

Finding Probably Better System Configurations Quickly

Takayuki Osogami
IBM Tokyo Research Laboratory
1623-14 Shimotsuruma, Yamato-shi
Kanagawa 242-8501, Japan
osogami@jp.ibm.com

Toshinari Itoko
IBM Tokyo Research Laboratory
1623-14 Shimotsuruma, Yamato-shi
Kanagawa 242-8501, Japan
itoko@jp.ibm.com

ABSTRACT

The performance of computer and communication systems can in theory be optimized by iteratively finding better system configurations. However, a bottleneck is the time required in simulations/experiments for finding a better system configuration in each iteration. We propose algorithms that quickly find a system configuration that is probably better than the “standard” system configuration, where the performance of a given system configuration is estimated via simulations or experiments. We prove that our algorithms make correct decisions with high probability, and various heuristics to reduce the total simulation time are proposed. Numerical experiments show the effectiveness of the proposed algorithms, and this leads to several guidelines for designing efficient and reliable optimization procedures for the performance of computer and communication systems.

Categories and Subject Descriptors

I.6.6 [Simulation and Modeling]: Simulation output analysis

General Terms

Algorithms, Performance, Theory.

Keywords

Simulation, Performance optimization, Local search, Ranking and Selection, Screening.

1. INTRODUCTION

Motivation

Achieving high performance (such as short response time, high throughput, and fairness among users) is often a central goal in designing computer and communication systems, including Web servers, supercomputers, and computer networks. However, although it is easy to come up with many

possible configurations of a system, it is a difficult and time consuming task to choose the best possible configuration among many such possibilities so that the system performance is optimized. For example, designing a high performance Web server system involves finding the best possible combination of hardware components, scheduling or dispatching policies, and configuration parameter values such as cache sizes and timeout values for each of the HTTP and application servers and the databases. Since there are a large number of possible configurations and since estimating the performance of each configuration via (computer) simulations or (physical) experiments often requires tens of minutes, it would be impractical to accurately evaluate the performance of all of the possible configurations to identify the best configuration. Our goal is to develop a framework that allows us to quickly identify the best configuration of a system when there are many possible configurations and the performance of each configuration can be estimated via simulations or experiments.

A popular approach for finding the optimal solution (or configuration, in particular) from a huge set of possible solutions is local search, which iteratively finds a better solution from “neighbors” of a current solution and replaces the current solution with the neighboring better solution [1]. A local search algorithm may be augmented with existing mechanisms to escape from a locally optimal but globally suboptimal solution. Although local search is originally developed for *non-stochastic* optimization problems, it has recently started to receive attention as a promising approach for optimizing *stochastic* systems whose performance can be estimated via simulations or experiments. For example, Ye and Kalyanaraman apply a recursive random search algorithm to the performance optimization of network protocols, where the performance is estimated via simulations [14], and Xi *et al.* apply a hill-climbing algorithm to optimizing the configuration parameters of Web application servers, where the performance is measured via experiments [13].

Although local search allows us to limit the search space to a smaller number of system configurations, simulations or experiments often require many iterations (e.g., many samples of simulated performance) simply to estimate the performance of a single configuration. For example, a test run in the experiment of Xi *et al.* [13] requires fifteen minutes to estimate the mean response time for each configuration of a Web application server. As a result, only four out of many configuration parameters are optimized via local search. Therefore, to optimize the performance of stochastic systems via local search, it is of paramount importance

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGMetrics/Performance'06, June 26–30, 2006, Saint Malo, France.
Copyright 2006 ACM 1-59593-320-4/06/0006 ...\$5.00.

to minimize the time needed for simulations to find a better configuration from neighborhood configurations.

In this paper, we propose algorithms for finding a neighborhood configuration that is better than the current configuration when the performance of a configuration is estimated via simulations (or experiments), so that the total simulation time is minimized. Our algorithms can substantially speed up local search for optimizing the performance of stochastic systems. A key idea in our algorithm is to evaluate *only the necessary neighborhood configurations* and to evaluate them *only with the necessary accuracy*. Note that we do not need to estimate the performance of each configuration accurately, simply to determine one configuration is better than another. Below, a system configuration is also referred to as a *solution*. Further, the current solution is also referred to as the *standard* solution or simply as the standard, and a neighborhood solution is also referred to as a *candidate* solution or simply as a candidate.

Related work

To the best of our knowledge there is no prior work with our goal of quickly finding a candidate solution that is better than the standard solution. The work closest to ours is known as *ranking and selection* [2, 12], where the goal is to find the best solution, to find a subset containing the best solution, or to find a fixed number of best solutions among all solutions in consideration. The algorithms for finding the best solution can be used in local search to iteratively find the best neighborhood solution and replace the current solution by the best neighborhood solution (see e.g. [10]). However, we will see that algorithms for finding the best solution can be much less efficient than our algorithm for finding a better candidate, and further the properties of the most existing algorithms for finding the best solution are *not* fully appropriate for the purpose of local search.

Ranking and selection approaches find the best solution among all solutions in consideration, where it is assumed that simulating solution π_i yields a sample of the normal random variable with mean μ_i and standard deviation σ_i (both μ_i and σ_i are unknown). Although this assumption may sound unrealistic, it turns out that samples in most simulations can be made arbitrarily close to independent normal random variables by appropriate batching [7]. For example, consider the mean response time in a queueing system (such as an M/M/1 queue). Although the simulated response time of the i -th job, X_i , does *not* have a normal distribution, the average of a batch of n response times, $\bar{X}_m = \frac{1}{n} \sum_{i=1}^n X_{mn+i}$, can be made arbitrarily close to a normal random variable by letting n become large. Further, \bar{X}_m and $\bar{X}_{m'}$ become independent as $n \rightarrow \infty$ for $m \neq m'$. Thus, the mean response time is estimated by $\frac{1}{N/m} \sum_{i=1}^{N/m} \bar{X}_m$, where samples, $\{\bar{X}_m\}$, can be made arbitrarily close to a sequence of independent normal random variables. Similarly, samples of any function, $f(X)$, of the response time X , including X^2 and an indicator random variable $I_{\{X>c\}}$, can be made arbitrarily close to a sequence of independent normal random variables.

The goal of ranking and selection algorithms for finding the best solution is to find the solution π_b that has the largest mean performance, μ_b , with high probability ($\geq 1 - \alpha$ for a given α) so that the total number of samples is minimized. We assume that a larger mean performance is better, in the rest of this paper. It is standard to allow an error within

an *indifference zone* δ , so that any solution whose mean is $\geq \mu_b - \delta$ is considered to be one of the “best” solutions. Dudevicz and Dalal [3] and Rinott [11] propose two-stage algorithms for finding the best solution. (The work preceding to [3, 11] assumes known or common σ_i , which is unrealistic for the performance of computer and communication systems.) Stage 1 of the algorithms in [3, 11] collects a fixed number of samples from each solution and estimates its variance. Based on the estimated variance, the number of samples needed in Stage 2 is determined for each solution. At the end of Stage 2, the solution with the largest estimated mean is selected as the best solution. Nelson *et al.* [9] propose a screening procedure that can be used with the two-stage algorithms in [3, 11]. The screening procedure screens out clearly poor solutions after Stage 1, and this can significantly reduce the number of samples needed in Stage 2. The two-stage algorithm with screening can be extended to more than two stages or to a fully sequential algorithm, where a screening procedure is applied at each stage until only the best solution is left (see e.g. [5]).

Algorithms for finding the best solution may be used in local search, where the current solution is replaced with the best solution among all of the current solution and its neighborhood solutions. However, due to the indifference zone, local search with the abovementioned algorithms for finding the best solution may continually replace the current solution with a *worse* neighborhood solution in the situation where the best neighborhood solution is always worse than the current solution but the difference is within δ . For the purpose of local search, the algorithm proposed by Nelson and Goldsman, which we refer to as the Nelson-Goldsman algorithm or simply **NG**, is more appropriate [8]. **NG** finds the best solution among all solutions in consideration, but one of the solutions is designated as the standard. **NG** selects the standard with high probability if the standard is the best, and it selects a solution that is better than the standard and whose mean is $\geq \mu_b - \delta$ with high probability if the best solution is not the standard.

Summary of contributions

In this paper, we propose two-stage algorithms for selecting a candidate solution that is better than the standard or for selecting the standard if no candidate is better than the standard, where we assume that simulating each solution yields a sample from a normal distribution (as is standard in the literature on ranking and selection). More formally, let π_0 be the standard solution and $\mathcal{C} \equiv \{\pi_1, \dots, \pi_k\}$ be the set of candidate solutions, and let μ_i be the mean of π_i for $i = 0, 1, \dots, k$. Then, our algorithms have the following properties:

- If π_0 is the best ($\mu_0 \geq \mu_i, \forall \pi_i \in \mathcal{C}$), we select π_0 with high probability ($\geq 1 - \alpha$).
- If there is a *significantly better* candidate ($\exists \pi_i \in \mathcal{C}$ such that $\mu_i \geq \mu_0 + \delta$), we select a *better* candidate ($\pi_i \in \mathcal{C}$ such that $\mu_i > \mu_0$)¹ with high probability.
- If there is a better candidate but no significantly better ones, we select π_0 or a better candidate with high probability.

These properties are carefully designed so that an algorithm meeting these properties will have expected outcomes when

¹A better candidate may not be *significantly* better.

it is used in local search for optimizing the performance of stochastic systems. The definitions of these properties and the new algorithms that satisfy these properties constitute the primary contributions of this paper. (See Section 2.)

The secondary contributions of this paper are various heuristics to minimize the total number of samples needed to find a better candidate (or to conclude that the standard is the best), which in turn minimizes the total simulation time. Since our algorithm terminates as soon as a better candidate is found (unlike the existing algorithms for selecting the *best* solution), we can reduce the total number of samples by first taking samples from the candidates that are more likely to be better than the standard and require a smaller number of samples to take. We will prove that there is an optimal order on the candidate solutions so that the expected total number of samples is minimized. (See Section 2.3.)

