

Finding probably best system configurations quickly

Takayuki Osogami
IBM Tokyo Research Laboratory
1623-14 Shimotsuruma
Yamato-shi, Kanagawa 242-8502, Japan.

1. INTRODUCTION

Computer systems often have many possible configurations, and designing a high performance system often requires selecting the best configuration. Unfortunately, the performance of complex systems can often be estimated only via simulations, or with measurements of real systems. Since longer simulation times are required to estimate the performance more accurately, it is often computationally intractable to estimate the performance of all configurations accurately via simulations. (Measurements of real systems can take even longer.)

Our goal is to design an algorithm for finding the best configuration *without estimating the performance of each configuration accurately*, so that the total simulation time is minimized. It is also desirable that the algorithm does not require frequently changing the configurations to be simulated, even though frequent configuration changes can reduce the total simulation time. For example, when we want to find the configuration having the best *steady state* performance, frequent configuration changes can result in wasting much of the simulation time in transient (non-steady) states. Although the system state could be checkpointed when configurations are changed so that simulation can be resumed from the checkpoint, frequent checkpointing can significantly increase the complexity of the algorithm. Algorithms with infrequent configuration changes are also suitable when the performance is estimated with measurements of real systems, where configuration changes are more difficult than in simulations.

Prior work

The problem of selecting the best configuration, where the performance is estimated via simulations, has been studied as Ranking and Selection in the literature [4]. It is standard to assume that simulating configuration π_i yields samples from a normal distribution with mean μ_i and standard deviation σ_i for each π_i , since samples in most simulations can be made arbitrarily close to independent normal random variables by appropriate batching [4] (see also [7] or a longer version [6] of this abstract). Under this assumption, a ranking and selection algorithm finds, with high probability ($\geq 1 - \alpha$ for a given α), the configuration π_b having the largest mean performance μ_b , so that the total number of samples is minimized. It is common to accept an error within an *indifference zone* δ , so that any configuration whose mean performance is $> \mu_b - \delta$ is considered to be one of the “best” configurations.

Dudewicz and Dalal [1] and Rinott [9] propose two-stage algorithms for finding the best configuration. Stage 1 collects a fixed number of samples, and determines the number of samples needed for each configuration. In Stage 2, precisely this number of samples are collected, and the configuration with the largest sample mean is selected as the best configuration. Nelson *et al.* [5] propose a

screening procedure, which eliminates clearly poor configurations after Stage 1, that can be used with the two-stage algorithms. The two-stage algorithm with screening can be extended to more than two stages or to a sequential-stage algorithm, where a screening procedure is applied at each stage until only the best configuration is left [3]. Although sequential-stage algorithms can reduce the total number of samples, they require changing the configurations to be simulated for *every* sample.

To balance the number of samples and the number of configuration changes, Hong and Nelson propose a two-stage algorithm that makes decisions sequentially [2], which we refer to as the HN algorithm. HN interrupts simulation at most once for each configuration (two-stage). Also, HN stops sampling from a configuration at any time during Stage 2 when sufficient samples have been collected to make correct decisions (sequential-decision). Due to this sequential decision making, HN uses far fewer samples than classical two-stage algorithms [1, 9].

Contributions

Our primary contribution is a two-stage sequential-decision (TSSD) algorithm that improves upon HN. TSSD uses far fewer samples than HN, and TSSD changes the configurations to be simulated as infrequently as HN. The key idea in TSSD is to take advantage of the low variability of the sample mean after sampling at Stage 2, which is not fully utilized in HN. We will show that the number of samples required in TSSD is up to 60% fewer than HN, while the two algorithms make roughly the same numbers of configuration changes. Also, the number of configuration changes in TSSD is orders of magnitude smaller ($\sim 1/200$) than the sequential-stage algorithm by Kim and Nelson (KN) [3], although TSSD requires slightly more samples (0-20% more samples) than KN.

2. TSSD AND ITS PROPERTIES

TSSD selects, with high probability, one of nearly best configurations from the set of candidate configurations, $\mathcal{C} = \{\pi_1, \dots, \pi_k\}$. The performance of configuration π_i is assumed to have a normal distribution with mean μ_i and standard deviation σ_i for each $\pi_i \in \mathcal{C}$. Let π_b be any one of the best configurations such that $\mu_b \geq \mu_i, \forall \pi_i \in \mathcal{C}$. Let $\mathcal{B} = \{\pi_i \in \mathcal{C} \mid \mu_i > \mu_b - \delta\}$ be the set of nearly best configurations. Formally, the following theorem characterize the properties of TSSD:

THEOREM 1. *Let π be the configuration selected by TSSD. If $\mathcal{B} = \{\pi_b\}$, then $\Pr(\pi = \pi_b) \geq 1 - \alpha$. If $|\mathcal{B}| \geq 2$, then $\Pr(\pi \in \mathcal{B}) \geq 1 - \frac{k - |\mathcal{B}|}{k - 1} \alpha$.*

A proof of the theorem is provided in [6]. Below, we describe TSSD in Section 2.1, and evaluate the effectiveness of TSSD via numerical experiments in Section 2.2.

