PARALLEL STATISTICAL SELECTION IN MULTIPROCESSORS

Arthur Ieumwananonthachai and Benjamin W. Wah
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
1101 West Springfield Avenue
Urbana, Illinois 61801
wah@aquinas.csl.uiuc.edu

ABSTRACT

In this paper, we develop parallel multistage selection strategies for guiding the concurrent evaluation of candidates from a large pool of candidates, with the objective of finding the candidate with the maximum (resp. minimum) value in a limited amount of time. We present a statistical model for finding the appropriate number of candidates to be sampled and when each should be sampled. We verify the analytical results by simulations and evaluate the performance of strategies that are analytically intractable. Our performance results show that there is linear speedup in parallel processing over selection done sequentially. Results obtained from application of our selection method to learn heuristics for a real-world application demonstrate that heuristics with improved performance can be found by a systematic search of the space of feasible heuristics in limited time. Due to space limitation, we are unable to present the details of the experiments and the exact heuristics found.

1. INTRODUCTION

Given a pool of candidate populations, the problem of finding the one with the maximum (or the minimum) average value through a sequence of tests is known as the *selection problem.* We are interested in finding the best one from a large pool of candidates, assuming that the pool is too large to be sampled fully within the given time constraint. This problem has not been studied in statistics and occurs naturally in the design and performance evaluation of heuristics.

The parallel processing environment targeted for this study consists of m processor clusters connected by an interconnection network. Each processor cluster has q homogeneous processors and a shared memory. Processors in the same cluster share information through the shared memory, while processors in different clusters exchange information through the network.

We have previously developed a generate-and-test method for scheduling the evaluation of heuristics under time constraints [3-5]. We have also studied a statistical model for trading between the number of candidates tested and the amount of evaluation performed on each in the sequential case. In this paper, we extend these results summarized in Section 2 to the parallel case.

This research was supported in part by National Aeronautics and Space Administration Grants NCC 2-481 and NGT 50743 (NASA Graduate Fellowship Program) and in part by National Science Foundation Grant MIP 88-10584.

Proceedings of International Conference on Parallel Processing, August 1992.

2. PREVIOUS WORK

Existing learning methods rely on domain knowledge to efficiently generate good heuristics by modifying the incumbent heuristics. For knowledge-lean applications where these methods are not feasible, the alternative is to maintain a pool of competing heuristics and to find the best one in the pool. Resource scheduling algorithms that identify promising heuristics for testing are, therefore, an important element of such a system. We have developed TEACHER (or TEchniques for Automated Creation of HEuRistics), a general framework for generating, selecting, and testing heuristics in knowledge-lean application domains under resource constraints. The sequential version of TEACHER has been used in learning new heuristics for several applications.

To tackle the resource scheduling problem, we have developed an efficient sequential selection strategy. Its solution entails finding the best guidance function that identifies the population to test next. Work in this area was pioneered by Bechhofer [2]. Recent work in statistics [1] deals with finding the stopping criteria for testing a finite number of populations.

When resources are limited and there are too many candidates to be evaluated in the available time, the selection strategy must trade rationally between the number of candidates to be tested and the confidence of the sample-mean values. To perform this tradeoff, we formulate the general *multistage selection strategy*, $G(T)$, as a series of stages, $G_i(g_i,t_i,p,n_i)$, where i ranges from 1 to m (see Figure 2.1). Each G_i is characterized by a quadruplet consisting of a) g_i, the guidance function used in the stage, b) t_i, the duration of the stage, c) p, the number of parallel processors (equals 1 in the sequential case), and d) n_i, the number of populations to be considered for testing in this stage [3, 5].

Within this multistage procedure, the first stage corresponds to coarse initial testing to weed out unworthy candidates, followed by a more careful evaluation of the better candidates. Only candidates with the top n_{i+1} sample-mean values at the end of stage i are used in stage $i+1$ for further

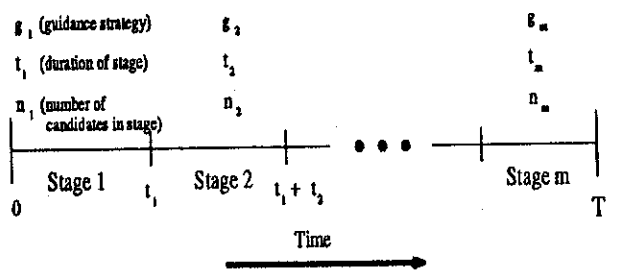


Figure 2.1. Sequential Multistage Testing Procedure.

testing.

