0 , all other parameters for the procedure are the same as the ones used in Ni et al. (2017). For Procedure \mathcal{KT}^+ , CRNs are used while solving these problems. Specifically, we let all alternatives in a match use one identical random number stream to generate observations. In this situation, the observations generated by the alternatives are positively correlated. For GSP and Procedure \mathcal{KT}^+ , both simulations and comparisons are done locally within each processor. Similar to Ni et al. (2017), we use *utilization* to measure how efficiently these procedures use processors to simulate observations and make comparisons. We calculate it as follows:

$$\text{Utilization} = \frac{\text{total simulation time} + \text{total comparison time}}{\text{number of processors} \times \text{wall clock time}},$$

where the total simulation time and the total comparison time are summed over all processors.

6.3.1. Experiments on a Local Server. We first use GSP and Procedure \mathcal{KT}^+ to solve the problem with 3,249 alternatives on a local server with 48 available processors, 128 Gigabytes (GB) of memory, and Linux Red Hat Enterprise 7.4 operating system. As a reference, we also include the results of Procedure $\mathcal{KT}^+ - \mathcal{PAC}$, Rinott’s procedure, and a standard knockout-tournament procedure in this experiment. The standard knockout-tournament procedure has the same selection structure as that of Procedure \mathcal{KT}_0 , except that in every match, it does not use Procedure \mathcal{KN} to compare alternatives. It simply takes a fixed number τ of observations for each alternative and picks the alternative with the largest sample mean to the next round.⁴ We call the procedure Procedure \mathcal{KT} -EA, where EA stands for equal allocation. It should be noted that, among these five procedures, GSP, Procedure $\mathcal{KT}^+ - \mathcal{PAC}$, and Rinott’s procedure satisfy the PAC guarantee. Procedure \mathcal{KT}^+ only satisfies the PCS guarantee, and Procedure \mathcal{KT} -EA is a heuristic procedure. For Procedure \mathcal{KT}^+ and Procedure $\mathcal{KT}^+ - \mathcal{PAC}$, we set the number of alternatives in every match g to be 20 and 50, respectively. For Procedure \mathcal{KT} -EA, we set $\tau = 45$ so that it can solve the problem with roughly the same wall clock time as that of Procedure \mathcal{KT}^+ . We estimate all interested statistics based on 100 independent macro replications. The results are summarized in Table 4.

From Table 4, we can draw following conclusions. First, except for Procedure \mathcal{KT} -EA, all the procedures numerically demonstrate the ability to select an

Table 4. A Comparison of GSP, Procedure \mathcal{KT}^+ , Procedure $\mathcal{KT}^+ - \mathcal{PAC}$, Rinott’s Procedure, and Procedure \mathcal{KT} -EA on the Problem with 3,249 Alternatives: $m = 48$

Procedure	PAC	Total sample size ($\times 10^5$)	Wall clock time (seconds)	Total simulation time (seconds)	Total comparison time (seconds)	Utilization (%)
Hadoop						
GSP	1.00	5.36	852.31	387.31	0.31	0.95
Procedure \mathcal{KT}^+	1.00	2.07	68.29	101.53	0.08	3.10
Procedure $\mathcal{KT}^+ - \mathcal{PAC}$	1.00	8.91	100.71	421.73	0.29	8.73
Rinott’s procedure	1.00	22.96	100.77	1159.48	0.00	23.97
Procedure \mathcal{KT} -EA	0.88	2.90	69.76	138.07	0.00	4.12
Spark						
GSP	1.00	5.41	15.09	373.83	0.39	51.67
Procedure \mathcal{KT}^+	1.00	2.06	6.03	163.69	0.15	56.61
Procedure $\mathcal{KT}^+ - \mathcal{PAC}$	1.00	9.02	40.45	597.88	0.25	30.81
Rinott’s procedure	1.00	22.62	38.99	1573.61	0.00	84.08
Procedure \mathcal{KT} -EA	0.89	2.90	6.25	216.65	0.00	72.21