We also find that randomizing the sampling order helps in reducing the number of necessary samples. The number of samples that need to be taken in a particular algorithm is chosen such that the above properties are met for the *worst* case instance, for which the algorithm is most likely to make a mistake. It turns out that an algorithm with a randomized order is not very likely to make mistakes for *any* instance with a smaller number of samples. (See Section 3.)

We also propose two procedures, *first stage selection* (FSS) and *screening to the best* (STB), which are motivated by screening procedures for algorithms to find the *best* solution [9]. FSS allows us to terminate in the middle of the first stage when a better candidate is found, and this can reduce the total number of necessary samples by orders of magnitude. STB is similar to the existing screening procedures, but it is designed to work when there is a standard solution. STB can also be used to reduce the number of samples needed in NG [8]. (See Section 4.)

Finally, we show the effectiveness of the proposed algorithms via numerical experiments. In particular, we show that our algorithms, which find a *better* candidate, can be much faster than NG, which finds the *best* solution. This is to be expected, since NG guarantees stronger properties than ours. Also, the effectiveness of the optimal sampling order, randomization, FSS, and STB are presented and discussed. We then provide guidelines on designing good local search algorithms for stochastic systems. (See Section 5).

2. BASIC ALGORITHM AND ITS PROPERTIES

We start by a simpler algorithm, which we refer to as the basic algorithm or simply as **Basic**. **Basic** selects a candidate $\pi \in \mathcal{C} \equiv \{\pi_1, \dots, \pi_k\}$ that is better than the standard π_0 or selects π_0 if there is no better candidate in \mathcal{C} . Let $\mathcal{A} \equiv \{\pi_0\} \cup \mathcal{C}$ denote all solutions in consideration. Let μ_i and σ_i be the mean and standard deviation, respectively, of $\pi_i \in \mathcal{A}$. Also, let $\mathcal{B} \equiv \{\pi_i \in \mathcal{C} \mid \mu_i > \mu_0\}$ denote the set of better candidates, and let π_b denote any one of the best candidates, so that $\mu_b \geq \mu_i, \forall \pi_i \in \mathcal{C}$. Further, let $\mathcal{W} \equiv \{\pi_i \in \mathcal{C} \mid \mu_i \leq \mu_0\}$ denote the set of worse candidates. Formally, the following theorem characterizes the properties of **Basic**.

Stage 1:
For each $\pi_i \in \mathcal{A}$, collect n_0 samples and calculate the sample variance S_i^2 .

Stage 2:
For each $\pi_i \in \mathcal{A}$, calculate the number of samples to be collected, $N_i = \max\{n_0, \lceil (g^* S_i / \delta)^2 \rceil\}$, where g^* is a constant defined in Section 2.1.1. Take $N_0 - n_0$ samples from π_0 , and calculate the sample mean, $\overline{X}_0^{(N_0)}$.

For each $\pi_i \in \mathcal{C}$ (in an arbitrary order),
take $N_i - n_0$ samples from π_i ;
calculate the sample mean, $\overline{X}_i^{(N_i)}$;
let $c = \gamma^* \delta$, where γ^* is a constant defined in Section 2.1.1;
if $\overline{X}_i^{(N_i)} \geq \overline{X}_0^{(N_0)} + c$, select $\pi = \pi_i$ and exit.
If $\overline{X}_i^{(N_i)} < \overline{X}_0^{(N_0)} + c$ for all $\pi_i \in \mathcal{C}$, select $\pi = \pi_0$.

Figure 1: The basic algorithm, Basic

THEOREM 1. *Let π be the solution selected by **Basic**. Then,*

$$\Pr(\pi = \pi_0 \mid \mu_0 \geq \mu_b) \geq 1 - \alpha \quad (1)$$

$$\Pr(\pi \in \mathcal{B} \mid \mu_b \geq \mu_0 + \delta) \geq 1 - \alpha \quad (2)$$

$$\Pr(\pi \in \mathcal{B} \cup \{\pi_0\} \mid \mu_0 + \delta > \mu_b > \mu_0) \geq 1 - \alpha. \quad (3)$$

We describe **Basic** in Section 2.1, and prove Theorem 1 in Section 2.2.

2.1 Basic algorithm

Basic consists of two stages. The goal of Stage 1 is to estimate the variance of each solution, which is used to determine the number of samples needed in Stage 2. Stage 2 then selects a candidate that is better than the standard, or conclude that there are no better candidates. Figure 1 summarizes **Basic**, on which we elaborate below.

Stage 1 collects a fixed number, n_0 , of samples and calculate the sample variance, S_i^2 , for each $\pi_i \in \mathcal{A}$. Here, S_i^2 is calculated via

$$S_i^2 = \frac{1}{n_0 - 1} \sum_{\ell=1}^{n_0} \left(X_{i,\ell} - \overline{X}_i^{(n_0)} \right)^2,$$

where $X_{i,\ell}$ is the ℓ -th sample from π_i , and

$$\overline{X}_i^{(n_0)} = \frac{1}{n_0} \sum_{\ell=1}^{n_0} X_{i,\ell}$$

is the sample mean for $\pi_i \in \mathcal{A}$. In theory, n_0 can be any integer > 1 . In practice, n_0 should be chosen large enough such that the sample variance becomes a reasonable estimate of the true variance, since too small an n_0 results in a greater number of samples in Stage 2. For example, we set $n_0 = 20$ for our experiments in Section 5.

Stage 2 first determines the number of samples, N_i , that should be collected from π_i via

$$N_i = \max\left\{n_0, \left\lceil (g^* S_i / \delta)^2 \right\rceil\right\} \quad (4)$$

for each $\pi_i \in \mathcal{A}$. We postpone an explanation on the confidence parameter g^* to Section 2.1.1.

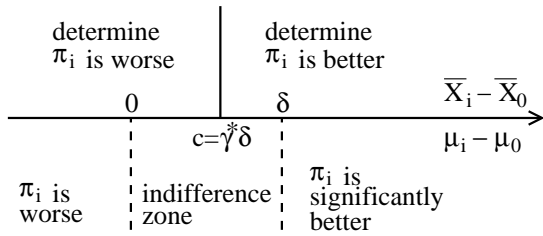


Figure 2: The choice of $c = \gamma^* \delta$ in Basic.

Once N_i 's are determined, the additional samples are collected from the standard, π_0 , and then from the candidates in \mathcal{C} . Here, we may take samples from \mathcal{C} in any order, but an optimal order on \mathcal{C} is provided in Section 2.3 such that the expected total number of samples is minimized. After taking the N_i samples, the sample mean is calculated via

$$\overline{X}_i^{(N_i)} = \frac{1}{N_i} \sum_{\ell=1}^{N_i} X_{i,\ell}.$$

Basic next determines whether $\overline{X}_i^{(N_i)}$ is significantly larger than $\overline{X}_0^{(N_0)}$ to conclude that π_i is a better candidate. The criterion is

$$\overline{X}_i^{(N_i)} \geq \overline{X}_0^{(N_0)} + c. \quad (5)$$

Namely, if (5) holds for $\pi_i \in \mathcal{C}$, **Basic** determines that π_i is a better candidate, and terminates. If (5) does not hold for any $\pi_i \in \mathcal{C}$, **Basic** determines that the standard π_0 is no worse than any candidate in \mathcal{C} .

In Criterion (5), c is a constant such that $0 < c < \delta$. Figure 2 provides an intuition behind the choice of c . If $\mu_i = \mu_0$, we want to determine that π_i is *no better* than π_0 . However, if the criterion was $\overline{X}_i^{(N_i)} \geq \overline{X}_0^{(N_0)}$, we would need an infinite number of samples to make a correct decision (with probability $> \frac{1}{2}$), since the sample mean would need to match the true mean exactly. Hence, we need $c > 0$, and, in fact, *larger* c results in a smaller number of samples needed to make a correct decision when $\mu_i = \mu_0$. On the other hand, if $\mu_i = \mu_0 + \delta$, we want to determine that π_i is *better* than π_0 . If the criterion was $\overline{X}_i^{(N_i)} \geq \overline{X}_0^{(N_0)} + \delta$, however, we would need an infinite number of samples to make a correct decision (with probability $> \frac{1}{2}$). Hence, we need $c < \delta$, and, in fact, *smaller* c results in a smaller number of samples needed to make a correct decision when $\mu_i = \mu_0 + \delta$. The above argument suggests that c should be chosen such that it is sufficiently larger than 0 and sufficiently smaller than δ to minimize the necessary number of samples. We will next define the criterion ratio γ^* , which determines c via $c = \gamma^* \delta$.

2.1.1 Confidence parameter and criterion ratio

As discussed above, a criterion ratio, γ , which determines c via $c = \gamma \delta$, should be chosen such that N_i can be minimized. Since a confidence parameter, g , determines N_i via $N_i = \max\{n_0, \lceil (g S_i^2 / \delta)^2 \rceil\}$, our goal is reduced to find a γ that minimizes g . A difficulty is that g must be chosen large enough such that (1)-(3) in Theorem 1 are satisfied. The confidence parameter, g^* , and the criterion ratio, γ^* , used in **Basic** are our best choices, which we define below.

Formally, the confidence parameter g^* is an extension of the Rinott constant, introduced in Rinott's algorithm

for selecting the best solution [11]. The Rinott constant, $h_{k+1,1-\alpha,n_0-1}$, is the unique constant h such that $f_{k,n_0-1}(h) = 1 - \alpha$, where

$$f_{t,\nu}(h) \equiv \Pr \left(Z_i \leq \frac{h}{\sqrt{\nu(1/Q_i + 1/Q_0)}}, i = 1, \dots, t \right) \quad (6)$$

is Rinott's distribution function, where Z_i is a standard normal random variable, Q_i is a χ^2 random variable with degree of freedom ν for $i = 0, 1, \dots, t$, and Z_i 's and Q_i 's are independent. The Rinott constant can be calculated via an algorithm in [2], which finds h such that $f_{k,n_0-1}(h) = 1 - \alpha$ by binary search, where $f_{k,n_0-1}(h)$ is evaluated via numerical (Gauss-Laguerre) integration.