The performance of the multistage-selection method depends on the values of m and the four parameters of each stage. These parameters must be set appropriately based on the problem characteristics. Factors affecting the setting of these parameters include the size and distribution of each population, the total amount of testing time, and the number and distributions of populations. We have analyzed the performance of sequential multistage selection strategies in our previous work [4, 5].

Next we extend sequential resource scheduling strategies to the parallel environment. In this case, resources include both time and processors.

3. PARALLEL MULTISTAGE STATISTICAL SELECTION STRATEGIES

We study both synchronous strategies, where the sampling overhead is unity for all candidates, and asynchronous ones, where the sampling overhead is stochastic. For the latter, we assume that the overheads of sampling a population are i.i.d., drawn from a known distribution. In our analysis, we assume that each population has a normal distribution, that the variances of all populations are equal, and that samples within a population are i.i.d. We further assume that the distribution of population means is known, and that the values of population means are i.i.d. The above assumptions have been verified experimentally for a number of applications. Our analysis also assumes that the population variance is known.

The parallel multistage selection strategies studied in this paper include single-stage round-robin, two-stage round-robin, two-stage round-robin/greedy, and two-stage round-robin /minimum-risk [3].

3.1. Static Guidance Strategies

Since the sequence of populations to be sampled is fixed ahead of time in static strategies, there is no difference in performance between a parallel strategy with T time units and a sequential strategy with $T \cdot p$ time units [5].

For example, the performance of a *single-stage round-robin strategy*, $G(RR, T, p, n)$, when population means have a normal distribution $N(\mu_0, \sigma_0^2)$, can be expressed as

$$E[\mu_{select}] = \mu_0 + \int_{x=-\infty}^{+\infty} \frac{\sigma_0^2 x n e^{-\frac{x^2}{2}} [\Phi(x)]^{n-1}}{\sqrt{2\pi(\sigma^2/s + \sigma_0^2)}} dx \qquad (3.1)$$

where $\Phi(x)$ is the c.d.f. for a $N(0,1)$ distribution, μ_{select} is the population mean of the selected population, σ^2 is the population variance, n is the number of populations tested, and s is the number of samples drawn from each population. For the synchronous case, $s = T \cdot p / n$. For the asynchronous case, the performance can be approximated by using the average number of samples drawn in a stage and $\bar{c}$, the average time for each sample drawn. In this case, $\bar{s} = (T \cdot p) / (n \cdot \bar{c})$ [4].

Since the performance of static strategies with duration T and p processors is the same as that of sequential static strategies with duration $T \cdot p$, the optimal parameters for sequential static strategies, *i.e.*, the optimal duration and the optimal number of candidates to be selected in each stage [4, 5], applies to the parallel case as well.

3.2. Dynamic Guidance Strategies

Dynamic strategies can outperform static ones because they use dynamic information in selecting the population to be sampled. However, their analysis is complex because selection is based on the dynamic ordering of candidates in the pool, which in turn depends on the sequence of past decisions. The analysis of dynamic strategies requires the joint distribution of the performance values of all populations based on their performance ranking. Since the performance distribution changes after each sample is drawn, the joint distribution must be updated after each pick.

Evaluation of parallel dynamic selection strategies is more complex than that of sequential ones because the former may evaluate multiple candidates simultaneously. The evaluation also depends on whether the architecture has a shared memory or is distributed, since the candidates selected depend on whether the processor has access to all candidates in the system.

The simplest dynamic strategy is the *two-stage round-robin/greedy strategy* that selects for testing in the second stage populations with the smallest (resp. largest) sample-mean value. The performance of each population can be characterized by its sample mean ($\bar{x}$), population mean (μ), and the number of tests performed so far (s). We need to find the joint distribution of these values for all populations ordered by their sample mean values. The distributed-memory case, where each processor can access a unique and private set of candidates, is the easiest to analyze. However, even for this simple case, the resulting equation is still far too complex to be evaluated in a closed form.

