Proceedings of the 1997 Winter Simulation Conference
ed. S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson

AN INTEGRATED FRAMEWORK FOR DETERMINISTIC AND STOCHASTIC OPTIMIZATION

Leyuan Shi
Sigurdur Ólafsson

Department of Industrial Engineering
University of Wisconsin-Madison
Madison, WI 53706, U.S.A.

ABSTRACT

In recent articles we presented a general methodology for finite optimization. The new method, the Nested Partitions (NP) method, combines partitioning, random sampling, a selection of a promising index, and backtracking to create a Markov chain that converges to a global optimum. In this paper we demonstrate, through examples, how the NP method can be applied to solve both deterministic and stochastic finite optimization problems in a unified framework.

1 INTRODUCTION

Constructing efficient algorithms for solving either NP-hard deterministic or finite stochastic optimization problems is perhaps one of the most challenging tasks facing the computer science, control theory, and operations research communities. It is not surprising that the first of these problems is listed in the Grand Challenges of the Federal High-Performance Computing and Communication (HPCC) program. Beyond toy problems, computational complexity is an obstacle faced by even the most elegant and effective approaches. In the case of stochastic systems, the situation is complicated even further by the added element of uncertainty.

The need to solve such problems arises in a wide variety of system design and control applications. Examples include optimal design of air- and spacecraft for low cost and/or high efficiency, optimal resource allocation, optimal job scheduling on parallel processors, and modeling and simulation of large molecular systems.

In general, the optimization problem considered in this paper is as follows. Given a finite feasible region Θ, and a performance measure $f : \Theta \to \mathbf{R}$, solve

$$\min_{\theta\in\Theta} f(\theta). \tag{1}$$

The problem above is therefore a combinatorial optimization problem. We also consider a finite stochastic optimization problem which takes the following form.

$$\min_{\theta\in\Theta} J(\theta) = E[L(\theta,\omega)], \tag{2}$$

where $J(\theta)$ is the average performance measure of interest, $L(\theta,\omega)$ is the sample performance and ω represents the stochastic effects of the system.

It should be noted that solving (2) is much more difficult than solving (1) as in most cases, an analytical expression relating the performance function $J(\theta)$ to a solution $\theta \in \Theta$ does not exist. Often, one has to use real-time observations or to resort to simulation in order to evaluate the objective function. In this paper, we will address both of these optimization problems in a unified framework. In particular, we will use discrete event simulation techniques for solving the stochastic problem.

Solving large-scale combinatorial problems is, as we stated above, a very difficult problem. The emergence of parallel processing capabilities opens up an intriguing possibility for the development of new parallel search techniques. Several applications of parallel computing have already been successfully developed in the field of combinatorial optimization (Ferreira and Pardalos 1996). Unfortunately, the theory of computational complexity has established several important problems, such as the traveling salesman problem and quadratic assignment problem, to be intrinsically difficult in the sense that the problem of finding an ϵ-approximate solution for these problems remains NP-hard (Sahni and Gonzalea 1976; Pardalos and Wolkowicz 1996). It is, therefore, widely believed that neither sequential nor parallel deterministic algorithms can be used to efficiently solve these problems.

The last decade has witnessed a rapid growth in the area of *randomized* algorithms. In particular, many Markov chain based methods have been proposed in recent years. Earlier methods include the *Metropolis method* (Hastings 1970), and *Simulated Annealing* (Kirkpatrick, Gelatt, and Vecchi 1983). Recent methods include those proposed by Ho and Larson (1995), Ho, Sreenivas and Vakili (1992), Yan and Mukai (1992), Andradóttir (1995), and the *Nested Partitions* (NP) method (Shi and Ólafsson 1996a). The Metropolis method and simulated annealing are often used to solve combinatorial optimization problems. On the other hand, the methods proposed by Yan and Mukai, and Andradóttir, aim at solving discrete stochastic optimization problems. Our proposed NP method can be applied to both optimization problems. A notable feature of the NP method is that it combines global and local search in a natural way. It is also highly suitable for parallel computer structures.