2.1 The TSSD Algorithm

TSSD consists of two stages. Stage 1 collects a fixed number, n_0 , of samples and calculates the sample mean, $\bar{X}_i^{(n_0)}$, and sample variance, S_i^2 , for each $\pi_i \in \mathcal{C}$. In theory, n_0 can be any integer ≥ 2 , but too small an n_0 can result in a greater number of samples in Stage 2. For example, we set $n_0 = 20$ for our experiments in Section 2.2. At the end of Stage 1, we sort the configurations \mathcal{C} in the decreasing order of their sample means, so that the better configurations are more likely to be examined before the others in Stage 2.

Stage 2 first determines an upper bound on the number of samples to be collected from each $\pi_i \in \mathcal{C}$ via $N_i = \max \{n_0, \lceil (h S_i / \delta)^2 \rceil\}$. Here, h is a confidence parameter, which is defined as the smallest h satisfying

$$\mathbb{E} \left[\left(1 + \exp \left(\sqrt{\frac{h^2 Q_1}{n_0 - 1} \left(1 + \frac{Q_1}{Q_2} \right) Z + \frac{h^2 Q_1}{n_0 - 1}} \right) \right)^{-1} \right] \leq \frac{\alpha}{k - 1}, \quad (1)$$

where Z is a standard normal random variable, Q_1 and Q_2 are χ^2 random variables with degree of freedom $n_0 - 1$, and the three random variables are independent. A further explanation on the confidence parameter h is provided in Section 2.1.1.

Next we collect $N_1 - n_0$ samples from π_1 , and calculate the sample mean, $\bar{X}_1^{(N_1)}$, of the N_1 samples. Let $\pi' = \pi_1$ be the provisional best, and let $\bar{X}' = \bar{X}_1^{(N_1)}$ be the sample mean of π' .

The rest of the configurations, $\{\pi_2, \dots, \pi_k\}$, are then sampled, but they may not be sampled to the upper bounds. Note that all of the samples from π_i are collected before a sample is collected from π_{i+1} in Stage 2, for $i = 1, \dots, k - 1$. After we collect the r -th sample¹ from π_i , we calculate a criterion,

$$W_i(r) = \max \left\{ 0, \frac{\delta}{2r} (h^2 S_i^2 / \delta^2 - r) \right\}.$$

The sample mean, $\bar{X}_i^{(r)}$, of the r samples is then compared against \bar{X}' . Case (i): If $\bar{X}_i^{(r)} < \bar{X}' - W_i(r)$, we conclude that π_i is not the best configuration and stop sampling from π_i . In addition, if $\pi_i = \pi_k$, we return $\pi = \pi'$ as the best configuration; otherwise, we start sampling from π_{i+1} . Case (ii): If $\bar{X}_i^{(r)} > \bar{X}' + W_i(r)$, we expect that π_i is better than π' , and collect samples from π_i to the upper bound, N_i . If $\bar{X}_i^{(N_i)} > \bar{X}'$, we update $\pi' = \pi_i$ and $\bar{X}' = \bar{X}_i^{(N_i)}$ as we expected. If $\bar{X}_i^{(N_i)} \leq \bar{X}'$, we update neither π' nor \bar{X}' contrary to our expectation. In addition, if $\pi_i = \pi_k$, we return $\pi = \pi'$ as the best configuration; otherwise, we start sampling from π_{i+1} . Case (iii): If $\bar{X}' - W_i(r) \leq \bar{X}_i^{(r)} \leq \bar{X}' + W_i(r)$, then r samples are not sufficient to differentiate π_i and π' , and we continue sampling from π_i .

2.1.1 Criterion and confidence parameter

Figure 1 provides a further intuition behind the choice of the criterion, $W_i(r)$, and the confidence parameter, h . Observe that

$$[-r W_i(r), r W_i(r)]$$

for $0 \leq r \leq N_i$ defines a triangular area. If $r(\bar{X}_i^{(r)} - \bar{X}')$ exits the triangular area through the lower (respectively, upper) edge, then TSSD concludes (respectively, expects) that π_i is worse (respectively, better) than π' .

¹The r samples include the n_0 samples collected in Stage 1.