alternative within δ to the best. Comparing Procedure \mathcal{KT}^+ and Procedure \mathcal{KT} -EA, we may conclude that using Procedure \mathcal{KN} to conduct the matches and the modifications we have made on Procedure \mathcal{KT}^+ are indeed necessary to ensure the quality of the final output. Second, in this experiment, when we run the procedures on Apache Hadoop, the processor utilizations are low. This is because, on Apache Hadoop, the communication cost among the processors is considerably high. Because this problem does not need these procedures to simulate many observations, most of the wall clock time is spent on communications. Compared with the other four procedures, GSP needs more frequent communications among the processors. Therefore, in terms of the wall clock time, the performance of GSP is not satisfactory on Apache Hadoop. Third, the total sample size of Procedure \mathcal{KT}^+ - \mathcal{PAC} is much larger than those of GSP and Procedure \mathcal{KT}^+ . It suggests that as we alter Procedure \mathcal{KT}^+ to theoretically satisfy the PAC guarantee, the procedure loses a significant amount of efficiency on the total sample size. Even though the total sample size of Procedure \mathcal{KT} - \mathcal{PAC} theoretically remains to grow linearly in k , the constant on the upper bound is too large to allow the procedure to translate its theoretical advantage into good practical performance for reasonable sizes of problems. However, Procedure \mathcal{KT}^+ - \mathcal{PAC} still requires fewer observations than the stagewise procedure, Rinott's procedure, does. Because of the inefficiency of the total sample size of Procedure \mathcal{KT}^+ - \mathcal{PAC} and Rinott's procedure, and the failure of ensuring the quality of the final output of Procedure \mathcal{KT} -EA, in the rest of this subsection, we focus on comparing GSP and Procedure \mathcal{KT}^+ .

Under the same setting as the previous experiment, we then run GSP and Procedure \mathcal{KT}^+ to solve the problem with 57,624 alternatives. We report the statistics based on 10 independent macro replications. Because both procedures can always make a "good selection," we no longer report the estimated PAC in the rest of the experiments. In Table 5, similar results to those in the previous experiment can be found. It is clear that as the problem becomes harder and more observations are required to find the best, for both procedures, the

processor utilizations increase on both parallel computing platforms. This is consistent with our previous discussion.

From these two experiments, we can conclude that Procedure \mathcal{KT}^+ may have better practical performance than GSP across different problems and different parallel computing platforms because the total sample size and the wall clock time of Procedure \mathcal{KT}^+ are always smaller than those of GSP.

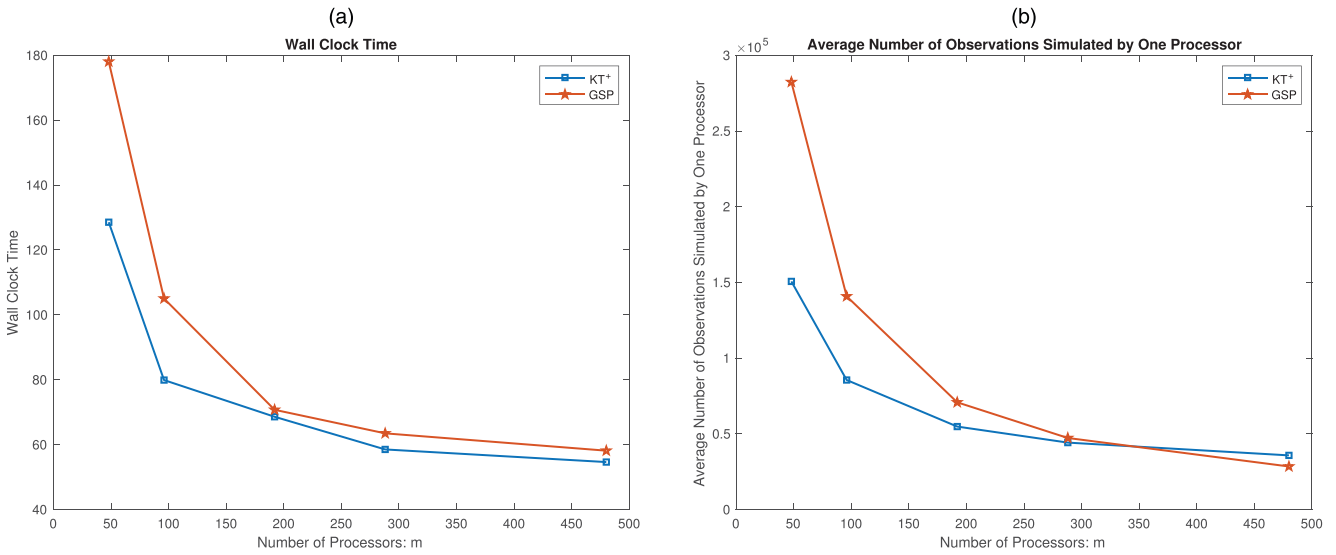
6.3.2. Experiments on Google Cloud Platform. Then, we conduct our experiments on Google Cloud Platform. On GCP, we set up a cluster with 31 n1-standard-16 virtual machines (VMs). Each VM has 16 virtual central processing units (vCPUs), 60 GB of memory, and 128 GB primary disk size. The operating system for each VM is Debian GNU/Linux 9 (stretch). Master/Worker structure is adopted for the cluster. One VM is served as the master, and all others are workers. The master performs managerial work for parallel computing. It is responsible for assigning parallel jobs to workers, and instructing and coordinating workers. The master also regularly receives reports from all workers. In case of a job failure, the master can choose a new processor to resume the job. The workers are the actual VMs that execute parallel jobs. The hourly on-demand charge for each VM is about 1.0634 United States Dollar (USD) in Hong Kong (asia-east2), resulting in a total cost of 32.9654 USD per hour for the cluster. We run GSP and Procedure \mathcal{KT}^+ on such a cluster to solve problems. Because of the high cost of the cluster, all statistics reported in the following experiments are based on one macro replication.