This paper is organized as follows. Section 2 provides a generic NP algorithm for both deterministic and stochastic optimization. Section 3 discusses the implementation of the NP method to a well-known combinatorial problem, namely the Traveling Salesman Problem. Section 4 presents an application of the NP algorithm to a stochastic discrete optimization problem. Some conclusions are drawn in the final section.

2 THE NESTED PARTITIONS METHOD

Given a finite solution space Θ, and a performance function denoted by either $f(\theta)$ or $J(\theta)$, the Nested Partitions (NP) method can be briefly described as follows. In each iteration of the algorithm we assume that we have a region, i.e., a subset of Θ, that is considered the *most promising.* We then *partition* this most promising region into M subregions and aggregate the entire surrounding region into one region. At each iteration, we therefore look at $M + 1$ disjoint subsets of the feasible region Θ. Each of these $M + 1$ regions is *sampled* using some random sampling scheme, and for each region a *promising index* is calculated. These promising indices are then compared to determine which region is the most promising in the next iteration. If one of the subregions is found to be best, this subregion becomes the most promising region. However, if the surrounding region is found to be best, the algorithm *backtracks* and a larger region containing the current most promising region becomes the new most promising region. The new most promising region is then partitioned and sampled in a similar fashion.

In the first iteration we normally use the entire feasible region Θ as the most promising region. Since the surrounding region is empty, we sample only from M regions in the first iteration, or in any iteration where Θ is considered the most promising region. It is clear that since Θ is finite, the partitioning can be continued until eventually all the regions are singletons and cannot be partitioned further. These regions are called regions of maximum depth, and more generally, we talk about the *depth* of any region. This is defined iteratively in the obvious manner, with Θ having depth 0 and so forth.

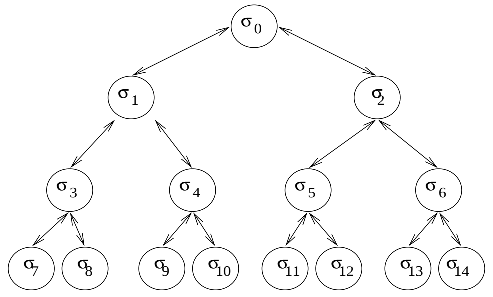

Figure 1: Partitioning Generated by the NP Method

2.1 Simple Example

As an example, consider a feasible region which consists of 8 points $\sigma_0 = \Theta = \{1, 2, 3, 4, 5, 6, 7, 8\}$. In each iteration the current most promising region is partitioned into two disjoint sets, that is, $M = 2$ (see Figure 1). At the first iteration, σ_0 is the most promising region and its subregions $\sigma_1 = \{1, 2, 3, 4\}$ and $\sigma_2 = \{5, 6, 7, 8\}$ are sampled. Assume that the promising index for σ_1 is better than for σ_2. We then select σ_1 as the most promising region in the second iteration and further partition it to obtain $\sigma_3 = \{1, 2\}$ and $\sigma_4 = \{3, 4\}$. In the second iteration, σ_3, σ_4, and the surrounding region, $\Theta \setminus \sigma_1 = \sigma_2$, are sampled. If the promising index of σ_3 is best, we then select σ_3 to be the most promising region in the third iteration and partition it further into another two subregions $\sigma_7 = \{1\}$ and $\sigma_8 = \{2\}$. On the other hand, if the promising index of the surrounding region, σ_2, is best, we backtrack and select σ_0 as the most promising region in the third iteration. Now assume that σ_3 is the most promising region. In the third iteration, σ_7, σ_8, and the surrounding region, $\sigma_0 \setminus \sigma_3$, are sampled. If the promising index of σ_7 is best, we select σ_7 as the most promising region. If the promising index of the surrounding region is the best, we select σ_1 as the most promising region. This proceeds until some stopping criterion is satisfied.

As the algorithm evolves, a sequence of the most promising regions $\{\sigma(k)\}_{k=0}^{\infty}$ is generated. Here $\sigma(k)$ is the most promising region in the k-th iteration. In Shi and Ólafsson (1996a, 1996b), we have shown that $\{\sigma(k)\}_{k=0}^{\infty}$ is a Markov chain which converges to a global optimum with probability one. In particular, we have shown that in the deterministic case $\{\sigma(k)\}_{k=0}^{\infty}$ is a Markov chain with all global optima as absorbing states, and $\{\sigma(k)\}_{k=0}^{\infty}$ is ergodic in the stochastic case.

