AIAA-91-3757-CP

LEARNING PROCESS MAPPING HEURISTICS UNDER STOCHASTIC SAMPLING OVERHEADS

Arthur Ieumwananonthachai and Benjamin W. Wah

ABSTRACT

The problem of optimal process mapping is NP-hard and involves the optimal placement of processes on the distributed system and the optimal routing of messages from one computer to another. The heuristics for solving this problem are often ad hoc, and are guided by intuition and experience of the designers. We have developed previously a statistical method for improving process mapping heuristics. The method explores systematically the space of possible heuristics under a specified time constraint. Its goal is to get the best possible heuristics while trading between the solution quality of the process mapping heuristics and their execution time. In this paper, we extend this statistical selection method to take into consideration the variations in the amount of time used to evaluate heuristics on a problem instance. We present the improvement in performance using this more realistic assumption along with some methods that alleviate the additional complexity.

KEYWORDS AND PHRASES. Distributed computing system, generate-and-test, heuristics, loosely coupled computers, process mapping, sequential selection, time constraint.

1. INTRODUCTION

Process mapping is important in resources scheduling in a distributed computing environment. However, its exponential complexity forbids the search of optimal solutions in many practical cases. As a result, most solutions to the process mapping problem are based on heuristic methods. An example is the post-game analysis system developed by Yan and Lundstrom [5,6]. These heuristics are developed in an ad hoc fashion and are based on the experience of the designers. Since the space of possible heuristics is very large, it is likely that there are heuristics that perform better than the ones selected by the designer.

We have developed previously TEACHER 4.1 (*TEchniques for Automatic Creation of HEuRistics*) [3], a system that extends the post-game analysis system through systematic and automatic exploration of its space of possible heuristics. The system attempts to find the best heuristics while trading between the quality of the solution found and its execution time. The results show improvement in performance over the original heuristics used.

As there is little knowledge available for generating good mapping heuristics, TEACHER 4.1 focuses on efficient scheduling of time in evaluating alternative heuristics. The scheduling algorithm is based on a statistical model we developed earlier in TEACHER 4.0 for trading off between the number of new heuristics to be generated and the amount of tests to be performed on each. This statistical method has been applied to find good parameters for depth perception in stereo vision [4].

The statistical method we developed earlier assumes that the sampling overhead, *i.e.*, the amount of time to test a candidate heuristics on a test case, is the same for all candidates. This is a crude approximation in in many problem domains, including the process mapping problem. Moreover, the performance value of the candidate may also depend on its sampling overhead. We present in this paper an enhanced model that addresses this problem.

2. AUTOMATED SYSTEM FOR SELECTING HEURISTICS

In this section, we present an overview of TEACHER 4.1 [3], a system for automating the selection of process mapping heuristics under resource constraints. The goal of the system is to explore systematically the space of possible heuristics and find the heuristics that provide the best tradeoff between the quality of the solution and the execution time. First, we discuss an overview of the post-game analysis system, which we have chosen as the heuristic method for solving the process mapping problem. We then present the objective of our selection prob-

A. Ieumwananonthachai and B. W. Wah (contact author: wah@aquinas.csl.uiuc.edu) are with Center for Reliable and High-Performance Computing, Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, 1101 West Springfield Avenue, Urbana, IL 61801.

This research was supported in part by National Aeronautics and Space Administration Grant NCC 2-481 and in part by National Science Foundation Grant MIP 88-10584.

Proceedings of Computing in Aerospace 8 Conference, American Institute of Aeronautics and Astronautics, Baltimore, MD, October 1991.

lem, *i.e.*, the type of heuristics we are seeking. Next, we present the generate-and-test paradigm in TEACHER. Last, we present the statistical selection method for scheduling resources.

2.1. Post-Game Analysis

The objective of the process mapping problem is to find a mapping of a set of communicating processes on a distributed computing system so that the completion time of the processes is minimized. Post-game analysis system is an efficient heuristic method for solving this problem. The system iteratively refines the mapping of the processes based on information collected during the execution using the previous mapping. The system starts off by executing (or simulating) the processes using an initial random mapping. Based on information gathered during the execution, heuristics are applied to propose changes to this mapping, with the goal of reducing the completion time. The set of processes are then executed using the new mapping. These steps are repeated until no further change to the mapping can be found.

The goal of the post-game system is more complex than simply minimizing the completion time of processes. It trades between the amount of time (or cost) it uses to find the mapping and the quality of the mapping found (*i.e.*, the completion time of the processes based on this mapping). This is true because the cases that one set of heuristics takes unlimited time to find the optimal mapping and one that takes zero time to find a trivial mapping cannot be considered as satisfactory. Unfortunately, there is no concrete way of defining the optimal level of tradeoff between the overhead and the solution quality. In the next section we define a parametrized objective function that encompasses all possible levels of tradeoff.