As in practice, one may not have the prior knowledge on the optimal number of processors needed to solve a problem, in the following experiment, we examine the wall clock times that GSP and Procedure \mathcal{KT}^+ require to solve the problem with 57,624 alternatives as different numbers of processors are used. Because processor utilizations are always low on Apache Hadoop, in this situation, as we change the number of processors, the impact on the wall clock time may not be significant. We only compare the performance of the two procedures on Apache Spark. We summarize the results in Figure 5.

Table 5. A Comparison of GSP and Procedure \mathcal{KT}^+ on the Problem with 57,624 Alternatives: $m = 48$

Procedure	Total sample size ($\times 10^6$)	Wall clock time (seconds)	Total simulation time (seconds)	Total comparison time (seconds)	Utilization (%)
Hadoop					
GSP	13.09	946.58	7134.49	5.93	15.71
Procedure \mathcal{KT}^+	6.98	201.09	3598.88	1.21	37.30
Spark					
GSP	13.54	202.23	8563.71	6.03	88.28
Procedure \mathcal{KT}^+	7.30	123.66	4587.47	1.54	77.31

Figure 5. (Color online) Scaling Performance of GSP and Procedure \mathcal{KT}^+ on Apache Spark: $k = 57,624$



Notes. (a) Wall clock time. (b) Average number of observations simulated by one processor.

From Figure 5(a), we can observe that, for both procedures, as more processors are used, the marginal effect of reducing the wall clock time diminishes. The reason is that when m is large, as we use more processors in the parallel computing environment, the improvement of the simulation time of each processor is limited (see Figure 5(b)). However, for GSP, it always needs to spend a relatively large amount of time on communications, and for Procedure \mathcal{KT}^+ , there always exists some efficiency loss because of the variations in completion times of different processors. Therefore, after the number of processors reaches a certain level, for both procedures, we hardly observe any improvement on the wall clock times. Because the wall clock times of Procedure \mathcal{KT}^+ are smaller than those of GSP in all instances, we may conclude that, in this experiment, for Procedure \mathcal{KT}^+ , the efficiency loss because of the variations in completion times of different processors is tolerable. It is also worth noting that, when $m = 480$, Procedure \mathcal{KT}^+ tends to require more observations to identify the best than GSP does. It is because the total sample size of Procedure \mathcal{KT}^+ can increase with the number of processors m , and

GSP does not. However, in this case, the wall clock time of Procedure \mathcal{KT}^+ is still smaller than that of GSP. This verifies our discussion in Section 5.2 that the computation cost for simulations is no longer a dominant factor that affects the wall clock time when the number of processors is very large compared with the size of the problem.