When several processors can share the same partition of candidates, the derivation is more complex because there are many more permutations of possible changes in the order of performance after each pick. If n is the number of candidates that q processors in a cluster can access ($q \le n$), then the number of permutations is equal to ${}^{n}P_{q}$ Each permutation corresponds to a term in the final expression and involves q

integrations.

For the *two-stage round-robin/minimum-risk strategy* [3], the statistical measure used for ordering populations is far more complex than the sample-mean values used by the greedy strategy. For this reason, we do not analyze this strategy.

Monte Carlo simulations are used to evaluate the dynamic strategies. The parameters for dynamic two-stage round-robin/greedy and round-robin/minimum risk strategies are set heuristically using the values derived for the corresponding static cases. We found minor degradation in performance due to this approximation [5].

3.3. Dynamic Guidance Strategies with Redistribution

We consider the effect of redistributing the populations dynamically using the two-stage round-robin/greedy strategy. We assume a distributed network of p processors connected by a linear ring, and samples are drawn synchronously. In the second stage where the greedy strategy is used, each processor sends a fixed number of its top populations to its immediate neighbor after making a pick. Each processor then sorts the new set of populations before making further picks.

The analysis of this strategy requires the joint distribution of all populations in the system, because redistribution causes dependencies among candidates in different processors. This additional complication renders the already complex equations for the greedy strategy unusable. To approximate the effect of redistribution on performance, we ignore the dependencies among candidates after redistribution. This assumption is justified when the number of candidates exchanged is small.

Let n be the number of candidates in each processor, and k be the number of candidates exchanged after each pick. Let $F_i(x)$ be the c.d.f. of the i^{th}-order population. The c.d.f. of the first-order population after exchange is

$$F_n^{(1)}(x) = F_n(x)\,F_{n-k}(x). \qquad (3.2)$$

This implies that the distribution of sample means of the top k populations will increase after each exchange, and that the average sample means of the bottom $n-k$ candidates will decrease. However, the rates of change will gradually diminish as good candidates are redistributed more evenly in successive exchange operations. The maximum number of exchanges needed to completely redistribute the populations is $p-1$ [6]. After $p-1$ exchanges, the performance of selection in a distributed network is identical to that of a shared-memory system.

We perform Monte Carlo simulations for $k=1$. Results shown in Figure 3.1 confirm that successive exchanges have diminishing returns, and that exchanges lead to minor improvement in performance in a two-stage selection algorithm.

A closer look at the sampling process reveals that the pool of populations and the statistics of each population change very slowly because the value of each pick contributes only a small fraction to the overall sample mean of a population. Another observation is that the time for each pick is large relative to the communication overhead. Hence, a more efficient way of maintaining global information is for each processor to dupli-

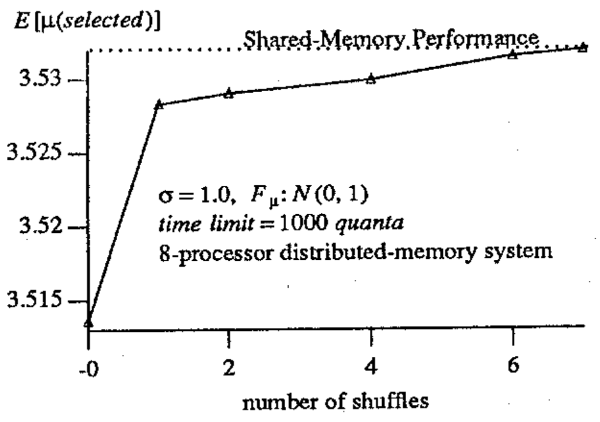


Figure 3.1. Simulation results showing effects of redistribution on performance of 2-stage RR/Greedy strategy.