2.2. Objective of Learning Process

The objective of the learning process is to improve the performance of the target problem solver, which is the post-game system in this case. However, as discussed in the previous section, the objective of the post-game system is not well defined. Consequently, the objective of the learning process is ill-defined as well. Instead of coming up with a single objective function for the learning process, we define a family of objectives that represent various desirable levels of tradeoff between the execution time (or cost) and the quality of the mapping found. The learning system will find the heuristics that perform the best for each objective drawn from this family. The users can then select the objective and the associated heuristics that suit their application requirements the best.

In order to create a parametrized objective function, we define the objective as a function of $t_{\max}$, a parameter that represents the maximum time that a set of heuristics are allowed to execute before a penalty is imposed.

We first define $c(h,v,t_{\max})$ as a piecewise continuous function with a discontinuity at $t_{\max}$, where h is the candidate heuristics, v is the process mapping problem, and $c(h,v,t_{\max})$ is the cost of the mapping for given h and v. Let $t(h,v)$ be the time to find the mapping.

$$c(h,v,t_{\max}) = \begin{cases} 1 & t(h,v) \le t_{\max} \\ c_r\,[t(h,v) - t_{\max}] + 1 & t(h,v) > t_{\max} \end{cases} \tag{2.1}$$

where c_r is a constant that defines the relationship between cost and overhead when $t_{\max}$ is exceeded. The reason for choosing $t_{\max}$ is to avoid the degenerate case in which $c(h,v,t_{\max})$ and $t(h,v)$ are both zeroes. The values of both $c(h,v,t_{\max})$ and $t(h,v)$ are averaged over a set of initial random mappings.

We define $q(h,v)$, the quality of h for problem v, as the reciprocal of the completion time of the processes mapped on the distributed system averaged over a set of initial random mappings using candidate heuristics h. That is, the quality is higher when the processes mapped complete sooner.

The objective of the learning process, $Q(t_{\max})$, is to find a candidate heuristics that maximizes the average quality-cost ratio for a given $t_{\max}$; that is,

$$Q(t_{\max}) = \max_h \sum_v \frac{q(h,v)}{c(h,v,t_{\max})} \tag{2.2}$$

2.3. Generate-and-Test Framework

For a given objective drawn from the parametrized objective function, the generate-and-test system finds the heuristics that maximize this objective. As there is little domain knowledge in the process mapping problem to guide the generation of good heuristics, and the space of good heuristics is very large, an efficient resource scheduling algorithm that focuses on finding good heuristics in a limited amount of time is very important.

There are three main components in the generate-and-test framework as depicted in Figure 2.1: the *candidate generator* generates new candidates to be considered, the *candidate evaluator* applies the selected candidate to the target problem and records its performance, and the *scheduler* decides on the best way to use the resources.

A candidate is a point within the search space, which in this case is a set of heuristics for the post-game analysis system. In order for the learning system to explore the

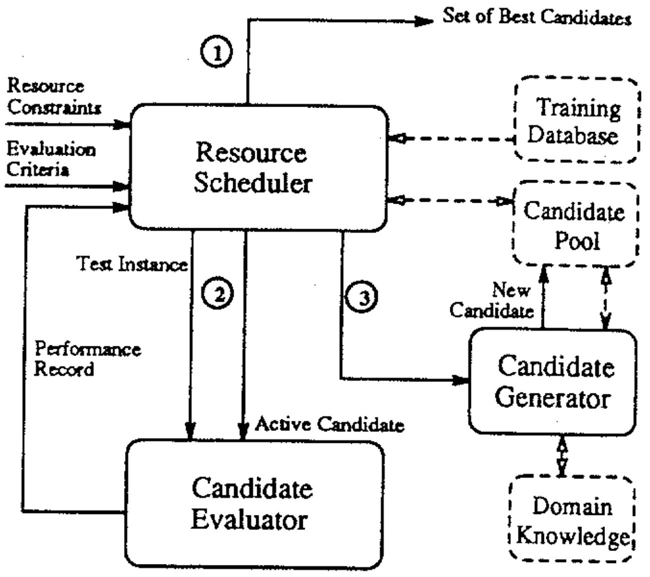


Figure 2.1. Generate-and-Test Framework.

space of possible post-game heuristics, the rules used in post-game analysis must be modifiable. To simplify the modification, the post-game system is changed so that the rules are represented as frames. Each candidate for the generate-and-test framework is then represented by a set of frames.

The evaluation process is divided into small tests called *quanta*. This is needed in order to avoid spending a large amount of time on a poor candidate. Additional tests are performed only on candidates that demonstrate some merits during the previous quanta. During a quantum of time, the candidate evaluator perform tests on the candidate selected using test cases randomly generated or supplied by the users.

In our implementation, the performance of each candidate is found by applying it to selected problem instances from the training database. A *problem instance* (or test case) is represented by the specification of the set of communicating processes to be mapped, the hardware on which the processes are mapped, and the input values to the distributed processes. The results from each evaluation of a candidate heuristics for a problem instance with an initial mapping are recorded and are used to compute its performance.