2.2 The Nested Partitions Algorithm

The general procedure of the NP algorithm is as follows.

1. **Partitioning.**

   Partition the most promising region $\sigma(k)$, into $M_{\sigma(k)}$ subregions $\sigma_1(k), ..., \sigma_{M_{\sigma(k)}}(k)$, and aggregate the surrounding region $\Theta \setminus \sigma(k)$ into one region, denoted $\sigma_{M_{\sigma(k)}+1}(k)$.

   If we reconsider the example in Section 2.1, we could, for example, have $\sigma(k) = \sigma_4$, so $\sigma_1(k) = \sigma_9$, $\sigma_2(k) = \sigma_{10}$ and $\sigma_3(k) = \sigma_0 \setminus \sigma_4 = \sigma_3 \cup \sigma_2$.

   It should be noted that the partitioning scheme determines the state space of the Markov chain and should be fixed during implementation of the method. This means that if a region is selected repeatedly as the most promising region, then the same partitioning rule should be applied to the region each time.

2. **Sampling.**

   Randomly sample N_j points from each of the regions $\sigma_j(k),\ j = 1, 2, ..., M_{\sigma(k)} + 1$,

   $$\theta^{j1}, \theta^{j2}, ..., \theta^{jN_j},\quad j = 1, 2, ..., M_{\sigma(k)} + 1,$$

   and calculate the corresponding performance values $f(\theta)$ (or $L(\theta, \omega)$ in the stochastic case).

   $$f(\theta^{j1}), f(\theta^{j2}), ..., f(\theta^{jN_j}),\ \ j = 1, 2, ..., M_{\sigma(k)}+1.$$

   There is a great deal of flexibility in selecting a sampling strategy. In fact the only restriction is that each point in the sample region must have a positive probability of being selected.

3. **Calculating the Promising Index.**

   For each region $\sigma_j,\ j = 1, 2, ..., M + 1$, define a promising index function, $I(\sigma_j)$, and calculate the *promising index.* For example, define $I(\sigma_j)$ as the best performance value in the region

   $$I(\sigma_j) = \min_{\theta \in \sigma_j} f(\theta),\quad j = 1, 2, ..., M_{\sigma(k)} + 1 \quad (3)$$

   and estimate $I(\sigma_j)$ using

   $$\hat{I}(\sigma_j) = \min_{i=1,2,...,N_j} f(\theta^{ji}),\ \ j = 1, 2, ..., M_{\sigma(k)}+1. \quad (4)$$

   We refer to this promising index as the *ordinal promising index.* Many other options exist, and we refer the reader to Shi and Ólafsson (1996a, 1996b) for details.

4. **Backtracking.**

   Determine the *most promising region* σ_{j_k}.

   $$j_k \in \arg \min_{j=1,...,M_{\sigma(k)}+1} \hat{I}(\sigma_j). \quad (5)$$

   If more than one region is equally promising, the tie can be broken arbitrarily. If the index corresponds to a subregion of $\sigma(k)$, then let this subregion be the most promising region in the next iteration. Otherwise, if the index corresponds to the surrounding region, backtrack to the region which is the parent of the current most promising region. The depth of the parent region is one less than the depth of the current most promising region.

It should be noted that the NP algorithm can be applied to both deterministic and stochastic optimization problems. If the performance function can be evaluated accurately then we have a deterministic optimization problem, otherwise we have a stochastic optimization problem.

From Figure 1, it is clear why the algorithm is termed the *nested partitions* method. The feasible region is partitioned iteratively so that each partition is nested within the last. Hence the basic idea of the algorithm is to shift the focus from points in the feasible region to a sequence of subsets of the feasible region. Due to this shift in focus, and the fact that the promising index is defined on these sets, we can incorporate any effective heuristic method into the algorithm by using it to define the promising index. This is possible because the only restriction for selecting an appropriate promising index is that a promising index function should agree with the performance measure on regions of maximum depth, i.e., on singletons. Therefore, finding the optimal solution for the original optimization problem is equivalent to finding the maximum depth region with the best promising index value.