We define the confidence parameter, g^* , as

$$g^* \equiv \min_{0 < \gamma < 1} \max\{g_1(\gamma), g_2(\gamma)\}, \quad (7)$$

where $g_1(\gamma)$ and $g_2(\gamma)$ are the unique g_1 and g_2 , respectively, that satisfy

$$f_{k,n_0-1}(\gamma g_1) = 1 - \alpha \quad (8)$$

$$f_{k-1,n_0-1}(\gamma g_2) + f_{1,n_0-1}(\gamma g_2) - 1 = 1 - \alpha \quad (9)$$

for $0 < \gamma < 1$. We denote the minimizer of (7) by γ^* . Roughly speaking, (1) and (3) are satisfied if g is chosen such that $g > g_1(\gamma)$, and (2) is satisfied if $g > g_2(\gamma)$. Note in particular that

$$f_{k,n_0-1}(\gamma^* g^*) \geq 1 - \alpha \quad (10)$$

$$f_{k-1,n_0-1}(\gamma^* g^*) + f_{1,n_0-1}((1 - \gamma^*) g^*) - 1 \geq 1 - \alpha \quad (11)$$

since $f_{t,\nu}(x)$ is an increasing function of x for any $t \geq 1$ and $\nu \geq 1$. Note that the confidence parameter g^* and the criterion ratio γ^* are constants that depend only on k , α , and n_0 .

A standard search algorithm can be used to find g^* and γ^* , where $g_1(\gamma)$ and $g_2(\gamma)$ are calculated for each given γ by a binary search algorithm in a similar way to the Rinott constant². In practice, it takes a few seconds to compute a pair (g^*, γ^*) for each triple (k, α, n_0) . When this computation becomes a bottleneck, we may compute the constants beforehand and store them in a table.

2.2 Proof of Theorem 1

First consider (1). Note that **Basic** selects π_0 iff (5) is not satisfied for any $\pi_i \in \mathcal{C}$. Thus,

$$\begin{aligned} & \Pr(\pi = \pi_0 \mid \mu_0 \geq \mu_b) \\ &= \Pr \left(\overline{X}_i^{(N_i)} < \overline{X}_0^{(N_0)} + c, \forall \pi_i \in \mathcal{C} \mid \mu_0 \geq \mu_b \right). \end{aligned} \quad (12)$$

Hence, the following lemma together with (10) and (12) implies (1).

LEMMA 1.

$$\begin{aligned} & \Pr \left(\overline{X}_i^{(N_i)} < \overline{X}_0^{(N_0)} + c, \forall i = 1, \dots, k \mid \mu_0 \geq \mu_i, \forall i = 1, \dots, k \right) \\ & \geq f_{k,n_0-1}(\gamma^* g^*), \end{aligned}$$

where $f_{k,n_0-1}(\cdot)$ is defined by (6), and g^* , γ^* , and c are the constants used in **Basic**.

²It can be shown that g^* can be obtained simply as $g^* \equiv \min_{0 < \gamma < 1} g_2(\gamma)$, and γ^* is the minimizer. We omit the proof in this paper.

The proof of Lemma 1 is postponed in the appendix.

Inequality (3) can be proved in a similar way to (1), since

$$\begin{aligned} & \Pr(\pi \in \mathcal{B} \cup \{\pi_0\} \mid \mu_0 + \delta > \mu_b > \mu_0) \\ & \geq \Pr\left(\overline{X}_i^{(N_i)} < \overline{X}_0^{(N_0)} + c, \forall \pi_i \in \mathcal{W}\right) \\ & \geq f_{|\mathcal{W}|, n_0-1}(\gamma^* g^*) \quad \text{by Lemma 1} \\ & \geq 1 - \alpha \quad \text{by (10) and } |\mathcal{W}| \leq k. \end{aligned}$$

Below, we prove (2). We first show that

$$\begin{aligned} & \Pr(\pi \in \mathcal{B} \mid \mu_b \geq \mu_0 + \delta) \\ & \geq \Pr\left(\overline{X}_j^{(N_j)} < \overline{X}_0^{(N_0)} + c, \forall \pi_j \in \mathcal{W}\right. \\ & \quad \left. \text{and } \overline{X}_b^{(N_b)} \geq \overline{X}_0^{(N_0)} + c \mid \mu_b \geq \mu_0 + \delta\right) \equiv P. \quad (13) \end{aligned}$$

To show (13), consider a modified algorithm, which is the same as **Basic** except that the modified algorithm discards a selected candidate, π , if π is better than π_0 but not the best and continues until $\pi \in \mathcal{W} \cup \{\pi_b, \pi_0\}$ is selected. Note that **Basic** makes a correct selection if the modified algorithm makes a correct selection. Observe that the right hand side, P , of (13) is the probability that the modified algorithm makes a correct selection.

Then, P can be bounded from below as follows.

$$\begin{aligned} P & \geq \Pr\left(\overline{X}_j^{(N_j)} < \overline{X}_0^{(N_0)} + c, \forall \pi_j \in \mathcal{W}\right) \\ & \quad - \Pr\left(\overline{X}_b^{(N_b)} < \overline{X}_0^{(N_0)} + c \mid \mu_b \geq \mu_0 + \delta\right). \quad (14) \end{aligned}$$

Consider the first term of the right hand side of (14). By Lemma 1,

$$\begin{aligned} & \Pr\left(\overline{X}_j^{(N_j)} < \overline{X}_0^{(N_0)} + c, \forall \pi_j \in \mathcal{W}\right) \\ & \geq f_{|\mathcal{W}|, n_0-1}(\gamma^* g^*) \geq f_{k-1, n_0-1}(\gamma^* g^*), \quad (15) \end{aligned}$$

where the last inequality follows from $|\mathcal{W}| \leq k - 1$. The following lemma can be used to bound the second term of the right hand side of (14).

LEMMA 2.

$$\begin{aligned} & \Pr\left(\overline{X}_b^{(N_b)} < \overline{X}_0^{(N_0)} + c \mid \mu_b \geq \mu_0 + \delta\right) \\ & \leq 1 - f_{1, n_0-1}((1 - \gamma^*) g^*), \end{aligned}$$

where $f_{k, n_0-1}(\cdot)$ is defined by (6), and g^* , γ^* , and c are the constants used in **Basic**.

A proof of Lemma 2 is postponed to the appendix. Now, (14), (15), and Lemma 2 imply

$$P \geq f_{k-1, n_0-1}(\gamma^* g^*) + f_{1, n_0-1}((1 - \gamma^*) g^*) - 1.$$

Hence, $P \geq 1 - \alpha$ follows from (11). By (13), this completes the proof of (2).

2.3 Optimal sampling order

Basic allows an arbitrary order on \mathcal{C} in which we take samples in Stage 2. However, since **Basic** terminates as soon as a better candidate is found, the expected total number of samples depends on a particular order in which the samples are taken. In this section, we show an optimal order on \mathcal{C} . Unfortunately, the optimal order depends on unknown parameters, and we will also discuss how these parameters should be estimated.

Let N be the number of samples required *after* we collect N_0 samples from π_0 in Stage 2, *given* $X_{0,\ell}$ for $\ell = 1, \dots, N_0$ and $X_{i,\ell}$ for $\ell = 1, \dots, n_0$ for all $\pi_i \in \mathcal{C}$. Let t be the moment immediately after we collect N_0 samples from π_0 , and let \mathcal{F}_t be all the information up to the moment t (specifically, the σ -field generated by $X_{0,\ell}$ for $\ell = 1, \dots, N_0$ and $X_{i,\ell}$ for $\ell = 1, \dots, n_0$ for all $\pi_i \in \mathcal{C}$). Also, let \mathcal{C} be ordered such that π_i is the i -th candidate that is sampled in Stage 2. Then,

$$\mathbb{E}[N] = \sum_{i=1}^k (N_i - n_0) \prod_{j=1}^{i-1} \Pr\left(\overline{X}_j^{(N_j)} < \overline{X}_0^{(N_0)} + c \mid \mathcal{F}_t\right) \quad (16)$$

since $N_i - n_0$ is counted in N iff $\overline{X}_j^{(N_j)} < \overline{X}_0^{(N_0)} + c$ for all $j = 1, \dots, i - 1$, and $\overline{X}_j^{(N_j)}$'s are independent. Below, we consider only those candidates $\pi_i \in \mathcal{C}$ such that $N_i > n_0$, since a candidate π_i with $N_i = n_0$ does not contribute to N and should be considered beforehand.

Now, let p_i be the probability that π_i is selected as a better candidate if its sample mean is compared against the standard, given \mathcal{F}_t . Namely, by (5),

$$p_i = \Pr\left(\overline{X}_i^{(N_i)} \geq \overline{X}_0^{(N_0)} + c \mid \mathcal{F}_t\right). \quad (17)$$

We claim

THEOREM 2. $\mathbb{E}[N]$ is minimized when candidates, \mathcal{C} , are ordered such that

$$(N_i - n_0)/p_i \leq (N_{i+1} - n_0)/p_{i+1} \quad (18)$$

for all consecutive pairs (π_i, π_{i+1}) in \mathcal{C} .