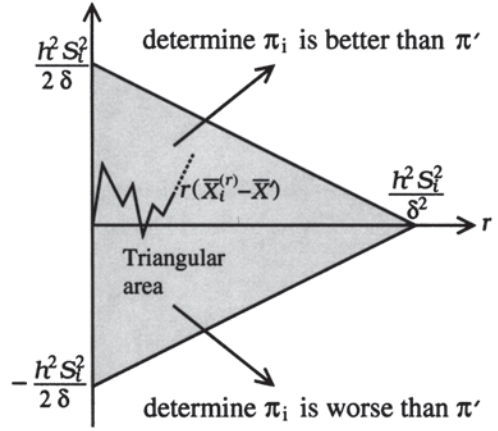


Figure 1: TSSD determines whether π_i is better or worse than π' when $r(\bar{X}_i^{(r)} - \bar{X}')$ exits the triangular area.

We choose the triangular area so that TSSD makes correct decisions with high probability, and yet the number of samples needed to make decisions is minimized. Observe that the confidence parameter h can be used to determine the size of the triangular area. A larger h implies a larger triangular area, which in turn makes TSSD more conservative in the sense that it collects stochastically more samples before concluding π_i is better or worse than π' . Since TSSD makes correct decisions with higher probability with a larger h but requires stochastically more samples with a larger h , we choose the confidence parameter as small as possible under the condition that Theorem 1 holds.

2.1.2 Algorithm for computing confidence parameter

We find the smallest h satisfying (1) by using a binary search, where the left hand side of (1), $f(h)$, is evaluated via Monte Carlo integration. If $f(\cdot)$ is monotonic, a binary search is guaranteed to find the smallest h satisfying (1). Unfortunately, a proof of the monotonicity of $f(\cdot)$ does not appear to be straightforward, even though our numerical experiments suggest that $f(h)$ is a decreasing function of h . Note, however, that the smallest h is merely preferable to a larger h , since the smallest h minimizes the number of samples needed in TSSD, but a larger h also guarantees Theorem 1. Our binary search is guaranteed to find an h satisfying (1), even if $f(\cdot)$ is not monotonic. Also, it is easy to show that upper and lower bounds of $f(h)$ are decreasing with h . Specifically, there exists a function, $g(h)$, such that $g(h) - 0.01 < f(h) < g(h) + 0.01$ and $g(h)$ is decreasing with h (see [6]).

2.2 Results

Now, we compare TSSD to the two-stage sequential-decision algorithm by Hong and Nelson (HN) [2] and the sequential-stage algorithm by Kim and Nelson (KN) [3]. Throughout, we fix $\alpha = 0.2$, $\delta = 1$, $n_0 = 20$, and $k = 20$. In this setting, the confidence parameter is $h = 3.78$. In all figures, we assume that the standard deviations are identical for all configurations, and the total number of samples or the number of configuration changes in each algorithm is plotted against the standard deviation σ , varying between 0 and 10. Since the number of samples and the number of configuration changes depend on particular samples, we take the average of 1,000 runs to generate each data point. The settings of the μ_i 's are specified in each figure.

In Figure 2, the means of the 20 candidates are chosen between 0.25 and 5 so that they are equally spaced. Specifically, the i -th

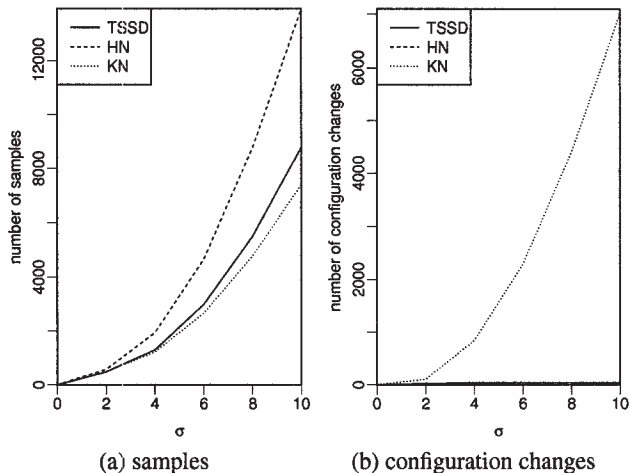


Figure 2: Comparison of (a) the number of samples and (b) the number of configuration changes in HN, KN, and TSSD. The means of the 20 configurations are equally spaced.

smallest mean is $i/4$ for $1 \leq i \leq 20$. Figure 2(a) plots the number of samples collected to find the best configuration as a function of the standard deviation, σ . When the variability is small ($\sigma < 1$), the samples collected in Stage 1 suffice to determine the best configuration, and the number of samples collected is $n_0 \times k = 400$ for all three algorithms. As the variability increases, the number of samples increases in all three algorithms. HN has the highest rate of increase, and requires as many as 60% more samples than TSSD at higher σ . KN has the lowest rate of increase, and requires 0-15% fewer samples than TSSD. However, KN changes the configurations to be simulated 180 times more frequently than TSSD (see Figure 2(b)).