Another important consequence of shifting the focus to subsets of the feasible region is that it makes the NP method highly compatible with parallel computer structures. Each subregion can be treated independently and in parallel. Therefore, the algorithm can take maximum advantage of the available parallel resources. As high performance computing facilities become more available, this feature will become increasingly important.

The method described so far can be considered a generic algorithm capable of supporting different partitioning techniques, sampling methods, promising indices, and backtracking rules. In the following we demonstrate through examples how the NP method can be applied to combinatorial and finite stochastic optimization problems.

3 THE NP METHOD FOR COMBINATORIAL OPTIMIZATION

In this section we describe how the NP algorithm can be applied to a well-known combinatorial problem, namely the Traveling Salesman Problem (TSP). The TSP is one of the most prominent members in the rich set of combinatorial optimization problems (Lawler, Lenstra, Rinnooy, and Shmoys 1985; Reinelt 1992). Originally formulated as the problem of finding the shortest route for a traveling salesman to visit all of his customers, the problem has found many important applications such as routing robots through automatic warehouses, sending couriers to automatic teller machines, and drilling holes through printed circuit boards. The list of applications continues, making the TSP one of the most important combinatorial optimization problems.

3.1 The Traveling Salesman Problem

As stated above, the TSP is the task of finding a route with the shortest possible length through a given set of cities. Formally, the problem consists of a number of cities, represented by vertices in a graph, and a number of connections, or edges, between the cities. Each edge is associated with a cost, which represents the cost of traveling between the two cities connected by the edge. The objective is to find the tour that passes through each city exactly once and returns to the starting point, such that the overall cost of traveling is minimized. Therefore, given a cost matrix $C = (c_{ij})_{i,j=0,1,...,n}$, where c_{ij} is the cost of going from city i to city j, the TSP problem can be stated mathematically as follows

$$\min_{\theta\in\Theta} f(\theta) \equiv \min_{\theta\in\Theta}(c_{i_0 i_1} + c_{i_1 i_2} + ... + c_{i_n i_1}), \qquad (6)$$

where $\theta = (i_0, i_1, ..., i_n)$ is a permutation of $\{0, 1, ..., n\}$ and Θ is the set of all such permutations.

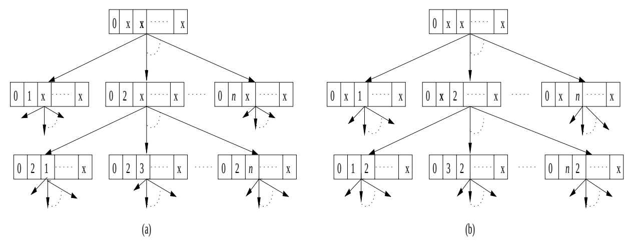

Figure 2: Generic Partitioning for the TSP

We assume the TSP is symmetric. This implies that $c_{ij} = c_{ji}$, that is, the cost of traveling from city i to city j is exactly the same as that from city j to city i for all cities i and j.

3.2 Partitioning

To apply the NP algorithm to the TSP problem, we first consider how to partition the solution space into subregions. This must be done in such a way that they can be partitioned further until each subregion is a singleton. Although there are no further restrictions on the partitioning strategy, it determines the efficiency of the algorithm. If the partitioning is such that good solutions are clustered together, the NP algorithm quickly identifies a set of near optimal solutions. In this paper, we introduce one partitioning technique, namely generic partitioning. Other partitioning approaches can be found in Shi, Ólafsson, and Sun (1997).

We present a generic way to partition the solution space. By this we mean a partitioning of the solution space that does not consider the objective function. Given $n+1$ cities, suppose we choose city 0 as the starting point and other cities are labeled as $1, 2, ..., n$. The whole solution space becomes all permutations of $\{1, 2, ..., n\}$. First, divide the solution space into n equal parts by fixing the first city on the tour to be one of $1, 2, ...,$ or n (note that $M_{\sigma(0)} = n$). Then further partition each such subregion into $n-1$ parts by fixing the second city as any of the remaining $n-1$ cities on the tour. This procedure can be repeated until all the cities on the tour are fixed. At this point the subregions are all singletons and the maximum depth is hence reached. Figure 2a illustrates this approach.