PROOF. Suppose there exists an order, O_{before} , on \mathcal{C} that minimizes $\mathbb{E}[N]$ but does not satisfy (18) for at least one of the consecutive pairs. Let (π_v, π_{v+1}) be the one that violates (18); i.e. $(N_v - n_0)/p_v > (N_{v+1} - n_0)/p_{v+1}$. Let N_{before} be the total number of samples needed when candidates are in the order O_{before} . Now, let O_{after} be the order where only π_v and π_{v+1} are exchanged in O_{before} . Also, let N_{after} be the total number of samples needed when candidates are in the order O_{after} . Further, let $q_i = 1 - p_i$. Then, by (16)

$$\begin{aligned} & \mathbb{E}[N_{\text{before}}] - \mathbb{E}[N_{\text{after}}] \\ & = N_v \prod_{j=1}^{v-1} q_j + N_{v+1} q_v \prod_{j=1}^{v-1} q_j - N_{v+1} \prod_{j=1}^{v-1} q_j - N_v q_{v+1} \prod_{j=1}^{v-1} q_j \\ & = (N_v (1 - q_{v+1}) - N_{v+1} (1 - q_v)) \prod_{j=1}^{v-1} q_j \\ & = (N_v p_{v+1} - N_{v+1} p_v) \prod_{j=1}^{v-1} q_j \end{aligned}$$

However, since (π_v, π_{v+1}) violates (18),

$$\mathbb{E}[N_{\text{before}}] > \mathbb{E}[N_{\text{after}}].$$

This contradicts the assumption that O_{before} is the optimal order. Hence the optimal order must satisfy (18) for all consecutive pairs (π_i, π_{i+1}) in \mathcal{C} . \square

The remaining question is how p_i should be determined. A difficulty is in that p_i depends on unknown parameters μ_i and σ_i . However, these parameters can be *estimated* based on the samples collected in Stage 1. Let Y_i be the sample

average of π_i in Stage 2; i.e. $Y_i = \frac{1}{N_i - n_0} \sum_{\ell=n_0+1}^{N_i} X_{i,\ell}$. Now, simple transformation of (17) yields

$$p_i = \Pr \left(\frac{Y_i - \mu_i}{\sqrt{\frac{\sigma_i^2}{N_i - n_0}}} \geq \frac{\frac{N_i (\overline{X_0}^{(N_0)} + c) - n_0 \overline{X_i}^{(n_0)}}{N_i - n_0} - \mu_i}{\sqrt{\frac{\sigma_i^2}{N_i - n_0}}} \middle| \mathcal{F}_t \right).$$

Note that the right hand side of the expression in $\Pr(\cdot | \mathcal{F}_t)$ is a constant, given \mathcal{F}_t . A reasonable estimate of p_i is given by replacing μ_i and σ_i^2 by $\overline{X_i}^{(n_0)}$ and S_i^2 , respectively. Since, the left hand side of the expression in $\Pr(\cdot | \mathcal{F}_t)$ is a standard normal random, Z , we now obtain an *approximation* of p_i as a value of the standard normal distribution function (19), which can be evaluated easily:

$$\begin{aligned} p_i &\approx \Pr \left(Z \geq \frac{N_i}{\sqrt{S_i^2 (N_i - n_0)}} \left(\overline{X_0}^{(N_0)} + c - \overline{X_i}^{(n_0)} \right) \right) \\ &= \Pr \left(Z \leq -\sqrt{\frac{N_i}{N_i - n_0}} \left(\overline{X_0}^{(N_0)} + c - \overline{X_i}^{(n_0)} \right) \frac{g^*}{c} \right), \end{aligned} \quad (19)$$

where the last equality follows from (4).

3. RANDOMIZED ALGORITHM

The choice of N_i in **Basic** is conservative, since it guarantees Theorem 1 for any order (in particular, the worst order) of sampling in Stage 2. For example, consider the case of ten candidates with $\mu_{10} = 10$ and $\mu_i = 0$ for $i = 1, \dots, 9$, and a standard with $\mu_0 = 2$. Let $\delta = 1$, so that we must select π_{10} as a better candidate, and any other selection is a mistake. In this setting, **Basic** is most likely to make a mistake when π_{10} is the last candidate to be sampled in Stage 2, since **Basic** needs to determine that π_i is no better than π_0 (i.e., $\overline{X_i}^{(N_i)} < \overline{X_0}^{(N_0)} + c$ needs to hold) for $i = 1, \dots, 9$ and that π_{10} is better than π_0 (i.e., $\overline{X_{10}}^{(N_{10})} \geq \overline{X_0}^{(N_0)} + c$ needs to hold). However, if π_{10} was the *first* candidate, only $\overline{X_{10}}^{(N_{10})} \geq \overline{X_0}^{(N_0)} + c$ would need to hold, and the relationship between $\overline{X_i}^{(N_i)}$ and $\overline{X_0}^{(N_0)}$ would not matter for $i = 1, \dots, 9$. Namely, we may reduce N_i by first sampling from candidate solutions that are likely to be better than the standard.

In this section, we consider a randomized algorithm, **Rand**, which takes samples in the uniformly random order of \mathcal{C} in Stage 2. Although the random order does not necessarily bias towards better candidates, with high probability it avoids biasing towards worse solutions for *any* given candidates, which allows us to reduce N_i . Specifically, **Rand** is the same as **Basic**, except that we run the for-loop of Stage 2 in the uniformly random order on \mathcal{C} and we replace the confidence parameter g^* with \hat{g}^* and the criterion ratio γ^* with $\hat{\gamma}^*$. We define \hat{g}^* and $\hat{\gamma}^*$ in Section 3.1. We then prove the following theorem in Section 3.2.

THEOREM 3. *Rand achieves (1)-(3), where π is the solution selected by Rand.*

We remark that the probabilities in Theorem 1 are over the probability space of the random numbers that are used to generate the workload that causes random samples of the simulated performance. On the other hand, the probabilities in Theorem 3 are over the probability space of the random numbers used to generate the workload and the random numbers that are used for randomization of ordering.

3.1 Confidence parameter and criterion ratio

The confidence parameter \hat{g}^* and the criterion ratio $\hat{\gamma}^*$ for **Rand** are determined in a similar way to g^* and γ^* in Section 2.1.1. We define the confidence parameter, \hat{g}^* , as

$$g^* \equiv \min_{0 < \gamma < 1} \max\{g_1(\gamma), g_2(\gamma)\},$$

where $g_1(\gamma)$ and $g_2(\gamma)$ are the unique g_1 and g_2 , respectively, that satisfy

$$f_{k, n_0-1}(\gamma g_1) = 1 - \alpha$$

$$\frac{1}{k} \sum_{i=0}^{k-1} f_{i, n_0-1}(\gamma g_2) + f_{1, n_0-1}((1 - \gamma) g_2) - 1 = 1 - \alpha,$$

for $0 < \gamma < 1$. We denote the minimizer of (7) by $\hat{\gamma}^*$. Roughly speaking, (1) and (3) are satisfied if a confidence parameter, g , is chosen such that $g > g_1(\gamma)$, and (2) is satisfied if $g > g_2(\gamma)$. Note in particular that

$$f_{k, n_0-1}(\hat{\gamma}^* \hat{g}^*) \geq 1 - \alpha \quad (20)$$

$$\frac{1}{k} \sum_{i=0}^{k-1} f_{i, n_0-1}(\hat{\gamma}^* \hat{g}^*) + f_{1, n_0-1}((1 - \hat{\gamma}^*) \hat{g}^*) - 1 \geq 1 - \alpha, \quad (21)$$

since $f_{t, \nu}(x)$ is an increasing function of x for any $t \geq 1$ and $\nu \geq 1$.

In a similar way to g^* and γ^* , a standard search algorithm can be used to find \hat{g}^* and $\hat{\gamma}^*$, where $g_1(\gamma)$ and $g_2(\gamma)$ are calculated for each given γ by a binary search algorithm.

3.2 Proof of Theorem 3

Since (1) and (3) for **Rand** can be proved similarly to those for **Basic**, we prove only (2). Consider a modified (randomized) algorithm as in the proof of Theorem 1. Recall that the modified algorithm discards a selected candidate, π , if the selected candidate is better than the standard but not the best candidate (i.e., discards $\pi \in \mathcal{B} \setminus \{\pi_b\}$). Note that **Rand** selects a correct solution if the modified randomized algorithm selects a correct solution.

Without loss of generality, let π_i be the i -th candidate among $\mathcal{W} \cup \{\pi_b\}$ from which we take samples in Stage 2 for $i = 1, \dots, |\mathcal{W}| + 1$. Note that $\Pr(\pi_i = \pi_b) = 1 / (|\mathcal{W}| + 1)$ by randomization. Now, we obtain

$$\begin{aligned} \Pr(\pi \in \mathcal{B} \mid \mu_b \geq \mu_0 + \delta) &\geq \sum_{i=1}^{|\mathcal{W}|+1} P_i \Pr(\pi_i = \pi_b) \\ &= \sum_{i=1}^{|\mathcal{W}|+1} P_i / (|\mathcal{W}| + 1) \end{aligned} \quad (22)$$

where

$$P_i = \Pr \left(\overline{X_j}^{(N_j)} < \overline{X_0}^{(N_0)} + c, \forall j = 1, \dots, i-1 \text{ and } \overline{X_i}^{(N_i)} \geq \overline{X_0}^{(N_0)} + c \mid \mu_i \geq \mu_0 + \delta \ \& \ \pi_j \in \mathcal{W}, \forall j \leq i-1 \right)$$

is the probability that the modified randomized algorithm makes correct decisions, given $\pi_i = \pi_b$, for $i = 1, \dots, |\mathcal{W}| + 1$. Now,

$$\begin{aligned} P_i &\geq \Pr \left(\overline{X_j}^{(N_j)} < \overline{X_0}^{(N_0)} + c, \forall \pi_j = \pi_1, \dots, \pi_{i-1} \in \mathcal{W} \right) \\ &\quad - \Pr \left(\overline{X_i}^{(N_i)} < \overline{X_0}^{(N_0)} + c \mid \mu_i \geq \mu_0 + \delta \right) \\ &\geq f_{i-1, n_0-1}(\hat{\gamma}^* \hat{g}^*) + f_{1, n_0-1}((1 - \hat{\gamma}^*) \hat{g}^*) - 1, \end{aligned} \quad (23)$$

where the last inequality follows from Lemmas 1 and 2. Thus, by (22) and (23),

$$\begin{aligned} & \Pr(\pi \in \mathcal{B} \mid \mu_b \geq \mu_0 + \delta) \\ & \geq \frac{1}{|\mathcal{W}| + 1} \sum_{i=1}^{|\mathcal{W}|+1} f_{i-1, n_0-1}(\hat{\gamma}^* \hat{g}^*) + f_{1, n_0-1}((1 - \hat{\gamma}^*) \hat{g}^*) - 1 \\ & \geq \frac{1}{k} \sum_{i=1}^k f_{i-1, n_0-1}(\hat{\gamma}^* \hat{g}^*) + f_{1, n_0-1}((1 - \hat{\gamma}^*) \hat{g}^*) - 1 \end{aligned}$$

where the last inequality holds since $f_{i,\nu}(x)$ is decreasing in i for any fixed ν and x . Hence, we obtain (2) by (21).