Within each quantum, the candidate evaluator evaluates the performance of the selected candidate on a single problem instance randomly drawn from a training database. As the performance of each candidate heuristics evaluated on the same problem with different initial mappings can vary widely, it is necessary to test each problem instance enough in order to get a good confidence on the value of its average performance. In our case, the candidate evaluator tests each test case until the 95% confidence interval (based on Student's t-distribution) of the average performance across the different initial mappings are within 5% of the average value.

At the end of a quantum, the scheduler decides on one of three following actions: (1) select the next candidate to evaluate from the candidate pool, (2) generate a new candidate to be placed in the candidate pool and possibly remove an existing one from the pool, and (3) select a set of best candidates and stop learning when time is expended. The decision are made based on the resource scheduling algorithm and the performance of existing candidates in the pool. In our case the scheduling algorithm is based on a statistical model to be presented in the next section.

If choice (2) is selected, then the candidate generator is used to generate a new candidate. Our current candidate generator is relatively primitive: a new candidate is generated by applying a sequence of operators that transform an existing candidate into a different candidate.

The strategy used in our current candidate generator is divided into 3 stages. In the first stage, the generator locates an existing candidate that performs the best in a region of heuristics space. In the second stage, it finds a sequence of operators that cause the greatest improvement in performance within that region of the heuristics space. In the last stage, a new candidate is generated from the candidate selected in the first stage using the sequence of operators found in the second stage. Our current implementation is based on a rule-based system, so additional domain knowledge, operators, and rules can be added easily.

2.4. Statistical Candidate-Selection Strategy under Time Constraints

The goal of the scheduler in the generate-and-test is to choose the best candidate from a pool of candidates, each of which has an associated set of performance values. In statistical term this can be restated as, given a set of populations consisting of normally distributed random numbers (with unknown means and variances), the problem is to select the one with the highest population mean by testing a certain number of samples from these populations.[1] In our case, the populations are candidate heuristics, and the numbers comprising the elements of the

1. Population mean and variance are properties of a population. They can be estimated by the sample mean and variance if limited samples are drawn from the population when the population is infinite in size.

populations are the performance values associated with applying the heuristics on the given problem instances. Making one pick from a population is analogous to testing the candidate on one problem instance. The goal is to choose the candidate with the highest mean within a given amount of tests (or picks). Since resources are limited and there are too many candidates to be tested in the available time, the selection strategy must make a succinct choice in trading between the number of candidate to be tested and the accuracy of the sample-mean values.

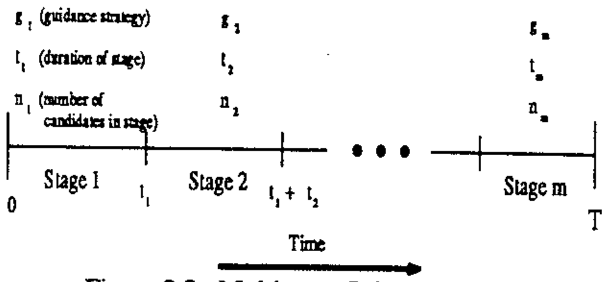
Existing methods can be classified into *static* and *dynamic*. Static strategies, such as a *round-robin* strategy, have a selection sequence that is fixed ahead of time independent of the values of the picks observed during the selection process. Dynamic strategies, on the other hand, select the candidate for testing based on previous sample values. A *greedy* method that selects the candidate with the highest sample mean for further testing is such an example. Dynamic strategies have an advantage over static ones because they less likely spend time on candidates that are obviously inferior after some initial tests. However, finding good dynamic strategies is usually harder.

Early and current work in this area was pioneered by Bechhofer [1]. However, the solutions found are not applicable directly because research in statistics deals with a finite number of populations and unlimited time.

2.4.1. Multistage Selection Strategy

Our general selection strategy, $G(T)$, is formulated as a series of stages, $G_i(g_i, t_i, n_i)$, where i ranges from 1 to the number of stages (see Figure 2.2). Each stage is charterized by a triplet consisting of (a) g_i, the guidance strategy used in the stage, (b) t_i, the duration of the stage, and (c) n_i, the number of candidates to be considered for testing in this stage. Our multistage selection strategy can accommodate both static and dynamic guidance strategies. In the 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 the candidates that have the top n_{i+1} sample mean values at the end of stage i are used in stage $i+1$ for further testing.

The performance of the multistage selection method depends on the number of stages used, the guidance strategy used in each stage, and the number of populations tested in each stage. Factors that affect the performance include the size and distribution of each population, the total amount of testing time, and the number of possible candidates and their distributions.



g_1 (guidance strategy)
t_1 (duration of stage)
n_1 (number of candidates in stage)
g_2
t_2
n_2
g_m
t_m
n_m
Stage 1
Stage 2
Stage m
0
t_1
$t_1 + t_2$
T
Time

Figure 2.2. Multistage Selection Procedure.

Currently we are only interested in single-stage and two-stage strategies. The strategies we have studied include single-stage round-robin, two-stage round-robin, two-stage round-robin/greedy, and two-stage round-robin/minimum-risk. The *Minimum-risk* strategy [3] selects the candidate that will minimize the risk of error in the estimation of the best mean value after the next pick. It takes uncertainty in the sample mean value into consideration and performs slightly better than the greedy method.