It should be noted that there exist many such partitions. For example, after choosing city 0 as the starting point, instead of fixing the first city on the tour, we could fix any i-th city on the tour to be one of cities $1, 2, ..., n$ (see Figure 2b). This provides a completely different set of subregions.

The advantage of generic partitioning is that the search tree is completely predictable in general and is highly regular in terms of branching degrees and searching depths. Therefore, this type of partitions is ideal for parallel algorithms.

3.3 Random Sampling

Assume that generic partitioning is used (see Figure 2), and the current most promising region is of depth k. This means that the first k edges in the tour have been determined. Obtaining a sample from this region entails finding the $n-k$ remaining edges. One approach would be simply to pick the edges consecutively, such that each feasible edge has equal probability of being picked. This corresponds to uniform sampling. However, this approach may not always give good results in practice. The intuitive reason is that uniform sampling considers only the solution space itself. To incorporate the objective function into the sampling, we present the following *weighted sampling* scheme.

Assume a current most promising region defined by the first k cities being fixed, that is, a sequence of vertices $v_1, v_2, ..., v_k$ has been selected. The following algorithm then determines the remaining $n-k$ vertices.

```
Given a constant $p \in [0,1]$.
**for** $i = k+1 : n$ **do**
    **remark** Given $v_{i-1}$ determine $v_i$
    **let** $u$ be generated from $U(0,1)$.
    **if** $u < p$,
        **let** the next edge $(v_{i-1}, v_i)$ be the lowest
            cost edge.
    **else**,
        **let** $(v_{i-1}, v_i)$ be a random edge according
            to a uniform distribution.
    **end**
**end**
```

It should be noted that the case $p = 1$ corresponds to a nearest neighbor search and the case $p = 0$ corresponds to uniform sampling. For $0 < p < 1$ it is a non-uniform sampling scheme that picks each feasible tour with a positive probability. Hence it is a valid sampling scheme for the NP method.

We call readers' attention to the fact that the weighted sampling scheme provides a new construction heuristic for the TSP, which as far as we know has not been previously reported in the literature.

3.4 Calculating the Promising Index

After using the weighted sampling scheme to select N_j points from each subregion, we need to define the *promising index*, $I(\sigma_j)$, that will be used to determine the most promising region. There exist many potential candidates for the function (Shi, Ólafsson, and Sun 1997). In this paper, we consider a promising index defined as follows.

Since the promising index function is defined on the set Σ, which is a collection of subsets, it is possible to incorporate many effective heuristic methods into the NP algorithm when the current subregion is not of maximum depth. That is, we can take the N_j sampling points as initial points and for each of these sampling points, perform a fixed number of improvements based on a given heuristic method. We define the promising index function as in (3) and estimate it as follows. Let $H : \Theta \times \Sigma \to \Theta$ be a function that transforms a point $\theta_0 \in \sigma$ into another point $\theta_1 = H(\theta_0, \sigma) \in \sigma$, by applying one or more iterations of some heuristic search method. The starting point of the search is $\theta_0 \in \sigma$, and the search is constrained to stay within $\sigma \in \Sigma$. Given this function, the estimated promising index may be defined as follows.

$$\hat{I}(\sigma_j) = \min_{i=1,2,...,N_j} f(H(\theta^{ji}, \sigma_j)). \tag{7}$$

To make this concrete, we can for example define H as m iterations of the well known 2-opt exchange heuristic (Reinelt 1992). Note that $m = 0$ implies that no heuristic improvements are made and $m \to \infty$ implies an exhaustive heuristic search.

3.5 Backtracking

The NP method provides great flexibility in selecting a backtracking rule. Perhaps the simplest rule is to always move to the immediate superregion of the current most promising region. However many other alternatives exist. For example, we could backtrack all the way to the entire feasible region, or to any region that is between the superregion and the entire feasible region. We can also consider the superregion of the best tour found in this iteration in the surrounding region.