Lastly, we run these two procedures to solve the problem with 1,016,127 alternatives. In this experiment, all other parameters are the same as those in previous experiments except that for Procedure \mathcal{KT}^+ , we set the number of alternatives in every match g to be 50. We summarize the results in Table 6. The approximate cost for each procedure in the table is calculated by

$$\text{Approximate Cost} = \frac{\text{wall clock time (seconds)}}{3600 \text{ seconds/hour}} \times 32.9654 \text{ USD/hour.}$$

We ignore the time spent on setting up the cluster. In Table 6, we observe very similar results to those of previous experiments. We may also conclude that Procedure \mathcal{KT}^+ has more potential to economically

Table 6. A Comparison of GSP and Procedure \mathcal{KT}^+ on the Problem with 1,016,127 Alternatives: $m = 480$

Procedure	Total sample size ($\times 10^8$)	Wall clock time (seconds)	Total simulation time (seconds)	Total comparison time (seconds)	Utilization (%)	Approximate cost (USD)
Hadoop						
GSP	7.19	8305.59	360931.35	60.58	9.05	76.05
Procedure \mathcal{KT}^+	3.22	760.38	164796.15	50.06	45.17	6.96
Spark						
GSP	6.96	1254.37	353259.66	59.31	58.68	11.49
Procedure \mathcal{KT}^+	3.23	623.69	164921.99	52.81	55.11	5.71

Table 7. Robustness Test for Procedure \mathcal{KT}^+ Using a Random Number of Warm-Up Periods on Apache Spark: $\mathcal{W} \sim \text{unif}(2000 - \omega/2, 2000 + \omega/2)$

Number of alternatives: k	ω	Wall clock time (seconds)	Utilization (%)
3,249	0	6.03	56.61
3,249	1,000	6.08	53.36
3,249	2,000	6.25	52.49
57,624	0	123.66	77.31
57,624	1,000	126.82	75.97
57,624	2,000	128.69	74.97

solve very large-scale R&S problems on commercial clouds across different parallel computing platforms.

6.3.3. Robustness to Random Simulation Times. In practice, the time needed to generate an observation may vary from one to another. The variability in simulation times may intensify the variations in completion times of different processors and thus, affect the performance of Procedure \mathcal{KT}^+ in parallel computing environments. In this experiment, we conduct a robustness test for Procedure \mathcal{KT}^+ . Specifically, while simulating an additional observation for every surviving alternative in a match, instead of warming up the system with exact 2,000 jobs, we warm up the system with \mathcal{W} jobs, where \mathcal{W} is drawn from a uniform distribution with mean 2,000 and width ω . In this example, we let $\omega = 1,000$ and $\omega = 2,000$ (i.e., $\mathcal{W} \sim \text{unif}(1500, 2500)$ and $\mathcal{W} \sim \text{unif}(1000, 3000)$) to represent different levels of variability. We use Procedure \mathcal{KT}^+ to solve the problems with 3,249 and 57,624 alternatives on the local server with 48 processors. We only conduct the experiment on Apache Spark and estimate the wall clock times and utilizations based on 100 macro replications. The results are summarized in Table 7. As a reference, in the table, we also include the results of the case where the number of warm-up periods is exact 2,000 (i.e., $\omega = 0$).

From Table 7, we observe that there is indeed an increase on the wall clock time and a decrease on the utilization as the variability in simulation times increases. However, compared with the base case where $\omega = 0$, the differences are not large, and we think they are acceptable. It suggests that Procedure \mathcal{KT}^+ is relatively robust against the randomness in simulation times.

7. Concluding Remarks

Different from single-processor computing environments, developing procedures for parallel computing environments needs one's careful consideration of the time spent on simulation, comparison, and communication. In this paper, inspired by the knockout-tournament arrangement of tennis Grand Slam tournaments, we develop procedures that are well suited for solving large-scale R&S problems in parallel computing

environments. The total sample sizes of our procedures grow linearly with the number of alternatives k no matter whether the variances of the alternatives are known or not. Therefore, our procedures are optimal in rate. The total comparison time in our procedures is negligible compared with the simulation time. When the procedures are implemented in parallel computing environments, the number of communications among the processors is minimal. Furthermore, CRNs can be easily applied to our procedures. The numerical results show that our procedures are competitive with the best parallel R&S procedures.

There are two topics that are worth future research. First, in this paper, we focus on using Procedure \mathcal{KN} to conduct the matches. In fact, depending on the specific problem encountered in practice, one also has the freedom to use other existing procedures. For example, the recent Bayes-inspired indifference zone (BIZ) procedure proposed by Frazier (2014) demonstrates the ability to use much fewer observations to identify the best than Procedure \mathcal{KN} does when the observations are independently generated from different alternatives. Thus, one may consider using the BIZ procedure to conduct the matches for the problems where CRNs cannot be used. It would be valuable to investigate the best choices of the procedures that can be used to conduct matches for different types of problems. Second, in this paper, we focus on developing procedures under the IZ formulation. To alter our procedures to satisfy the more strict PAC guarantee, the procedures experience a large amount of efficiency loss on the total sample size because smaller IZ parameters are assigned to the matches. Recently, there are some works studying IZ-free procedures. For these procedures, their performance is immune to the change of the IZ parameter. It is of both theoretical and practical interest to study how one can modify these procedures and use them to help our procedures satisfy the PAC guarantee without losing much efficiency on the total sample size.