3. PERFORMANCE EVALUATION OF MULTISTAGE SELECTION

In order for the multistage selection strategies to be successful, we must develop some systematic methods for determining the values of their parameters that are most suitable for the give problem. To accomplish this, we need to analyze the performance of the selection strategies. Previously, we have analyzed the case in which all *sampling overheads* (*i.e.* the time for testing a candidate) are unity [4]. Section 3.1 summarizes the result of this analysis. We extend the problem in Section 3.2 to include the case in which the sampling overhead is stochastic. The analysis of the new upper bounds on the performance of selection strategies is presented next. This is followed by a description of the effect of the new assumptions on the performance of the multistage strategies. Finally, we present an updated method for applying the multistage selection strategy at run-time.

3.1. Previous Work

In our previous analysis [4], we have derived the upper bound on the performance of multistage strategies, analyzed the performance for single-stage round-robin and a few two-stage selection methods, and developed the method for controlling the multistage selection at run time. We summarize these results in this section.

There are four sets of assumptions that we made in the analysis. The first set of assumptions are that each population (*i.e.*, candidate performance) is normally distributed, that the variances of all populations are equal, and that each sample within a population is independent and identically distributed (i.i.d.). The second set of assumptions are that the distribution of the population means is known, and that the values of population means are i.i.d. For statistical analysis, the population variance is also assumed to be known. The final assumption is that the sampling overhead is the same for all populations and is equal to unity.

3.1.1. Upper-bound on Selection Performance

The best that any selection strategy can do is to be able to determine after one pick from each candidate the one with the best population mean. From the assumptions stated, the maximum number of candidates that can be tested in T time units is T. The distribution of the performance for the best candidate among T populations is the T^{th} order statistic of the population-mean distribution. Let $F_\mu(x)$ and $f_\mu(x)$ be the cumulative distribution function (c.d.f.) and probability density function (p.d.f.) of the population mean of a random candidate, then the p.d.f. of the best of T random candidates is [2]

$$f_{best}(x) = T\,[F_\mu(x)]^{T-1} f_\mu(x) \qquad (3.1)$$

3.1.2. Analysis of Selected Multistage Strategies

We have developed a general procedure for analyzing the multistage guidance strategies that can be adapted to any given guidance strategy. The key to the analysis is the ability to determine the joint probability distribution (f_{joint}) between the population mean, the sample mean, the number of samples per population, and any other variables used by the given guidance strategies to determine the next selection that can vary during the selection process. For static strategies, only the population mean, the sample mean and the number of samples have to be considered. For dynamic strategies, since the candidate is usually selected based on some values within the joint distribution, each pick must be treated as an individual stage, and the joint distribution must be updated after each pick.

There are four steps in the analysis of each stage using information available from the analysis of previous stages. The first step is to determine the joint probability distribution (f_{joint}) of a random candidate at the end of the current stage. Then the probability distribution of the sample means of the candidates at the end of the current stage is determined. In the third step, the sample mean distribution of the candidate that will be selected for further evaluation in the following stages (*i.e.* candidate in one of the top k among n sample means, where n is the number of candidates evaluated in this stage, and k is the number of candidates that will be selected for further evaluation in the following stages, if any) is determined. Finally, the results from the first three steps are used to determine f_{joint}, the joint probability distribution for the selected candidate. f_{joint} is the only information that needs to be carried over to the next stage.

Applying these steps on single-stage round-robin and two-stage round-robin strategies results in equations that accurately predict the performance of these strategies based on their parameters values and the population distribution. If we assume that there is only one set of parameters that provide the maximum performance level for a given population distribution, then this set of optimal parameters can be found by searching through the sets of possible parameters using the values predicted analytically. For a two-stage round-robin strategy, the result indicates that the available time should be divided so that 75% of the time is allocated to the first stage and 25% to the second stage.

For dynamic strategies, the results derived from the above technique are far too complex and do not provide insight into the method for selecting appropriate values for their parameters. Monte Carlo simulation results for two-stage round-robin/greedy strategy with a variety of parameters values indicate that the parameter set that is the best for a two-stage round-robin strategy performs well here.

3.1.3. Application strategy

The analysis presented in the previous section provides a method for determining the parameters for multistage selection for a given population distribution. In practice, the distributions of the population means and variances are unknown. Hence, it is necessary to estimate them by statistical tests before analysis can be performed. We assume that the initial 10 percent of the allotted time is used to estimate these parameters. Four samples per candidate are tested with as many candidates as can be tested during this initial period. The parameters of the selection process are then determined analytically. In case of dynamic strategies (two-stage round-robin/greedy and two-stage round-robin/minimum-risk), we use the same parameters as those of the two-stage round-robin strategy.

3.2. Revised Problem Model

The analysis in the last section is based on some simplifying assumptions. In the rest of this section we extend the model and present improved performance results.