To provide a systematic way for backtracking, we adopt the following approach in this paper. If the depth of current region is less than the maximum depth, then the algorithm backtracks to a superregion of the best solution found during this iteration. This superregion is determined such that it has less depth than the current region. For example, if the current most promising region is $(i_0^*, i_1^*, ..., i_k^*, x, x, ..., x)$ and the best solution, $(j_0^*, j_1^*, ..., j_n^*)$, is found in the surrounding region, then the next most promising region is determined to be $(j_0^*, j_1^*, ..., j_{k-h}^*, x, x, ..., x)$. If the current region is at maximum depth, then the algorithm backtracks to a superregion of the best solution found in the current iteration. The superregion is determined by a predetermined depth $h > 0$ as before. For example, if the current region is at the maximum depth and the best solution, $(j_0^*, j_1^*, ..., j_n^*)$, is found in the surrounding region, then the next most promising region is determined to be $(j_0^*, j_1^*, ..., j_{n-h}^*, x, x, ..., x)$.

The advantage of the abovementioned backtracking rule is that if the ancestor of a better solution has a higher probability of containing the optimal tour than the current promising region, the algorithm will quickly identify the subregion which contains the optimal solution.

3.6 Selecting the Initial Most Promising Region

When describing the generic NP method in Section 2, we assumed that the most promising region in iteration zero is the entire feasible region. This assumes that at iteration zero there is no knowledge available about where good solutions might be located. However, if such knowledge is available, the NP method can use any other valid region $\sigma(0) \in \Sigma$ as the initial most promising region. The information leading to a specific initial region may for example come from previous numerical experience, but we can also go through an initialization phase to find such a region. In particular, for the TSP, we can use the nearest neighbor heuristic to quickly get a moderately good tour. This tour can then be truncated to any specific length k, and the valid region defined by these first k edges being fixed can be used as an initial most promising region.

As we can see, there exist many ways to select the initial most promising region. The advantage of using a subregion with k cities fixed on a tour as the initial most promising region is that such a region has only $n - k$ subregions. Therefore, the computational effort will be less in the first iterations then if there were n subregions in the beginning. Our numerical experience shows that this approach may be efficient when a very limited computation budget is available (Shi, Ólafsson, and Sun 1997).

4 THE NP METHOD FOR STOCHASTIC FINITE OPTIMIZATION

In this section, we use a machine allocation problem (Frenk, Labbe, van Vliet, and Zhang 1994) to illustrate the NP method for stochastic finite optimization problems.

4.1 Machine Allocation

A manufacturing system consists of N workstations and a total of M servers. There are several product types produced by the system. Products of each type arrive at the first workstation according to a certain distribution, and then follow a deterministic route through a subset of the workstations. The problem is to allocate the M servers to workstations in such a way that the total Work-in-Process (WIP) is minimized.

Mathematically, this problem can be formulated as follows.

$$\min_{\theta \in \Theta} J(\theta) = \sum_{i=1}^{N} E[L_t(\theta, \omega)] \tag{8}$$

s.t.

$$1 \leq \theta_j \leq M + N - 1, \quad j = 1, ..., N, \tag{9}$$

$$\sum_{j=1}^{N} \theta_j = M, \tag{10}$$

where Θ is a set of all feasible allocations, $J(\theta)$ is the total average WIP, $L_t(\theta, \omega)$ is the sample WIP for the i-th workstation when the t is the length of the sample path, and ω represents the stochastic effects of the system.

4.2 Partitioning

We consider two partitioning techniques. The first is generic and resembles the generic partitioning strategy for the TSP. The second builds on the generic partitioning approach, but incorporates information about the performance function. Hence, we refer to the second partitioning technique as knowledge-based partitioning.

4.2.1 Generic Partitioning

We first partition the feasible region without considering its objective function. For the machine allocation problem there is a total of M servers and a minimum of one server that is allocated to each workstation. Hence, we can first divide this solution space into $M - N + 1$ subregions by fixing the first workstation to have $1, 2, ...,$ or $M - N + 1$ servers. In the above terminology, this implies that θ_1 takes a fixed value in $\{1, 2, ..., M-N+1\}$. We can further partition each such subregion by fixing the second workstation to have any number of the remaining servers. This procedure can be repeated until all the servers are allocated to the workstations, i.e., the maximum depth is reached.

Generic partitioning focuses on the solution space only. Intuitively, it would seem that more efficient partitions could be constructed if the objective function was considered. This is the second approach to partitioning that we consider.