4. FIRST STAGE SELECTION AND SCREENING TO THE BEST PROCEDURES

Stage 1 of **Basic** (or **Rand**) simply collects n_0 samples from each solution, and it does not make any decisions. However, n_0 samples may suffice to determine that a candidate is better than the standard, so that the algorithm can terminate in the middle of Stage 1. Even if no better candidates are found, we may be able to determine that some candidates are worse than the standard. By screening these worse candidates out of consideration in Stage 2, we can reduce N_i 's, which are increasing functions of the number of candidates in consideration.

In this section, we propose two algorithms, which we refer to as first stage selection (FSS) and screening to the best (STB), which can substitute Stage 1 of **Basic** or **Rand**. Both FSS and STB may or may not select a better candidate or the standard. When FSS or STB selects a solution, the selection is correct with high probability. When FSS or STB does not select any solution, it provides a subset of candidates, $\mathcal{S} \subseteq \mathcal{C}$, that should be considered in Stage 2 of **Basic** (or **Rand**). More formally, the properties of FSS and STB are summarized in the following theorem.

THEOREM 4. *Let π be the solution selected by FSS or STB, and let $\mathcal{S} \subset \mathcal{C}$ be the set of candidates provided by FSS or STB when no solution is selected. Let $\pi = \emptyset$ denote that no solution is selected. Then,*

$$\Pr(\pi \in \{\pi_0, \emptyset\} \mid \mu_0 \geq \mu_b) \geq 1 - \alpha_1 \quad (24)$$

$$\begin{aligned} & \Pr((\pi = \emptyset \text{ and } \exists \pi_i \in \mathcal{S} \text{ s.t. } \mu_i \geq \mu_0 + \delta) \\ & \text{or } \pi \in \mathcal{B} \mid \mu_b \geq \mu_0 + \delta) \geq 1 - \alpha_1 \quad (25) \end{aligned}$$

$$\Pr(\pi \in \mathcal{B} \cup \{\pi_0, \emptyset\} \mid \mu_0 + \delta > \mu_b > \mu_0) \geq 1 - \alpha_1. \quad (26)$$

Note that when a solution is selected ($\pi \neq \emptyset$), the selection is correct with probability $\geq 1 - \alpha_1$. When no solution is selected ($\pi = \emptyset$), \mathcal{S} contains necessary candidates so that **Basic** can select a correct solution from $\mathcal{S} \cup \{\pi_0\}$ with high probability. Specifically, whenever there are significantly better candidates in \mathcal{C} (i.e., $\exists \pi_i \in \mathcal{C}$ such that $\mu_i \geq \mu_0 + \delta$), there is at least one significantly better candidate in \mathcal{S} . Note that when there is no significantly better candidate, π_0 is a correct selection, and \mathcal{S} does not have to contain any particular candidate.

When FSS or STB substitutes Stage 1 of **Basic** (or **Rand**), we run FSS or STB with error probability $\alpha_1 < \alpha$, where α is the overall error probability. If no solution is selected in Stage 1, we run Stage 2 of **Basic** where the error probability is $\alpha_2 = \alpha - \alpha_1$ and the set of candidate solutions is \mathcal{S} . Note

Collect n_0 samples from π_0 .
 For each $\pi_i \in \mathcal{C}$ (in any order),
 collect n_0 samples;
 calculate the sample mean, $\overline{X_{i0}}$, and variance, S_{i0}^2 , of the difference $X_{i,\cdot} - X_{0,\cdot}$;
 calculate $W_{i0} = h\sqrt{S_{i0}^2/n_0}$, where h is the smallest h that satisfies (30);
 if $\overline{X_{i0}} \geq W_{i0}$, select $\pi = \pi_i$ and exit.
 Let $\mathcal{S} = \{\pi_i \in \mathcal{C} \mid \overline{X_{i0}} > -W_{i0}\}$.
 If $\mathcal{S} = \emptyset$, select $\pi = \pi_0$.

Figure 3: The FSS algorithm

that if the algorithm is correct in Stage i with probability $\geq 1 - \alpha_i$ for $i = 1, 2$, then the algorithm is correct overall with probability $\geq 1 - (\alpha_1 + \alpha_2)$, which holds even when errors in the two stages have arbitrary correlation.

Below, we describe FSS in Section 4.1 and prove the FSS part of Theorem 4 in Section 4.2. We then describe STB in Section 4.3 and prove the STB part of Theorem 4 in Section 4.4.

4.1 First stage selection (FSS)

The key idea in FSS is to compare the sample mean of each candidate to that of the standard after n_0 samples are collected. If the difference between the sample means are large relative to the sample variance of the difference, we can determine that the candidate is better or worse than the standard. If the difference is small, FSS cannot determine whether the candidate is better or worse, and the candidate is put into \mathcal{S} . Figure 3 summarizes FSS, which we elaborate below.

FSS first takes n_0 samples from the standard π_0 , and then takes n_0 samples from each candidate in \mathcal{C} . After we collect n_0 samples from a candidate, π_i , the sample mean, $\overline{X_{i0}}$, and the sample variance, S_{i0}^2 , of the difference between a sample from π_i and a sample from π_0 are calculated via

$$\overline{X_{ij}} = \frac{1}{n_0} \sum_{\ell=1}^{n_0} (X_{i,\ell} - X_{j,\ell}) \quad (27)$$

$$S_{ij}^2 = \frac{1}{n_0 - 1} \sum_{\ell=1}^{n_0} ((X_{i,\ell} - X_{j,\ell}) - \overline{X_{ij}})^2, \quad (28)$$

where $j = 0$. FSS next determines whether $\overline{X_{i0}}$ is large enough to conclude that π_i is a better candidate. The criterion is

$$\overline{X_{i0}} \geq W_{i0} \equiv h\sqrt{S_{i0}^2/n_0}, \quad (29)$$

where h is the smallest h such that

$$\Pr(T_{n_0-1} \geq h) \leq \alpha_1/k, \quad (30)$$

where T_ν has the t-distribution with degree of freedom ν . It will turn out that the probability that the worse candidate (π_i with $\mu_i \leq \mu_0$) satisfies Criterion (29) is the same as $\Pr(T_{n_0-1} \geq h)$ (see Section 4.2). Since there are at most k worse candidates, the probability of selecting a worse candidate will be bounded by α_1 .

If no better candidate is found, FSS determines a set of candidates, \mathcal{S} , whose sample mean is not too small (and

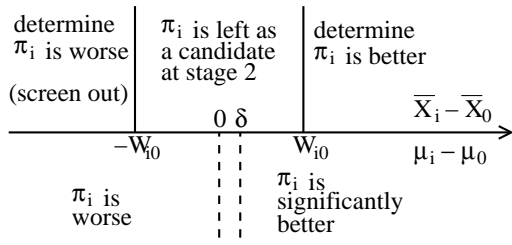


Figure 4: The choice of W_{i0} in the FSS algorithm.

hence the candidate *may be* better than the standard) via

$$\mathcal{S} = \{\pi_i \in \mathcal{C} \mid \bar{X}_{i0} > -W_{i0}\}.$$

The candidates that are not in \mathcal{S} are determined to be worse than the standard. If \mathcal{S} is empty, FSS selects the standard ($\pi = \pi_0$). Figure 4 provides a further intuition behind the choice of Criterion (29). Note that, unlike the threshold, c , in Criterion (5), the threshold, W_{i0} , in Criterion (29) is not a fixed constant but is a function of S_{i0}^2 .

Note that the candidates may be sampled in any order in FSS. For the same reason as discussed in Section 2.3, it is better to first sample from candidates that are more likely to be better than the standard. Thus, if we have prior information as to which candidates are more likely to be better, the candidates should be reordered.

4.2 Proof of Theorem 4 (FSS part)

In this section, we prove (24)-(26) for FSS. First consider (24), where $\mu_0 \geq \mu_b$ is assumed. If $\bar{X}_{i0} \leq W_{i0}, \forall \pi_i \in \mathcal{C}$, no candidate is selected ($\pi \in \{\emptyset, \pi_0\}$). Thus, it suffices to prove that

$$P \equiv \Pr(\bar{X}_{i0} \leq W_{i0}, \forall \pi_i \in \mathcal{C} \mid \mu_0 \geq \mu_i, \forall \pi_i \in \mathcal{C}) \geq 1 - \alpha_1.$$

Note that $\bar{X}_{i0} \leq W_{i0}$ is equivalent to

$$\frac{\sqrt{n_0}(\bar{X}_{i0} - (\mu_i - \mu_0))}{\sqrt{S_{i0}^2}} \leq \sqrt{\frac{n_0}{S_{i0}^2}}(W_{i0} - (\mu_i - \mu_0)). \quad (31)$$

Since the left hand side of (31) is a random variable, $T_{n_0-1}^{(i)}$, having the t-distribution with degree of freedom $n_0 - 1$ and $\mu_0 \geq \mu_i, \forall \pi_i \in \mathcal{C}$,

$$P \geq \Pr\left(T_{n_0-1}^{(i)} \leq \sqrt{n_0/S_{i0}^2} W_{i0}, \forall \pi_i \in \mathcal{C}\right).$$

Now, by (29),

$$P \geq \Pr\left(T_{n_0-1}^{(i)} \leq h, \forall \pi_i \in \mathcal{C}\right) \geq 1 - \sum_{i=1}^k \Pr(T_{n_0-1}^{(i)} \geq h).$$

Now, $P \geq 1 - \alpha_1$ follows from (30). This proves (24).