The assumption that we relax is on the equality of sampling overheads of candidate heuristics. In this paper we assume that each candidate can have an independent and identically distributed sampling overhead drawn from a given distribution. We further assume that this distribution is known. Moreover, there is a lower bound on the sampling overhead; in this case we assume that this lower bound is 1.

As defined in Eq's (2.1) and (2.2), the objective of the generate-and-test process is a function of sampling cost, $c(h,v,t_{max})$ and sample quality $q(h,v)$. Note that the sampling cost is a function of the sampling overhead $t(h,v)$. In general, the sampling cost should be a monotonically nondecreasing function of sampling overhead. In our analysis, we assume that the joint distribution of the population mean, the population variance, and the sampling overhead is known.

3.3. Performance Upper Bound

Since the sampling overhead of all populations are greater than or equal to 1 under the new assumption, the maximum number of candidates that can be tested within T time units must be less than or equal to T. Therefore, the naive upper bound derived in Eq. (3.1) is still valid. However, since the average sampling overhead is greater than 1, the average number of candidates that can be tested within T time units, with one sample for each candidate, would be less than T. Using $g_c(x)$ and $G_c(x)$, the p.d.f. and c.d.f. of the sampling overhead of a random candidate, we can derive a better upper bound based on the distribution of N, the number of candidates, that can be sampled within T time units.

$$f_{best}(x) = \sum_{n=1}^{T} n \, [F_\mu(x)]^{n-1} f_\mu(x) P[N=n] \qquad (3.2)$$

$$P[N=n] = \int_{x=0}^{T} P[\sum_{i=1}^{n-1} c_i = x] \, (1-G_c(T-x)) \, dx \qquad (3.3)$$

where c_i is the sampling overhead of candidate i, and $F_\mu(x)$ and $f_\mu(x)$ represent the c.d.f. and p.d.f. of the population mean of a random candidate. Unfortunately, the distribution of $\left(\sum_{i=1}^{n-1} c_i\right)$ is an $(n-1)^{th}$ convolution of the sampling-overhead distribution, $g_c(x)$. This makes it very difficult to derive the result from this equation for any reasonable value of T. However, this upper-bound can be found easily using Monte Carlo simulation.

An approximation to the above equation can be found by using $\bar{c} = E[c]$, the average sampling overhead, instead of the distribution of sampling overheads in Eq. (3.2). This gives an estimate of the number of candidates that can be sampled in T time units. The corresponding distribution is shown in Eq. (3.4).

$$f_{best}(x) = \frac{T}{\bar{c}} \, [F_\mu(x)]^{\frac{T}{\bar{c}}-1} f_\mu(x) \qquad (3.4)$$

An alternative method for finding the upper bound is by the oracle argument. In this approach, we assume that best candidate is known ahead of time. The problem is to find the order the samples should be drawn so that the probability that the candidate with the highest population mean will have the highest sample mean when testing is completed.

Using this strategy, the testing time T is divided into two periods. During the first period of length $r\,T$, a sample is drawn from each candidate. In the second period, the oracle strategy uses the knowledge on the actual population means and variances in order to make selection that will optimize P_{select}, the probability that the candidate with highest population mean will have the highest sample mean at the end of the testing time. To determine the optimal set of picks, all possible combinations of picks must be enumerated and probabilities computed for each. The upper-bound performance is the set of selections with the maximum P_{select} among all possible value of r. This upper bound is better than the naive upper bound because it takes into account the variance within each population.

Assuming that the performance values of each candidate are normally distributed, the probability can be computed for all combinations of n_i, where n_i is the number of additional picks for candidate i.

$$P_{select} = \int_{x=-\infty}^{+\infty} \phi\left(\frac{x-\mu_{best}+(x-\nu_i)/n_{best}}{\sigma_{best}/\sqrt{n_{best}}}\right) \prod_{i \neq best} \Phi\left(\frac{x-\mu_i+(x-x_i)/n_i}{\sigma_i/\sqrt{n_i}}\right) dx \qquad (3.5)$$

where $\phi(\bullet)$ and $\Phi(\bullet)$ represent the p.d.f. and c.d.f. of $N(0,1)$, respectively, μ_i, x_i, and σ_i represent the population mean, the sample mean, and the population standard deviation of candidate i, respectively, and *best* represent the candidate with the highest population mean.

$F(n,p)$, the number of possible combinations for evaluating Eq. (3.5) for p picks and n populations, can be stated in the following recurrence.

$$F(n,p) = F(n-1,p) + F(n,p-1) \qquad (3.6)$$

with $F(1,p) = F(n,0) = 1$. The complexity of this recurrence grows in the order of 2^{n+p}. For even small values of n and p, the number of combinations is too large to be evaluated. This means that the optimal selection set cannot be determined by exhaustive search for any reasonable time limit.