4.2.2 Knowledge-Based Partitioning

The fact that generic partitioning does not consider the objective function may lead to difficulties in distinguishing between regions, and consequently the algorithm may not find any particular region to concentrate the computational effort. This has been our experience with some large combinatorial problems (Shi, Ólafsson, and Sun 1997). If the NP method is applied using generic partitioning, it may backtrack frequently and not settle down in a particular region. On the other hand, the NP method is likely to perform much better if good solutions tend to be clustered together for a given partitioning. To impose such structure, we provide the following partitioning scheme for the machine allocation example. First identify a bottleneck workstation and then divide the solution space into $M-N+1$ subregions by fixing the bottleneck workstation to have $1, 2, ...,$ or $M-N+1$ servers. Each subregion can be partitioned further using the same approach.

Figure 3 illustrates this approach for a simple system with $N = 4$ workstations and $M = 10$ servers. Here the bottleneck workstation is workstation three, so that workstation is fixed in the first iteration. Notice, that by fixing this workstation first, there are more available servers to be allocated, and there is hence a higher probability of allocating a large number of servers to this station. Now, say that four servers have been allocated to the third workstation and that the first workstation is to be fixed in the second partitioning level (see Figure 3). Now, workstation one can get at most four servers, whereas the third workstation had the potential of getting up to seven servers. It is clear that the order in which the workstations are selected affects the probability distribution of how many servers they are allocated.

4.3 Random Sampling

Recall that the only restriction for a valid random sampling scheme is that each point in a sampling region should have a positive probability to be selected. One approach would be to use uniform sampling scheme. Alternatively we can combine the uniform sampling with other techniques to generate new sample points. For example, we could use the following sampling scheme.

We first define a probability distribution for how many of the available servers will be assigned to the next workstation. Assume that the current number of available servers is M_a, then we can assign a probability, $p_i = i/\sum_{l=1}^{M_a} l$ to assigning i servers to the next workstation. Thus, the workstations selected early will have a higher probability of getting more servers than the workstations selected when most of the servers may have been used up. In other words, the sampling distribution is skewed towards allocating many servers to workstations that are selected to be fixed early. To utilize this we assign a weight to each workstation. If the arrival rates, λ_j, at each workstation j are different, for example due to feedback rates, then we could assign a weight, $w_j = \lambda_j/\sum_{l=k+1}^{N} \lambda_l$ to the j-th workstation. These weights determine the probability with which each workstation is selected to be next. Therefore, the workstation with the greatest arrival rate will be selected with maximum probability and will consequently be the most likely to receive many servers. This procedure can be repeated until all the workstations have been assigned resources.

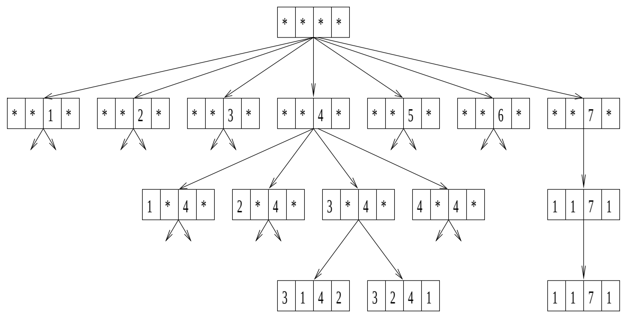

Figure 3: Knowledge-Based Partitioning

4.4 Calculating the Promising Index

As we have discussed in Section 3.4, there are many ways to select the promising index. As before, we use the ordinal promising index.

$$I(\sigma_j) = \min_{\theta \in \sigma_j} J(\theta), \quad j = 1, 2, ..., M+1.$$

Let $\hat{I}_t(\sigma_j, \omega)$ be an estimate of $I(\sigma_j)$, where t is the simulation time. Then $\hat{I}_t(\sigma_j, \omega)$ can be defined as follows.

$$\hat{I}_t(\sigma_j, \omega) = \min_{i=1,2,...,N_j} L_t(\theta_i^j, \omega), \quad j = 1, 2, ..., M+1. \tag{11}$$

Note that the estimated promising index function incorporates the sample performance $L_t(\theta_i^j, \omega)$ directly. This implies that for every sample point, we need to have only a single simulation run to estimate the promising index. Even a very short single simulation run will often suffice for the following reason. In Shi and Ólafsson (1996b) we show that to guarantee global convergence of the NP method all we need is to preserve the rank of the estimation, i.e., all we are interested is how fast the estimated rank converges to the true rank. It has recently been shown that, given certain conditions, the estimated rank converges to the true rank at an exponential rate (Dai 1996). This fast convergence rate provides an opportunity to use a single short simulation run to estimate the promising index.