Inequalities (25)-(26) can be proved similarly. Consider the case where $\mu_b \geq \mu_0 + \delta$. Note that $\bar{X}_{i0} \leq W_{i0}, \forall \pi_i \in \mathcal{W}$ and $\bar{X}_{b0} > -W_{b0}$ are sufficient for FSS to make correct decisions. Via a similar argument as in the proof of (24),

$$\begin{aligned} \Pr(\bar{X}_{i0} \leq W_{i0}, \forall \pi_i \in \mathcal{W} \text{ and } \bar{X}_{b0} > -W_{b0}) \\ \geq 1 - |\mathcal{W} + 1| \Pr(T_{n_0-1} \geq h) \\ \geq 1 - \alpha_1. \end{aligned}$$

This proves (25). Next, consider the case where $\mu_0 + \delta > \mu_b > \mu_0$. Note that $\bar{X}_{i0} \leq W_{i0}, \forall \pi_i \in \mathcal{W}$ is sufficient for FSS

Collect n_0 samples of π_i for each $\pi_i \in \mathcal{A}$.
 For each $\pi_i, \pi_j \in \mathcal{A}$,
 calculate the sample mean, \bar{X}_{ij} , and variance, S_{ij}^2 ,
 of the difference $X_{i,\cdot} - X_{j,\cdot}$;
 calculate $W_{ij} = h\sqrt{S_{ij}^2/n_0}$, where h is the smallest
 h that satisfies (30).
 Let $\mathcal{S} = \{\pi_i \in \mathcal{C} \mid \bar{X}_{ij} > -W_{ij}, \forall \pi_j (\neq \pi_i) \in \mathcal{A}\}$.
 If $\mathcal{S} = \emptyset$, select $\pi = \pi_0$.

Figure 5: The STB algorithm

to make correct decisions. Again, via a similar argument as in the proof of (24), (26) now follows from

$$\begin{aligned} \Pr(\bar{X}_{i0} \leq W_{i0}, \forall \pi_i \in \mathcal{W}) &\geq 1 - |\mathcal{W}| \Pr(T_{n_0-1} \geq h) \\ &\geq 1 - \alpha_1. \end{aligned}$$

4.3 Screening to the best (STB)

A motivation behind STB is that a candidate may be screened out, even if its sample mean is not (far) smaller than that of the standard, if the sample mean is far smaller than that of *another candidate*. Note that when there are better candidates, \mathcal{S} does not have to contain *all* of the candidates that may be better than the standard, but it suffices to contain only the best candidate. STB finds a set of candidates, \mathcal{S} , that may be the best from \mathcal{A} . If \mathcal{S} is empty, STB selects π_0 . Figure 4.3 summarizes STB, which we elaborate below.

STB first collects n_0 samples from each solution in \mathcal{A} . Then, for each pair of solutions in \mathcal{A} , the sample mean, \bar{X}_{ij} , and the sample variance, S_{ij} , of the difference between a sample from π_i and a sample from π_j are calculated via (27)-(28). STB then determines whether \bar{X}_{ij} is large enough to conclude that π_j is *not* the best solution. The criterion is

$$\bar{X}_{ij} > -W_{ij} \equiv -h\sqrt{S_{ij}^2/n_0}, \quad (32)$$

where h is the smallest h such that (30). If (32) holds for at least one $\pi_i \in \mathcal{A}$, π_j is not included in \mathcal{S} .

As a result, \mathcal{S} contains only candidate solutions that are likely to be the best. If $\mathcal{S} = \emptyset$, STB selects $\pi = \pi_0$.

4.4 Proof of Theorem 4 (STB part)

Since STB never selects a candidate, (24) and (26) obviously hold (in fact, $\Pr(\pi \in \{\emptyset, \pi_0\}) = 1$). To show (25), it suffices to prove that the best candidate, π_b , is in \mathcal{S} with high probability. Note that $\pi_b \in \mathcal{S}$ if $\bar{X}_{bi} > -W_{bi}$ for all solutions $\pi_i \in \mathcal{A} \setminus \{\pi_b\}$. Now, via the same argument as the proof of (24) for FSS (see Section 4.2),

$$\begin{aligned} \Pr(\bar{X}_{bi} > -W_{bi}, \forall \pi_i (\neq \pi_b) \in \mathcal{A}) &\geq 1 - k \Pr(T_{n_0-1} \geq h) \\ &\geq 1 - \alpha_1, \end{aligned}$$

which completes the proof.

5. RESULTS

In this section, we present and discuss the effectiveness of the proposed algorithms with respect to the total number of samples needed, comparing them to NG, the Nelson-Goldman algorithm [8]. Although our analysis in prior sections guarantees that our algorithms make correct selections

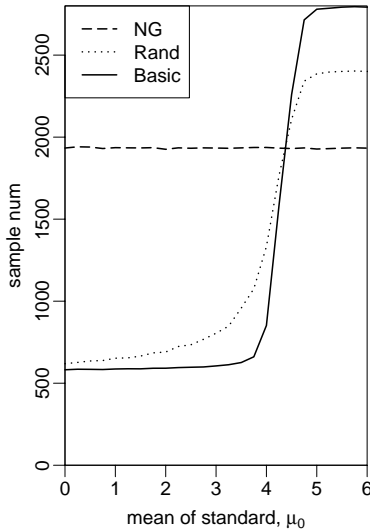


Figure 6: Comparison of Basic, Rand, and NG. The total number of samples needed in each algorithm is plotted against the mean of the standard, μ_0 . The means of the 20 candidates are equally spaced between 0.25 and 5. $\sigma_i = 2$ is fixed for all $\pi_i \in \mathcal{A}$.

with high probability ($\geq 1 - \alpha$), the total number of samples needed in each algorithm is not *explicitly* shown. Throughout, we fix the upper bound of the overall error probability $\alpha = 0.2$, the width of the indifference zone $\delta = 1$, and the number of candidates $k = 20$. The means of the 20 candidates, \mathcal{C} , are chosen between 0.25 and 5 such that they are equally spaced (specifically, the i -th smallest mean is $i/4$ for $i = 1, \dots, 20$). In all figures, the total number of samples needed in each algorithm is plotted against the mean of the standard μ_0 , which varies between 0 and 6. Since the number of samples depends on particular samples from the solutions, we take the average of 1,000 runs to generate each data point. Settings of the σ_i 's are specified in each figure.

Although the experimental settings in this section do not fully represent typical instances that appear in local search for optimizing stochastic systems, our experiments illuminate the advantages and disadvantages of our algorithms and NG. The results of our experiments lead to guidelines on designing good local search algorithms for optimizing stochastic systems.

5.1 Selecting a better can be much faster

Figure 6 shows the total number of samples needed in **Basic**, **Rand**, **NG** as a function of μ_0 . The solid line corresponds to **Basic** with the optimal sampling order on \mathcal{C} (and with an approximation of p_i), introduced in Section 2.3. Also, the dotted line corresponds to **Rand**, and the dashed line corresponds to **NG**. Here, $\sigma_i = 2$ is fixed for all $\pi_i \in \mathcal{A}$.

When the standard is poor (small μ_0), **Basic** requires far fewer samples than **NG**. This is to be expected, since **Basic** terminates as soon as one of the many better candidates is found, while **NG** collects samples from all of the candidates.

As the standard becomes better (larger μ_0), **Basic** requires more samples, while the number of samples needed by **NG** does not change. Again, this is to be expected. When μ_0 is larger, there are a smaller number of better candidates,

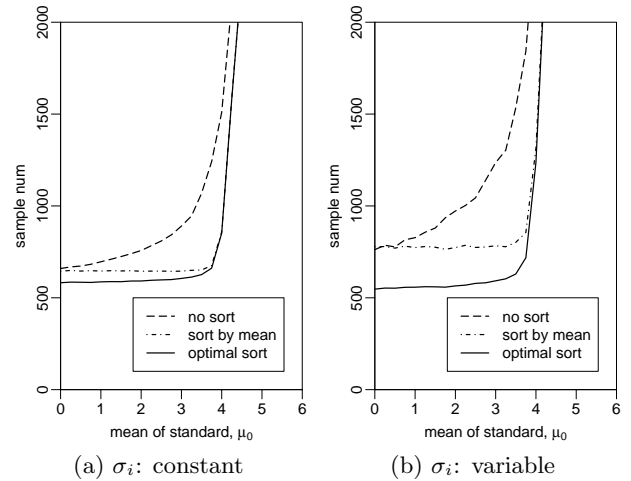


Figure 7: Effectiveness of optimal sorting. The total number of samples needed is plotted against the mean of the standard, μ_0 , for **Basic** with no sorting, sorting by the sample mean, and optimal sorting. The parameter settings are the same as in Figure 6, except that $\sigma_i = 2$ in (a) and σ_i is chosen uniformly at random between 1 and 4 in (b) for all solutions $\pi_i \in \mathcal{A}$.

and **Basic** needs to take samples from a larger number of candidates in Stage 2 before it finds a better candidate. On the other hand, **NG** takes samples from all of the candidates regardless of the value of μ_0 .

When the standard is good (large μ_0), **Basic** requires more samples than **NG**. This may be counterintuitive, since **Basic** always takes samples from a smaller or the same number of solutions than **NG** in Stage 2. However, **Basic** turns out to require a larger N_i for each solution. Intuitively, this is because **Basic** makes decisions (determines whether a candidate is better than the standard or not) multiple times after each candidate is sampled, while **NG** makes a decision (determines the best solution) once after all of the candidates are sampled. As a result, **Basic** is more likely to make a mistake than **NG** if the same number of samples are collected from each solution. In addition, N_i is simply an *upper bound* on the number of samples needed to satisfy the error bound $1 - \alpha$, and smaller N_i may suffice³.

Figure 6 also shows that **Rand** requires a smaller number of samples than **Basic** when μ_0 is large. This is to be expected, since both **Basic** and **Rand** take samples from most or all of the candidates, and N_i in **Rand** is smaller than **Basic** for each $\pi_i \in \mathcal{A}$. When μ_0 is small, however, **Basic** requires fewer samples than **Rand**, since **Basic** takes samples in the optimal order on \mathcal{C} and hence takes samples from fewer solutions in Stage 2.