Acknowledgments

The authors thank the editor-in-chief John Birge, the associate editor, and three referees for helpful comments that improved the presentation and structuring of the paper.

Endnotes

¹ Let $\mathcal{B}_\Delta(\cdot)$ denote the BM process with unit variance and drift Δ . As shown by Hong (2006), the random processes $\{Z_{ij}(t) = t[\bar{X}_i(t) - \bar{X}_j(t)] / \sigma_{ij}^2 : t = 1, 2, \dots\}$ and $\{\mathcal{B}_{\mu_i - \mu_j}(t/\sigma_{ij}^2) : t = 1, 2, \dots\}$ have the same joint distribution.

² If the observations simulated by different alternatives are independent, the desired false elimination probability can be further improved to $1 - (1 - \alpha)^{1/k-1}$. However, the rate optimality is still not achievable.

³ Because two alternatives engage in one match and there are $k - 1$ matches, the total number of matches that all alternatives engage in is $2(k - 1)$. Therefore, on average, each alternative engages in $2(k - 1)/k$ matches.

⁴ When every processor produces a local best alternative, the standard knockout-tournament procedure takes additional τ observations for each alternative and directly selects the alternative with the largest sample mean as the final best alternative.

References

- Bechhofer RE (1954) A single-sample multiple decision procedure for ranking means of normal populations with known variances. *Ann. Math. Statist.* 25(1):16–39.
- Bechhofer RE, Goldsman DM, Santner TJ (1995) *Design and Analysis of Experiment for Statistical Selection, Screening, and Multiple Comparisons*, 1st ed. (Wiley, New York).
- Buzacott JA, Shanthikumar JG (1993) *Stochastic Models of Manufacturing Systems*, vol. 4, 1st ed. (Pearson, Upper Saddle River, NJ).
- Chen CH, Lin J, Yücesan E, Chick SE (2000) Simulation budget allocation for further enhancing the efficiency of ordinal optimization. *Discrete Event Dynam. Systems* 10(3):251–270.
- Chick SE (2006) Subjective probability and Bayesian methodology. Henderson SG, Nelson BL, eds. *Handbook in Operations Research and Management Science: Simulation*, vol. 13 (Elsevier, Amsterdam), 225–257.
- Chick SE, Frazier P (2012) Sequential sampling with economics of selection procedures. *Management Sci.* 58(3):550–569.
- Chick SE, Gans N (2009) Economic analysis of simulation selection problems. *Management Sci.* 55(3):421–437.
- Chick SE, Inoue K (2001) New two-stage and sequential procedures for selecting the best simulated system. *Oper. Res.* 49(5):732–743.
- Cormen TH, Leiserson CE, Rivest RL, Stein C (2001) *Introduction to Algorithms*, 2nd ed. (The MIT Press, Cambridge, MA).
- Dudewicz EJ, Dalal SR (1975) Allocation of observations in ranking and selection with unequal variances. *Sankhya Indian J. Statist. Ser. B* 37(1):28–78.
- Even-Dar E, Mannor S, Mansour Y (2002) PAC bounds for multi-armed bandit and Markov decision processes. Kivinen J, Sloan R, eds. *Proc. 15th Internat. Conf. Computational Learning Theory* (Springer, Berlin), 255–270.
- Fan W, Hong LJ, Nelson BL (2016) Indifference-zone-free selection of the best. *Oper. Res.* 64(5):1499–1514.
- Frazier PI (2014) A fully sequential elimination procedure for indifference-zone ranking and selection with tight bounds on probability of correct selection. *Oper. Res.* 62(4):926–942.
- Frazier PI, Powell WB, Dayanik S (2008) A knowledge-gradient policy for sequential information collection. *SIAM J. Control Optim.* 47(5):2410–2439.
- Frazier PI, Powell WB, Dayanik S (2009) The knowledge-gradient policy for correlated normal beliefs. *INFORMS J. Comput.* 21(4):599–613.
- Hong LJ (2006) Fully sequential indifference-zone selection procedures with variance-dependent sampling. *Naval Res. Logist.* 53(5):464–476.