A comparison of all these upper bounds is shown in Table 3.1. For this case we assume that the sampling overhead x is exponentially distributed with a minimum value of 2 and α of 0.4; that is

$$f(x) = \begin{cases} 0.4\, e^{-0.4(x-2)} & x \ge 2 \\ 0 & x < 2 \end{cases} \cdot \qquad (3.7)$$

We further assume that the performance measure is a ratio of quality and the sampling overhead ($O = q/c$), and the quality measure, q, has a normal distribution $N(0,1)$. The time allowed is 100 time units. By comparing the Monte Carlo simulation results against the upper bound obtained by the oracle argument, we found that the latter is much tighter. We also show the performance of simulations for the 2-stage round-robin/greedy strategy for comparison. We use in the rest of this paper Eq. (3.2) as the upper-bound estimate.

3.4. Performance of Selection Strategies

This section presents the performance of multistage selection strategies assuming random sampling overhead.

For dynamic strategies, such as greedy and minimum risk, they look ahead one unit of time in the future to make the current selection; hence, the variation in sampling overhead has no effect on their performance. More complex dynamic strategies may look further into the future to find a sequence of good selections that can optimize some performance criteria when time run out. In this case, the sampling cost must be taken into consideration when finding the optimal sequence of picks.

As shown in the previous section, the amount of computation time to find the optimal sequence increases exponentially with the length of the sequence and the number of candidates considered. In addition, the optimization criteria are usually based on estimated values of candidate parameters, which are likely to be inaccurate. The further into the future the procedure attempts to examine, the more complex the evaluation becomes and the less likely that the results will be useful.

The assumption on stochastic sampling overhead strongly affects static strategies such as round-robin. As shown in our previous work [3,4], the performance of the round-robin strategy depends on the number of populations under consideration and the number of samples drawn from each population. With a constant sampling overhead, the performance of the single-stage round-robin strategy when the population means have distribution $N(\mu_0,\sigma_0^2)$ is shown as follows.

Table 3.1. Upper bounds on Performance of Selection Strategies.

Strategy	Original Eq. (3.1)	Var. Cost Eq. (3.2)	App. Cost Eq. (3.4)	Oracle Strategy	2-Stage RR/Greedy
Perf.	0.856	0.596	0.603	0.482	0.460

$$E[\mu_{select}] = \mu_0 + \int_{x=-\infty}^{+\infty} \frac{\sigma_0^2\, x\, n\, e^{-\frac{x^2}{2}} [\Phi(x)]^{n-1}}{\sqrt{2\pi(\sigma^2/s + \sigma_0^2)}}\, dx \qquad (3.8)$$

where $\Phi(x)$ is the c.d.f. for an $N(0,1)$ distribution, μ_{select} is the population mean of the selected population, σ^2 is the population variance, n is the number of populations, and s is the number of samples drawn from each population.

When the sampling overhead is stochastic, the number of samples drawn from a fixed number of populations, n, and a given duration, T, is not constant in a round-robin strategy. Assuming that the population means have distribution $N(\mu_0,\sigma_0^2)$, the following equation can be derived.

$$E[\mu_{select}] \qquad (3.9)$$

$$= \mu_0 + \sum_{s=1}^{T/n} P[s] \int_{x=-\infty}^{+\infty} \frac{\sigma_0^2\, x\, n\, e^{-\frac{x^2}{2}} [\Phi(x)]^{n-1}}{\sqrt{2\pi(\sigma^2/s + \sigma_0^2)}}\, dx\ ,$$

where $P[s]$ is the probability for s (equal to T/n) samples to be drawn from the assumed population. Note that this distribution is the n^{th} convolution of the sampling-overhead distribution and is very difficult to compute.

An approximation to the performance of single-stage round-robin strategies can be obtained by using the average of the number of samples drawn, $\bar{s} = T/(n^*\bar{c})$ in Eq. (3.9), instead of weighting by $P[s]$. The same method can be used to approximate the performance of two-stage round-robin strategies. Therefore, a heuristic method for determining the parameters for single-stage and two-stage round-robin strategies with stochastic sampling overheads is to derive the results based on fixed sampling overheads and $T/\bar{c}$ picks from each population.

When the objective function is a function of the sampling overhead, the sampling-overhead distribution of the candidates selected for the next stage is not the same as the sampling-overhead distribution in the current stage. This will mainly affect the performance of the two-stage round-robin strategy. Since the change in sampling-overhead distribution is dependent on the relationship between the objective function and the sampling overhead, we will ignore this effect in this paper.

3.5. Revised Application strategy

In our original strategy, the parameters of the selection strategy were determined by spending the initial 10 percent of the allotted time to estimate their distributions; a binary search was then used to find the appropriate parameters for single-stage round-robin and two-stage round-robin strategies. The two-stage dynamic strategies used the same parameters as the ones found by the static counterpart.

This method must be updated to take into account the variable sampling costs. In the original method, T unit of time were assumed; hence, a total of T samples could be drawn. If the sampling overhead is not unity, then the actual number of samples drawn will be less than T. As a result, the parameters for controlling the original strategies were estimated incorrectly, which leads to poor performance.

One alternative is to estimate the average sampling overhead during the initial presampling period and to use it in subsequent control. The control parameters used after the presampling period are found by the original method using the average sampling overhead instead of unity.