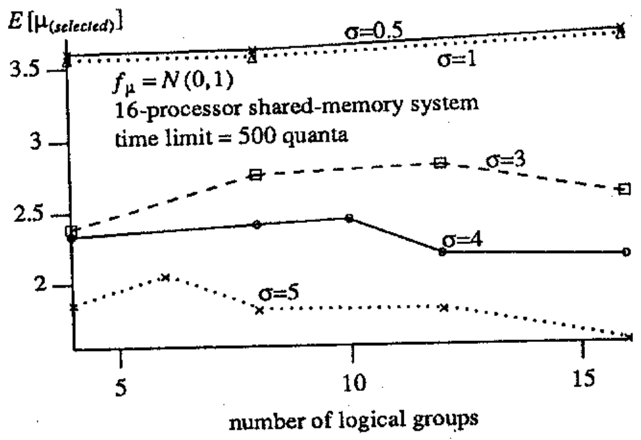
cate the populations used in other processors, and to broadcast to all processors a) the candidate it has chosen for evaluation before it starts testing, and b) the resulting performance after testing is done. As a result, all processors can maintain global information without extensive exchanges of populations. This enhancement allows parallel selection on a distributed-memory system to behave like that on a shared-memory system with negligible overhead.

3.4. Dynamic Guidance Strategies with Concurrent Sampling

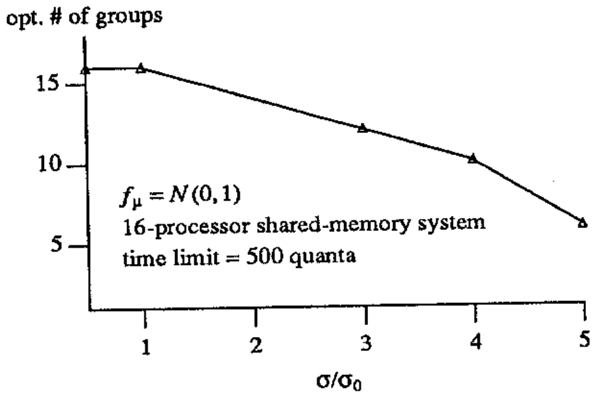
Given a fixed number of quanta, there is a tradeoff between the number of candidates that can be evaluated and the amount of evaluation that can be performed on each. It may be desirable in certain cases to evaluate the same candidate on multiple processors simultaneously. Concurrent sampling of the same candidate leads to more accurate evaluation of the chosen candidate but an increasing probability of ignoring other good candidates. This tradeoff is especially important for the greedy strategy which tends to always select the same candidate for testing.

The performance due to concurrent sampling depends on the ratio between σ, the standard deviation of each population, and σ_0, the standard deviation of the mean of all populations. As σ / σ_0 increases, it becomes harder to eliminate poor candidates; hence, the system should focus on a smaller number of candidates, while getting more samples from each concurrently. We define a logical group as a collection of processors that sample the same candidate concurrently, assuming that samples are drawn synchronously. Logical groups do not have to coincide with processor clusters, although the two are highly related.

Figure 3.2a depicts the performance of the two-stage round-robin/greedy strategy with respect to the number of logical groups and different population standard deviation σ. There exists an optimal number of logical groups for each σ. Moreover, more processors should be scheduled to sample the

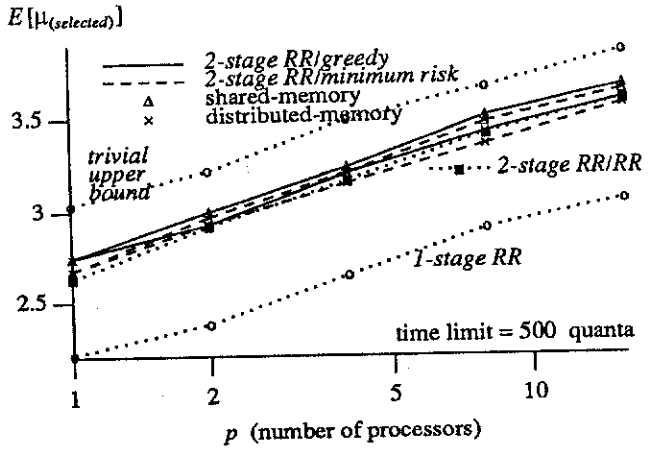


(a) Performance versus number of logical groups ($\sigma_0 = 1$).