In Figure 3, the means of the 20 candidates are chosen such that $\mu_b = \mu_i + \delta, \forall \pi_i \neq \pi_b$. This setting of μ_i 's is the worst case in the sense that the algorithms are most likely to select a configuration in \mathcal{W} . Note that π_b is the only nearly best configuration (i.e. $\mathcal{B} = \{\pi_b\}$), and that the mean of every configuration in \mathcal{W} is exactly δ separated from μ_b . Figure 3(a) plots the number of samples collected to find the best configuration as a function of σ . In a way similar to Figure 2(a), the number of samples increases in all three algorithms, as the variability of the performance increases. HN collects as many as 60% more samples than TSSD at higher σ . On the other hand, KN collects 0-20% fewer samples than TSSD (see Figure 2(b)), but change configurations 260 times more frequently than TSSD (see Figure 3(b)). Thus, both KN's advantage over TSSD with respect to the number of samples and TSSD's advantage over KN with respect to the number of configuration changes increase in the setting of worst case means.

Overall, Figures 2 and 3 suggest that TSSD achieves a good tradeoff between the number of samples and the number of configuration changes. Specifically, TSSD collects far fewer samples than HN and changes the system configurations to be simulated just as frequently as HN. Also, TSSD changes the system configurations to be simulated by orders of magnitude less frequently than KN and collects only slightly more samples than KN.

3. CONCLUDING REMARKS

TSSD is complementary to the work in [7], which appears in the main SIGMETRICS/Performance Conference. While TSSD is suitable for a relatively small number of candidate configurations,

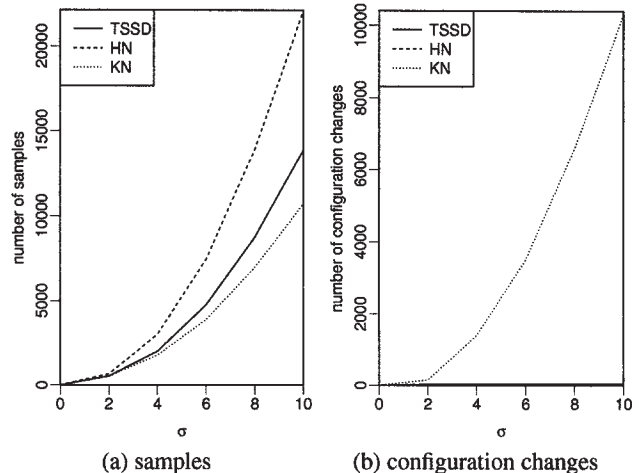


Figure 3: Comparison of (a) the number of samples and (b) the number of configuration changes in HN, KN, and TSSD. The means of the 20 candidates are chosen such that $\mu_b = \mu_i + \delta, \forall \pi_i \neq \pi_b$.

[7] provides a local search algorithm to deal with a larger number of configurations. TSSD and ideas in TSSD can be used to improve the speed of the local search algorithm in [7].

Issues discussed at the MAMA workshop include the cost of checkpoints when TSSD is used to optimize the configuration of a system where the performance is evaluated by measurements instead of computer simulations. Our current work is to extend TSSD so that it does not require checkpoints [8].

4. REFERENCES

- [1] E. J. Dudewicz and S. R. Dalal. Allocation of observations in ranking and selection with unequal variances. *Sankhya*, B37(1):28–78, 1975.
- [2] L. J. Hong and B. L. Nelson. The tradeoff between sampling and switching: New sequential procedures for indifference-zone selection. *IIE Transactions*, 37(7):623–634, 2005.
- [3] 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.
- [4] A. M. Law and W. D. Kelton. *Simulation Modeling and Analysis*. McGraw-Hill, third edition, 2000.
- [5] 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.
- [6] T. Osogami. Finding probably best systems quickly via simulations. Technical Report RT0684, IBM Tokyo Research Laboratory, 2006.
- [7] T. Osogami and T. Itoko. Finding probably better system configurations quickly. In *Proceedings of the ACM SIGMETRICS / IFIP PERFORMANCE 2006*, pages 264–275, June 2006.
- [8] T. Osogami and S. Kato. Optimizing system configurations quickly by guessing at the performance. Manuscript, 2006.
- [9] Y. Rinott. On two-stage selection procedures and related probability-inequalities. *Communications in Statistics — Theory and Methods*, A7(8):799–811, 1978.