Both the original and the new methods are applied to three problems in which the sampling overhead is defined based on Eq. (3.7) with different minimum values of 2, 3, and 4. It is assumed that performance is measured by the ratio of quality to sampling overhead, and the quality measure has a normal distribution $N(0,1)$. The performance for a two-stage round-robin strategy is plotted in Figure 3.2.

There are a couple of reasons why the performance of the original method is very close to the new one. Firstly, the performance degrades slowly when the values of the parameters deviate from the optimal. Since the old method overestimate the number of bags that can tested, the strategy end up with one sample drawn from each population. For the problem with population mean drawn from a normal distribution, the optimal number of samples drawn from each population is usually close to 1, so the performance of the old method did not degrade too badly. Secondly, candidates in the second stage is likely to have lower average sampling overhead than the original distribution due to the dependency of the objective value on the sampling overhead. This reduces the differences in performance between the original and the new methods further.

Figure 3.3. shows the performance for a two-stage round-robin/greedy strategy for the problems used in Figure 3.2. The differences in performance here is less than in two-stage round-robin since the parameters selected for two-stage round-robin/greedy strategy is heuristic and not necessarily optimal.

Figure 3.4. shows the performance of different strategies for this problem (with minimum value = 2) when using the new method for determining the control parameters. From this figure, the best strategy for this problem is the two-stage round-robin/minimum-risk method.

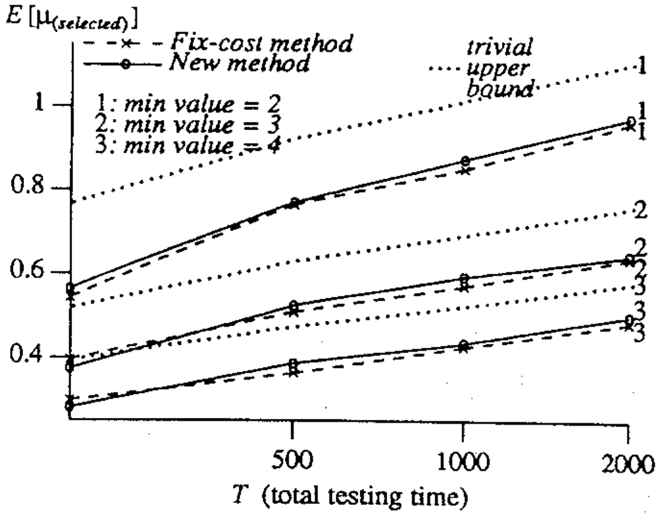


Figure 3.2. Performance of Run-Time Parameter Selection for 2-Stage Round-Robin.

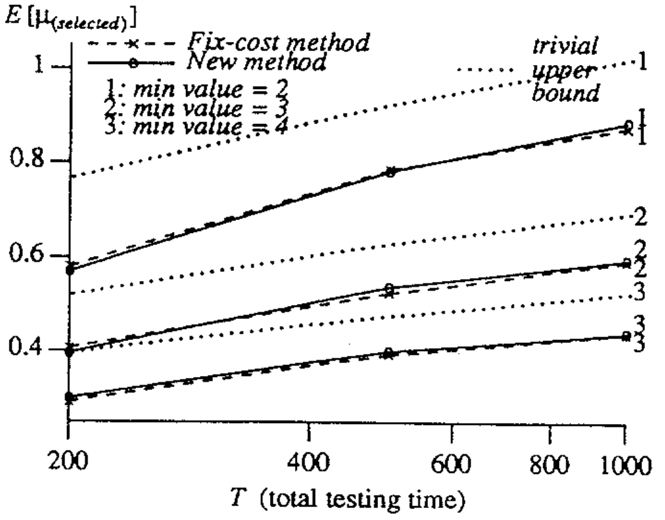


Figure 3.2. Performance of Run-Time Parameter Selection for 2-Stage Round-Robin/Greedy.

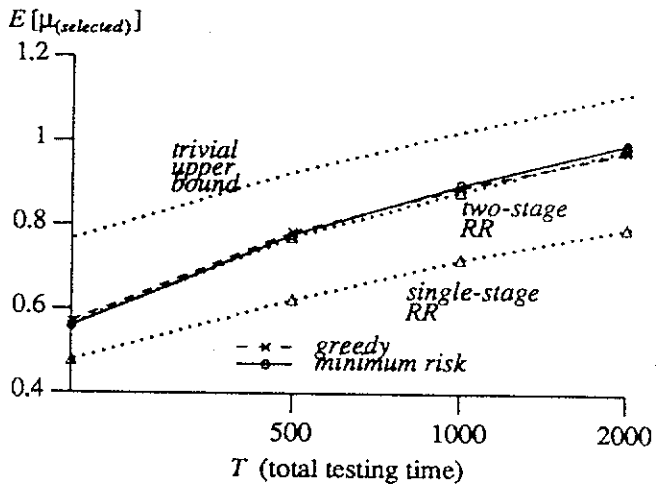


Figure 3.4. Performance of Multistage Strategies.