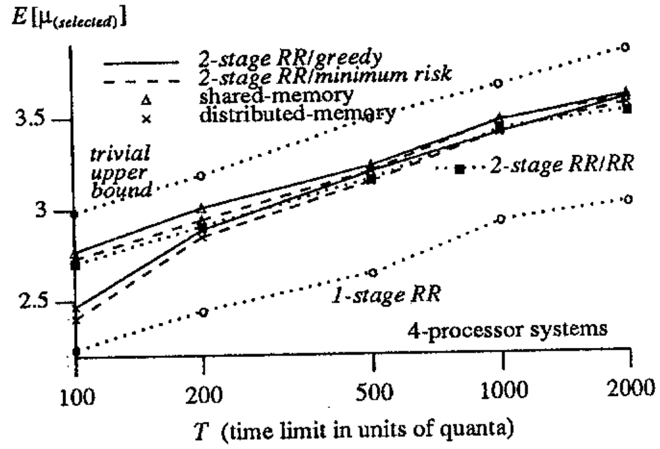


(b) Optimal number of logical groups versus ratio of population variances ($\sigma_0 = 1$).

Figure 3.2. Effect of logical grouping on performance of 2-stage RR/Greedy strategy.



(a) Performance improvement with respect to degree of parallel processing for time limit of 500 quanta.



(b) Performance improvement with respect to time limit for 4-processor systems.

Figure 3.3. Performance of parallel multistage selection strategies for $f_\mu = N(0,1)$ and $\sigma = 1$.

same population concurrently as σ/σ_0 increases. This is shown in Figure 3.2b which shows the optimal number of logical groups with respect to the ratio of standard deviations.

3.5. Guidance Strategies under Asynchronous Sampling

In the asynchronous case where sampling overhead is stochastic and the time limit is fixed, there are variations on the number of samples drawn from each candidate in different runs. An accurate analysis would require the knowledge of the distribution of the number of samples drawn from each population in a stage of the selection process, a term that is a high-degree convolution of the distribution of sampling overheads. The analysis is even more complex when the value of the sample drawn and its corresponding overhead are correlated, which happens frequently since better heuristics tend to take more time. Due to these reasons, we do not attempt to analyze exactly the performance of asynchronous strategies [4].

To find the approximate performance, we can use the average number of samples drawn in each stage instead of the exact distribution. This is equal to the total amount of processing time available ($T \cdot p$) divided by the average sampling overhead, $\bar{c}$. The performance of asynchronous multistage selection strategies with T time units and p processors is then approximated by the performance of synchronous multistage selection strategies with $T/\bar{c}$ time units and p processors.

3.6. Simulations Results

Figure 3.3 shows the average performance, $E[\mu_{(selected)}]$, of the simulation results for various one-stage and two-stage strategies. Figure 3.3a shows the improvement in performance of selection against the degree of parallel processing for a fixed time limit. Figure 3.3b shows the performance improvement

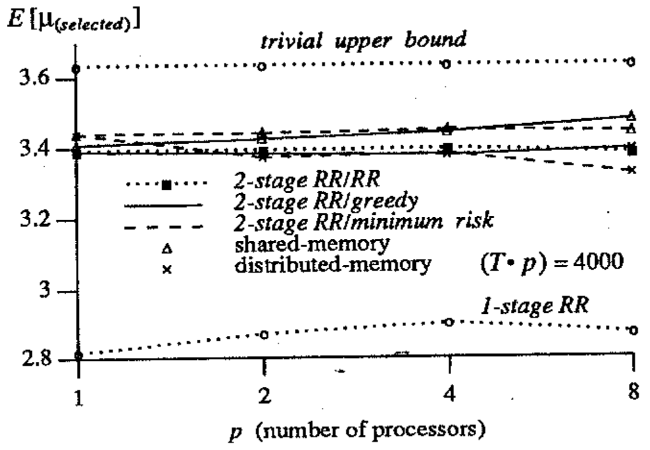


Figure 3.4. Performance of parallel multistage selection strategies for $f_\mu = N(0,1)$, $\sigma = 1$, and constant processor-time product.