We will not discuss the backtracking and selection of the initial promising region procedures, since they are similar to the TSP problem.

5 CONCLUSIONS

We have demonstrated how the NP method can be applied to solve both deterministic and stochastic optimization problems. This method uses partitioning and random sampling to globally search the entire solution space, and can incorporate local search heuristic into a promising index that is used to determine where to concentrate the search. The method is easy to implement and is highly matched to parallel computer architectures.

Future work will focus on numerical experiments with emphasis on stochastic optimization problems as well as a parallel version of the algorithm.

REFERENCES

Andradóttir, S. 1995. A method for discrete stochastic optimization. *Management Science* 41: 1946-1961.

Dai, L. 1996. Convergence properties of ordinal comparison in the simulation of discrete event dynamic systems. *Journal of Optimization Theory and Applications* 19:363-388.

Ferreira, A., and P. Pardalos (eds.). 1996. *Solving combinatorial optimization problems in parallel.* New York: Springer-Verlag.

Frenk, H., M. Labbe, M. van Vliet, and S. Zhang. 1994. Improved algorithms for machine allocation in manufacturing systems. *Operations Research* 42:523-530.

Hastings, W.K. 1970. Monte Carlo sampling method using Markov chains and their applications. *Biometrika* 57:92-109.

High Performance Computing and Communications: Foundation for America's Information Future, from *http://www.hpcc.gov/blue96/index.html.*

Ho, Y.C., and M. Larson. 1995. Ordinal optimization and rare event probability simulation. *J. Discrete Event Dynamic Systems* 5:281-301.

Ho, Y.C., R.S. Sreenivas, and P. Vakili. 1992. Ordinal optimization of DEDS. *J. Discrete Event Dynamic Systems* 2:61-88.

Lawler, E.L., J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys (eds.). 1985. *The traveling salesman problem.* Chichester: John Wiley & Sons.

Pardalos, P., and H. Wolkowicz. 1996. Quadratic assignment and related problems. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science,* American Mathematical Society.

Reinelt, G. 1994. *The Traveling salesman: computational solutions for TSP applications.* New York: Springer-Verlag.

Sahni, S., and T. Gonzalea. 1976. P-Complete approximation problems. *Journal of ACM* 23:555-565.

Shi, L., and S. Ólafsson. 1996a. Nested partitions method for global optimization. Submitted to *Operations Research.*

Shi, L., and S. Ólafsson. 1996b. Nested partitions method for stochastic optimization. Submitted to *Management Science.*

Shi, L., S. Ólafsson, and N. Sun. 1997. Nested partitions method for traveling salesman problem. Submitted to *Computers & Operations Research.*

Yan, D., and H. Mukai. 1992. Stochastic discrete optimization. *SIAM J. Control and Optimization* 30:594-612.

AUTHOR BIOGRAPHIES

LEYUAN SHI is an Assistant Professor in the Department of Industrial Engineering at the University of Wisconsin-Madison. She holds a B.S. degree in Mathematics from Nanjing Normal University, China (1982), an M.S. degree in Applied Mathematics from Tsinghua University, China (1985), and an M.S. and a Ph.D. degrees in Applied Mathematics from Harvard University (1990,1992). Her research interests include modeling, analysis, and optimization of discrete event systems, discrete-event simulation, and sensitivity analysis.

SIGURDUR ÓLAFSSON is a Ph.D. student in the Department of Industrial Engineering at the University of Wisconsin - Madison. He received a B.S. in Mathematics from the University of Iceland in 1995, and an M.S. in Industrial Engineering from the University of Wisconsin - Madison in 1996. His research interests include stochastic optimization and simulation.