- Hunter SR, Nelson BL (2017) Parallel ranking and selection. Tolk A, Fowler J, Shao G, Yücesan E, eds. *Advances in Modeling and Simulation: Seminal Research from 50 Years of Winter Simulation Conferences* (Springer, Cham, Switzerland), 249–275.
- Inoue K, Chick SE (1998) Comparison of Bayesian and frequentist assessments of uncertainty for selecting the best system. Medeiros DJ, Watson EF, Carson JS, Manivannan MS, eds. *Proc. 1998 Winter Simulation Conf.* (IEEE, Piscataway, NJ), 727–734.
- Jamieson K, Malloy M, Nowak R, Bubeck S (2014) Lil' UCB: An optimal exploration algorithm for multi-armed bandits. Balcan MF, Feldman V, Szepesvári C, eds. *Proc. 27th Conf. Learn. Theory* (PMLR, Barcelona, Spain), 423–439.
- Kao SC, Lai TL (1980) Sequential selection procedures based on confidence sequences for normal populations. *Comm. Statist. Theory Methods* 9(16):1657–1676.
- Kim SH, Nelson BL (2001) A fully sequential procedure for indifference-zone selection in simulation. *ACM Trans. Model. Comput. Simulation* 11(3):251–273.
- Kim S-H, Nelson BL (2006) Selecting the best system. Henderson SG, Nelson BL, eds. *Handbook in Operations Research and Management Science: Simulation*, vol. 13 (Elsevier, Amsterdam), 501–534.
- Kim MP, Suksompong W, Williams VV (2017) Who can win a single-elimination tournament? *SIAM J. Discrete Math.* 31(3):1751–1764.
- Luo J, Hong LJ (2011) Large-scale ranking and selection using cloud computing. Jain S, Creasey R, Himmelspach J, White KP, Fu M, eds. *Proc. 2011 Winter Simulation Conf.* (IEEE, Piscataway, NJ), 4046–4056.
- Luo J, Hong LJ, Nelson BL, Wu Y (2015) Fully sequential procedures for large-scale ranking-and-selection problems in parallel computing environments. *Oper. Res.* 63(5):1177–1194.
- Nelson BL, Matejcek FJ (1995) Using common random numbers for indifference-zone selection and multiple comparisons in simulation. *Management Sci.* 41(12):1935–1945.
- Ni EC, Henderson SG, Hunter SR (2014) A comparison of two parallel ranking and selection procedures. Tolk A, Diallo S, Ryzhov IO, Yilmaz L, eds. *Proc. 2014 Winter Simulation Conf.* (IEEE, Piscataway, NJ), 3761–3772.
- Ni EC, Ciocan DF, Henderson SG, Hunter SR (2017) Efficient ranking and selection in parallel computing environments. *Oper. Res.* 65(3):821–836.
- Paulson E (1964) A sequential procedure for selecting the population with the largest mean from k normal populations. *Ann. Math. Statist.* 35(1):174–180.
- Peng Y, Chong EK, Chen CH, Fu MC (2018) Ranking and selection as stochastic control. *IEEE Trans. Automatic Control* 63(8):2359–2373.
- Rinott Y (1978) On two-stage selection procedures and related probability-inequalities. *Comm. Statist. Theory Methods* 7(8):799–811.
- Zhong Y, Hong LJ (2017) A new framework of designing sequential ranking-and-selection procedures. Chan WKV, D'Ambrogio A, Zacharewicz G, Mustafee N, Wainer GA, Page E, eds. *Proc. 2017 Winter Simulation Conf.* (IEEE, Piscataway, NJ), 2237–2244.

Ying Zhong is an assistant professor in the Department of Management Science and E-commerce at the University of Electronic Science and Technology of China. His research interests include simulation optimization, the multiarmed bandit problem with covariates, and parallel computing.

L. Jeff Hong is the Fudan Distinguished Professor and the Hongyi Chair Professor appointed by the School of Management and the School of Data Science at Fudan University. His research interests are in the broad areas of machine learning and business analytics: stochastic modeling, stochastic simulation, stochastic optimization, statistical learning, and reinforcement learning, with applications in supply chain management, revenue management, financial risk management, and healthcare analytics.