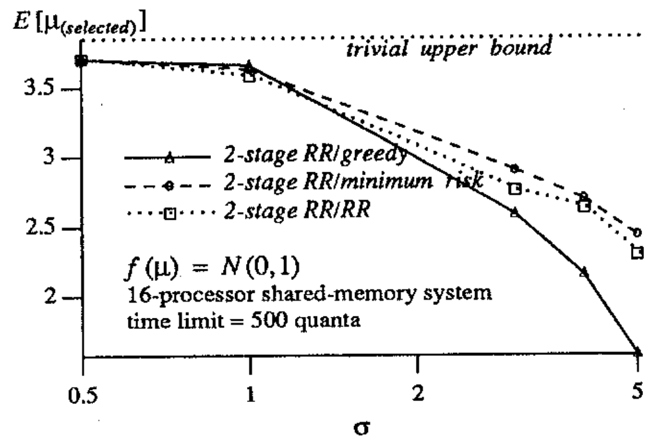


Figure 3.5. Improved performance of 2-stage round-robin/minimum-risk strategy over other strategies when σ is large.

under different time limits when the number of processors is fixed. We see that two-stage strategies outperform the single-stage ones by a significant margin. Our results also confirms the results of Section 3.3, which show that dynamic strategies on a shared-memory system outperform those on a distributed-memory system with private pools.

Figure 3.3 shows that the performance of all strategies tends to increase in a log-linear fashion with both the number of processors and the time limit. Hence, it seems likely that the performance of multistage selection strategies is a function of $T \cdot p$, the number of tests that can be performed. To verify this hypothesis, we plot the performance of multistage strategies for different number of processors while keeping $T \cdot p$ fixed. The results, shown in Figure 3.4, corroborate our hypothesis for most strategies. Two strategies that violate our hypothesis are the two-stage round-robin/minimum-risk strategy on a distributed-memory system and the two-stage round-robin/greedy strategy on a shared-memory system. In the first case, the amount of information available to each processor decreases as the number of processors increases, while the total number of samples drawn is fixed. This causes the candidates to be selected using less and less accurate performance predictor. Variations in performance in the second case are due to a tradeoff between the number of candidates examined and the amount of tests performed on the top candidates, a phenomenon we discussed in Section 3.4 already.

While the two-stage round-robin/greedy strategy outperforms the two-stage round-robin/minimum-risk strategy when the population variance is small, the minimum-risk strategy has the ability to discern between good and poor candidates when the variance of populations is large. This phenomenon is demonstrated in Figure 3.5.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the derivation of the initial theoretical results on the sequential case by A. Aizawa and S. Schwartz [3, 5].

REFERENCES

[1] *Design of Experiments: Ranking and Selection*, Marcel Dekker, Inc., New York, NY, 1984.

[2] R. E. Bechhofer, "A Single-Sample Multiple Decision Procedure for Ranking Means of Normal Populations with Known Variances," *Ann. Math. Statist.*, vol. 25, no. 1, pp. 16-39, Institute of Mathematical Statistics, Ann Arbor, MI, March 1954.

[3] A. Ieumwananonthachai, A. N. Aizawa, S. R. Schwartz, B. W. Wah, and J. C. Yan, "Intelligent Mapping of Communicating Processes in Distributed Computing Systems," *Proc. Supercomputing 91*, pp. 512-521, ACM/IEEE, Albuquerque, NM, Nov. 1991.

[4] A. Ieumwananonthachai and B. W. Wah, "Learning Process Mapping Heuristics under Stochastic Sampling Overheads," *Proc. Computing in Aerospace 8 Conference*, American Institute of Aeronautics and Astronautics, Baltimore, MD, Oct. 1991.

[5] A. Ieumwananonthachai, A. Aizawa, S. R. Schwartz, B. W. Wah, and J. C. Yan, "Intelligent Process Mapping Through Systematic Improvement of Heuristics," *J. of Parallel and Distributed Computing*, vol. 13, Academic Press, July 1992.

[6] B. W. Wah and Y. W. Ma, "MANIP--A Multicomputer Architecture for Solving Combinatorial Extremum Problems," *Trans. on Computers*, vol. C-33, no. 5, pp. 377-390, IEEE, May 1984.