4. EXPERIMENTAL RESULTS

To verify the result we show in the previous section, we use TEACHER 4.1 to train the post-game system using a benchmark problem implementing a divide-and-conquer algorithm. There are 105 processes mapped to a 3-by-3 mesh architecture; 20 problem instances with CPU times drawn from the same distribution are included in the test set.

In the first experiment, the sampling overhead is ignored (*i.e.*, they are taken as unity). The time limit is 200 units. We apply four different strategies in this experiment: single-stage round-robin, two-stage round-robin, two-stage round-robin/greedy, and two-stage round-robin/minimum-risk. The number of candidates tested for the two-stage methods is 150. Table 4.1 shows the performance of different multistage selection strategies for various t_{max}.

In the second experiment, we consider the variation in sampling cost for the post-game heuristics. The time limit is 5000 minutes. The average time for one evaluation of a candidate is about 27 minutes. We apply the same strategies as in the first experiment using the our new method for determining the strategies parameters. The average number of candidates tested is 144. Table 4.2 shows the performance of different multistage selection strategies for various t_{max}.

Comparing the result in Tables 4.1 and 4.2, we see that the level of performance is about the same in most cases. Our limited experiments show that the multistage selection strategies we develop are robust to handle the variable sampling overhead and dependent objective function.

Table 4.1. Result of learning using different strategies under different t_{max}. Time allowed for learning is 200 quanta. The time for the initial testing period is 20 quanta.

t_{max} (secs)	Performance				
	Orig. Heur.	**1-stage RR**	**2-stage RR**	**2-stage RR/GR**	**2-stage RR/MR**
20	0.1425	0.6965	0.6965	0.6965	0.6965
40	0.4291	0.7722	0.7722	0.7722	0.7722
70	0.6039	0.7822	0.7916	0.7801	0.7916
120	0.7356	0.8082	0.8082	0.8036	0.7990
300	0.8086	0.8070	0.8063	0.8164	0.8164
600	0.8167	0.8219	0.8189	0.8132	0.8189
900	0.8182	0.8222	0.8171	0.8171	0.8171
∞	0.8182	0.8236	0.8186	0.8082	0.8186

Table 4.2. Result of learning using different strategies for different t_{max}. Time allowed for learning is 5000 minutes. The time for the initial testing period is 500 minutes.

t_{max} (secs)	Performance				
	Orig. Heur.	**1-stage RR**	**2-stage RR**	**2-stage RR/GR**	**2-stage RR/MR**
20	0.1425	0.6965	0.6965	0.6965	0.6965
40	0.4291	0.7722	0.7722	0.7722	0.7722
70	0.6039	0.7787	0.7920	0.7914	0.7920
120	0.7356	0.8054	0.8054	0.7970	0.7965
300	0.8086	0.8078	0.8095	0.8154	0.8095
600	0.8167	0.8143	0.8290	0.8174	0.8290
900	0.8182	0.8195	0.8200	0.8317	0.8183
∞	0.8182	0.8191	0.8156	0.8153	0.8156

Table 4.3. Number of selections for different t_{max} for two-stage round-robin/minimum-risk. Time allowed for learning is 5000 minutes.

t_{max}	20	40	70	120	300	600	900	∞
selection	281	223	202	199	162	174	185	180

It is not always true that the results in the case with variable sampling overhead is the same as those in the case with fixed sampling overhead. As indicated earlier, for objective functions that depend on the sampling overhead, the sampling-overhead distribution of the candidate selected is likely to be different from the original distribution. This is illustrated in Table 4.3 in which we show the number of selections done for different t_{max} using two-stage round-robin/minimum-risk method. With different levels of dependency between the objective and the sampling overhead, we see that there is a variation in the number of selections due to variations in sampling-cost distribution in the second stage. This is a phenomenon that our current strategies do not take into consideration. We plan to develop a method that takes advantage of this characteristic in the future.

5. FUTURE WORK

Our future work in this research includes (a) the application of our learning system to a variety of target process mapping problems; (b) the improvement of the candidate generator; (c) further analysis on the dependence between the variable sampling overhead and the objective function; and (e) the evaluation of parallel selection.

REFERENCES

[1] 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.

[2] R. V. Hogg and E. A. Tanis, *Probability and Statistical Inference*, 3rd Edition, Macmillan Publishing Company/Collier Macmillan Publishers, New York, NY/London, England, 1988.

[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*, ACM/IEEE, Albuquerque, NM, Nov. 1991.

[4] S. R. Schwartz, *Resource Constrained Parameter Tuning Applied to Stereo Vision*, M.Sc. Thesis, Department of Electrical and Computer Engineering, University of Illinois, Urbana, IL, August 1991.

[5] J. C. Yan, *Post-Game Analysis--A Heuristic Resource Management Framework for Concurrent Systems*, Ph.D. Dissertation, Dept. Elec. Eng., Stanford Univ., Dec. 1988.

[6] J. C. Yan and S. F. Lundstrom, "The Post-Game Analysis Framework--Developing Resource Management Strategies for Concurrent Systems," *Trans. on Knowledge and Data Engineering*, vol. 1, no. 3, IEEE, Sept. 1989.