5.2 Optimal sorting is effective

Figure 7 illustrates the effectiveness of the optimal sampling order introduced in Section 2.3 by plotting the total number of samples needed in **Basic** with three different sorting schemes. Solid lines correspond to **Basic** with optimal

³Our experiments suggest that the actual error probability of **Basic** is slightly smaller than that of **NG**, which suggests that the N_i in **Basic** can be made smaller.

sorting (and with an approximation of p_i), and dashed lines correspond to the one without sorting (which is equivalent to a random order in our experimental settings). We also consider the order where the candidates are in decreasing order of the sample mean (dotted-dashed lines). Sorting by the sample mean makes intuitive sense, since it biases towards candidates that are likely to be better than the standard. However, sorting by the sample mean is insufficient, since it does not take into account the number of samples needed for each candidate. For example, sorting by the sample mean may start sampling from a candidate that are likely to be better than the standard but requires many samples (large N_i). By comparing optimal sorting and sorting by the sample mean, we can isolate the effect of N_i in Condition (18) of the optimal order.

Figure 7(a) shows the number of samples needed by each sorting scheme when $\sigma_i = 2$ for all $\pi_i \in \mathcal{A}$. For clarity, the vertical axis is truncated at 2,000. As can be expected, the number of samples needed when the candidates are in the optimal order is far (10-45%) smaller than that when the candidates are not sorted. However, the difference between optimal sorting and sorting by the sample mean is quite small ($\leq 10\%$). This is because σ_i is identical for all candidates, which in turn implies that N_i is *stochastically* identical for all candidates. Observe that optimal sorting reduces to sorting by the sample mean if N_i was identical for all $\pi_i \in \mathcal{A}$.

In Figure 7(b), we vary σ_i between 1 and 4 by choosing each σ_i uniformly at random between 1 and 4 for each run of the 1,000 runs for a data point. When σ_i varies, sorting by the sample mean requires far (20-45%) more samples than sorting in the optimal order. This makes intuitive sense, since the optimal order now depends heavily on N_i , which is not taken into account in sorting by the sample mean. In extreme cases, where better candidates have larger σ_i 's, the advantage of optimal sorting against sorting by the sample mean becomes even more significant.

However, an additional benefit of sorting by the sample mean is that, in practice, it selects one of *nearly best* solutions (π_i such that $\mu_i \geq \mu_b - \delta$) with high probability. This is because the candidate with the largest sample mean at Stage 1 is very likely to be one of the nearly best solutions, and it is the first solution to be sampled at Stage 2 when the candidates are sorted by their sample means.

5.3 FSS is usually more effective than STB

Figure 8 illustrates the effectiveness of FSS and STB when they are used in **Basic** and in **NG**. Here, σ_i is chosen to be relatively small, $\sigma_i = 2$, in columns (a)-(b), and relatively large, $\sigma_i = 8$, in columns (c)-(d), for all $\pi_i \in \mathcal{A}$.

Figure 8(a) shows the total number of samples needed in **Basic** (solid line) as well as in **Basic** when its Stage 1 is replaced with FSS (dashed line) or with STB (dotted-dashed line). Overall, FSS can reduce the total number of samples substantially for a range of μ_0 . In particular, the reduction is by orders of magnitude, when μ_0 is small. This is to be expected, since FSS can terminate after sampling one of the far better candidates in the middle of Stage 1. The benefit of FSS diminishes as μ_0 becomes larger, since it becomes less likely that FSS finds a better candidate. However, when μ_0 becomes very large, the benefit of FSS becomes significant again. This is because FSS can now screen out a large number of poor candidates.

We can also observe that STB is not as effective as FSS, particularly when μ_0 is small. In fact, the use of STB can slightly increase the total number of samples needed. This counter effect stems from the fact that the error bound, α_2 , in Stage 2 of **Basic** with STB needs to be set smaller than α , and smaller error bound implies larger N_i for each candidate, $\pi_i \in \mathcal{S}$, considered in Stage 2. Also, since STB does not terminate in the middle of Stage 1, it does not provide the huge benefit which FSS does. When μ_0 is large, however, STB can reduce the total number of samples. This is because the benefit of screening becomes larger when μ_0 is larger, as it becomes more likely that most or all of the candidates considered in Stage 2 are sampled, and a smaller number of candidates, \mathcal{S} , implies smaller N_i .

Overall, FSS can provide much larger benefit than STB when μ_0 is small. When μ_0 is large, STB can become more beneficial than FSS, but the difference between the two is insignificant.

Figure 8(b) shows the total number of samples needed in **NG** (solid line) and in **NG** when it is used with STB (dotted-dashed line). In our settings, STB reduces the total number of samples roughly by half. Taking a close look, we can observe that the number of samples in **NG** with STB has a small peak around $\mu_0 = 4$. This can be explained as follows. When μ_0 is small, the standard is likely to be screened out as well as poor candidates, but when μ_0 is large, only poor candidates are screened out. Also, as μ_0 becomes larger than μ_b , a larger number of poor candidates are screened out, since the best solution (i.e., π_0) becomes better.

In Figures 8(c)-(d), we set σ_i to a relatively large value, $\sigma_i = 8$, for all $\pi_i \in \mathcal{A}$. The advantages of FSS and STB are diminished in both **Basic** and **NG**. For most values of μ_0 , the use of FSS and STB increase the total number of samples. This is because few solutions are screened out and few candidates are found to be better than the standard at Stage 1, while the error bound at Stage 2 needs to be set smaller when STB or FSS is used in Stage 1.

5.4 Guidelines

The results of our experiments (including those not shown) lead to guidelines on designing good local search algorithms for stochastic systems. We start by summarizing our findings in our experiments.

- When there exists a significantly better candidate ($\mu_b \geq \mu_0 + \delta$), **Basic** with FSS is a good choice (see Figure 8(a)), and the candidates should be sampled in the optimal order introduced in Section 2.3 (see Figure 7). When the solutions are highly variable (σ_i and σ_0 are large relative to $\mu_i - \mu_0$), however, the benefit of FSS diminishes, and **Basic** alone may be better (see Figure 8(c)).
- When no candidates are better than the standard ($\mu_b \leq \mu_0$), **NG** possibly with STB is a good choice. When there are sufficiently poor candidates ($\mu_0 - \mu_i$ is large relative to σ_0 and σ_i), **NG** should be used with STB (see Figure 8(b)). Otherwise, **NG** alone is better, but the use of STB does not hurt much. (see Figure 8(d)).

In the context of local search for stochastic systems, our experience suggests that there are often many neighborhood solutions that are significantly better than the current solution at initial phases of local search. Hence, we expect that **Basic** with FSS performs well at the beginning of local search. Also, the candidates should be in the optimal or-

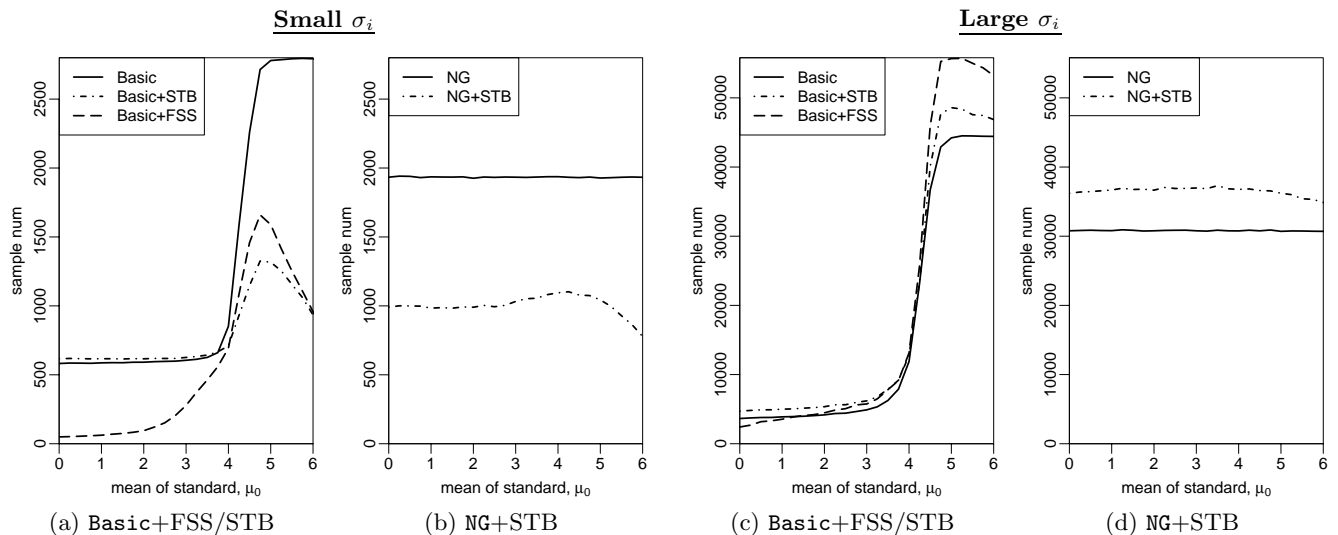


Figure 8: Effectiveness of FSS and STB. The total number of samples needed is plotted against the mean of the standard, μ_0 . The parameter settings are identical to those in Figure 6, except that $\sigma_i = 2$ in (a)-(b) and $\sigma_i = 8$ in (c)-(d) for all solutions $\pi_i \in \mathcal{A}$. In (a) and (c), the effectiveness of FSS and STB is shown when it is used in Basic. In (b) and (d), the effectiveness of STB is shown when it is used in NG.

der introduced in Section 2.3 to minimize the total number of samples needed. Sorting the candidates by their sample means from Stage 1 is another possibility, since in practice it selects one of the best neighborhood solutions with high probability (see Section 5.2). Further study on the selection of a sorting scheme is left as future work.

As local search proceeds and the current solution becomes better, there may be fewer better neighborhood solutions. Thus, we expect that NG possibly with STB performs better at final phases of local search. Whether STB is beneficial or not depends on particular stochastic systems, but STB is worth considering since it can provide substantial benefit.

6. CONCLUSION

In this paper, we propose first algorithms for quickly finding a candidate solution that is better than the standard solution with high probability, where the performance of a solution is estimated via simulations or experiments. To reduce the total number of samples, which determines the total simulation time, various heuristics are proposed, including sorting in the optimal order of sampling, randomization, first stage selection (FSS), and screening to the best (STB). The proposed algorithms are compared to the Nelson-Goldman algorithm (NG), which finds the *best* solution. Our experiments show that the number of samples needed in our algorithms can be orders of magnitude smaller than that in NG when there is a significantly better candidate ($\mu_b \geq \mu_0 + \delta$). In addition, we find that our algorithm can select one of the *best* solutions with high probability in practice, by first sampling from the candidates that are likely to be the best.

Our two-stage algorithms may be extended to more than two stages or to a fully sequential algorithm, where FSS is applied at each stage until a better candidate is found or only the standard is left. We expect that a fully sequential algorithm can further reduce the total simulation time. However, two-stage algorithms have advantages when there

is a cost in changing the solutions to be simulated. For example, we may want to checkpoint the system state of solution π_i so that we can resume simulating π_i from the checkpoint.

Alternatively, our two-stage algorithms can be transformed into a single stage, where the standard solution is compared against the candidate solutions one after another. Although a single-stage algorithm can minimize the number of changes in solutions to be simulated, it requires more samples to find a better candidate particularly when there are many better candidates. For example, a single-stage algorithm cannot take advantage of the benefit of FSS, which is quite effective in the basic algorithm when there are many better candidates.

Our algorithms can substantially accelerate local search for optimizing the performance of stochastic systems, where a bottleneck is the simulation time required to find a neighborhood solution that is better than the current solution. Future work includes applying the proposed algorithms to performance optimization of real computer and communication systems. We believe that the guidelines provided in this paper will help in designing good local search algorithms.

7. REFERENCES

- [1] E. H. L. Aarts and J. K. Lenstra, editors. *Local Search in Combinatorial Optimization*. John Wiley & Sons, 1997.
- [2] R. E. Bechhofer, T. J. Santner, and D. M. Goldsman. *Design and analysis of experiments for statistical selection, screening, and multiple comparisons*. John Wiley & Sons, 1995.
- [3] E. J. Dudewicz and S. R. Dalal. Allocation of observations in ranking and selection with unequal variances. *Sankhya*, B37(1):28–78, 1975.
- [4] G. Grimmett and D. Stirzaker. *Probability and Random Processes*. Oxford University Press, third edition, 2001.

- [5] S. Kim and B. L. Nelson. A fully sequential procedure for indifference-zone selection in simulation. *ACM Transactions on Modeling and Computer Simulation*, 11(3):251–273, 2001.
- [6] S. Kotz and N. L. Johnson, editors. *Encyclopedia of Statistical Sciences*. John Wiley & Sons, second edition, 2005.
- [7] A. M. Law and W. D. Kelton. *Simulation Modeling and Analysis*. McGraw-Hill, third edition, 2000.
- [8] B. L. Nelson and D. Goldsman. Comparisons with a standard in simulation experiments. *Management Science*, 47(3):449–463, 2001.
- [9] B. L. Nelson, J. Swann, D. Goldsman, and W. Song. Simple procedures for selecting the best simulated system when the number of alternatives is large. *Operations Research*, 49(6):950–963, 2001.
- [10] J. Pichitlamken and B. Nelson. A combined procedure for optimization via simulation. *ACM Transactions on Modeling and Computer Simulation*, 13(2):155–179, 2003.
- [11] Y. Rinott. On two-stage selection procedures and related probability-inequalities. *Communications in Statistics — Theory and Methods*, A7(8):799–811, 1978.
- [12] J. R. Swisher, S. H. Jacobson, and E. Yücesan. Discrete-event simulation optimization using ranking, selection, and multiple comparison procedures: Survey. *ACM Transactions on Modeling and Computer Simulation*, 13(2):134–154, 2003.
- [13] B. Xi, Z. Liu, M. Raghavachari, C. H. Xia, and L. Zhang. A smart hill-climbing algorithm for application server configuration. In *Proceedings of the 13th International Conference on World Wide Web*, pages 287–296, 2004.
- [14] T. Ye and S. Kalyanaraman. A recursive random search algorithm for large-scale network parameter configuration. In *Proceedings of the ACM SIGMETRICS*, pages 196–205, 2003.

APPENDIX: PROOF OF LEMMAS 1 AND 2

In this section, we prove Lemmas 1 and 2, which hinges on the following well known results.

PROPOSITION 1. [4] Let \bar{X} and S^2 be the sample mean and variance, respectively, of n independent samples from a normal random variable with mean μ and variance σ^2 . Then, $Z = \frac{\bar{X} - \mu}{\sqrt{\sigma^2/n}}$ has the standard normal distribution, and $Q = (n-1)S^2/\sigma^2$ has the χ^2 distribution with degree of freedom $n-1$. Also, Z and Q are independent.

Proof of Lemma 1

Lemma 1 can be proved via the techniques used in [8]. Let $\mu_0 \geq \mu_i, \forall \pi_i \in \mathcal{D} \subseteq \mathcal{C}$. We prove

$$\Pr\left(\bar{X}_i^{(N_i)} < \bar{X}_0^{(N_0)} + c, \forall \pi_i \in \mathcal{D}\right) \geq f_{|\mathcal{D}|, n_0-1}(\gamma^* g^*) \quad (33)$$

where $f_{|\mathcal{D}|, n_0-1}(\cdot)$ is Rinott’s distribution function defined by (6), and g^*, γ^* , and c are the constants used in **Basic**.

Observe that

$$\begin{aligned} & \Pr\left(\bar{X}_i^{(N_i)} < \bar{X}_0^{(N_0)} + c, \forall \pi_i \in \mathcal{D}\right) \\ & \geq \Pr\left(\bar{X}_i^{(N_i)} - \bar{X}_0^{(N_0)} - (\mu_i - \mu_0) < c, \forall \pi_i \in \mathcal{D}\right) \\ & = \Pr\left(\frac{\bar{X}_i^{(N_i)} - \bar{X}_0^{(N_0)} - (\mu_i - \mu_0)}{\sqrt{\frac{\sigma_i^2}{N_i} + \frac{\sigma_0^2}{N_0}}} < \frac{c}{\sqrt{\frac{\sigma_i^2}{N_i} + \frac{\sigma_0^2}{N_0}}}, \forall \pi_i \in \mathcal{D}\right) \\ & \geq \Pr\left(\frac{\bar{X}_i^{(N_i)} - \bar{X}_0^{(N_0)} - (\mu_i - \mu_0)}{\sqrt{\frac{\sigma_i^2}{N_i} + \frac{\sigma_0^2}{N_0}}} < \frac{\gamma^* g^*}{\sqrt{\frac{\sigma_i^2}{S_i^2} + \frac{\sigma_0^2}{S_0^2}}}, \forall \pi_i \in \mathcal{D}\right) \end{aligned}$$

where the first inequality follows from $\mu_0 \geq \mu_i, \forall \pi_i \in \mathcal{D}$, and the last inequality follow from (4). Now Proposition 1 can be used to show that

$$\begin{aligned} & \Pr\left(\bar{X}_i^{(N_i)} < \bar{X}_0^{(N_0)} + c, \forall \pi_i \in \mathcal{D}\right) \\ & \geq \Pr\left(Z^{(i)} < \frac{\gamma^* g^*}{\sqrt{(n_0-1)(1/Q^{(i)} + 1/Q^{(0)})}}, \forall \pi_i \in \mathcal{D}\right), \end{aligned}$$

where $Z^{(i)}$ is a standard normal random variable and $Q^{(i)}$ is a χ^2 random variable with degree of freedom n_0-1 . Observe that the covariance of $Z^{(i)}$ and $Z^{(j)}$ is positive for any i and j . By the positive correlation, Slepian’s inequality [6] implies (33), using the definition of $f_{t,\nu}(\cdot)$.

Proof of Lemma 2

Let $\mu_b \geq \mu_0 + \delta$. We prove

$$\Pr\left(\bar{X}_b^{(N_b)} < \bar{X}_0^{(N_0)} + c\right) \leq 1 - f_{1, n_0-1}((1-\gamma^*)g^*) \quad (34)$$

where $f_{|\mathcal{D}|, n_0-1}(\cdot)$ is Rinott’s distribution function defined by (6), and g^*, γ^* , and c are the constants used in **Basic**. Observe that

$$\begin{aligned} & \Pr\left(\bar{X}_b^{(N_b)} < \bar{X}_0^{(N_0)} + c\right) \\ & \leq \Pr\left(\bar{X}_b^{(N_b)} - \bar{X}_0^{(N_0)} - (\mu_b - \mu_0) < c - \delta\right) \\ & = \Pr\left(\frac{\bar{X}_b^{(N_b)} - \bar{X}_0^{(N_0)} - (\mu_b - \mu_0)}{\sqrt{\frac{\sigma_b^2}{N_b} + \frac{\sigma_0^2}{N_0}}} < \frac{c - \delta}{\sqrt{\frac{\sigma_b^2}{N_b} + \frac{\sigma_0^2}{N_0}}}\right) \\ & \leq \Pr\left(\frac{\bar{X}_b^{(N_b)} - \bar{X}_0^{(N_0)} - (\mu_b - \mu_0)}{\sqrt{\frac{\sigma_b^2}{N_b} + \frac{\sigma_0^2}{N_0}}} < -\frac{(1-\gamma^*)g^*}{\sqrt{\frac{\sigma_b^2}{S_b^2} + \frac{\sigma_0^2}{S_0^2}}}\right) \end{aligned}$$

where the first inequality follow from $\mu_b \geq \mu_0 + \delta$, and the last inequality follows from (4) and $c = \gamma^* \delta$. Now Proposition 1 can be used to show that

$$\Pr\left(\bar{X}_b^{(N_b)} < \bar{X}_0^{(N_0)} + c\right) \leq \Pr\left(Z < -\frac{(1-\gamma^*)g^*}{\sqrt{\frac{n_0-1}{Q^{(1)}} + \frac{n_0-1}{Q^{(0)}}}}\right),$$

where Z has the standard normal distribution, and $Q^{(i)}$ has the χ^2 distribution with degree of freedom n_0-1 for $i = 1, 2$. Then, (34) follows from the definition of $f_{t,\nu}(\cdot)$.