HOW TO COPE WITH ANOMALIES IN PARALLEL APPROXIMATE BRANCH-AND-BOUND ALGORITHMS

Guo-jie Li and Benjamin W. Wah
School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

Abstract: A general technique for solving a wide variety of search problems is the branch-and-bound (B&B) algorithm. We have adapted and extended B&B algorithms for parallel processing. Anomalies owing to parallelism may occur. In this paper sufficient conditions to guarantee that parallelism will not degrade the performance are presented. Necessary conditions for allowing parallelism to have a speedup greater than the number of processors are also shown. Anomalies are found to occur infrequently when optimal solutions are sought; however, they are frequent in approximate B&B algorithms. Theoretical analysis and simulations show that a best-first search is robust for parallel processing.

1. INTRODUCTION

The search for solutions in a combinatorially large problem space is very important in artificial intelligence (AI) [8]. Combinatorial-search problems can be classified into two types. The first type is decision problems that decide whether at least one solution exists and satisfies a given set of constraints. Theorem-proving, expert systems and some permutation problems belong to this class. The second type is optimization problems that are characterized by an objective function to be minimized or maximized and a set of constraints to be satisfied. Practical problems, such as traveling salesman, job-shop scheduling, knapsack, vertex cover, and game-tree search belong to this class.

A general technique for solving combinatorial searches is the B&B algorithm [6]. This is a partitioning algorithm that decomposes a problem into smaller subproblems and repeatedly decomposes until infeasibility is proved or a solution is found [6]. It can be characterized by four constituents: a branching rule, a selection rule, an elimination rule and a termination condition. The first two rules are used to decompose the problem into simpler subproblems and appropriately order the search. The last two rules are used to eliminate generated subproblems that are infeasible or that cannot lead to a better solution than an already-known feasible solution. Kumar et al. have shown that the B&B approach provides a unified way of formulating and analyzing AND/OR tree searches such as SSS* and Alpha-Beta search [4]. The technique of branching and pruning in B&B algorithms to discover the optimal element of a set is the essence of many heuristic procedures in AI.

To enhance the efficiency of implementing B&B algorithms, approximations and parallel processing are two major approaches. It is impractical to use parallel processing to solve intractable problems with exponential complexity because an exponential number of processors must be used to solve the problems in polynomial time in the worse case. For these problems, approximate solutions are acceptable alternatives. Experimental results on vertex-cover, 0-1 knapsack and some integer-programming problems reveal that a linear reduction in accuracy may result in an exponential reduction in the average computational time [10]. On the other hand, parallel processing is applicable when the problem is solvable in polynomial time (such as finding the shortest path in a graph), or when the problem is NP-hard but is solvable in polynomial time on the average [9], or when the problem is approximately solvable in polynomial time (such as game-tree search).

Analytical properties of parallel approximate B&B (PABB) algorithms have been rarely studied. In general, a k-fold speedup (ratio of the number of iterations in the serial case to that of the parallel case) is sought when k processors are used. However, simulations have shown that the speedup for PABB algorithms using k processors can be (a) less than one--"*detrimental anomaly*" [3,5]; (b) greater than k--"*acceleration anomaly*" [3,5]; or (c) between one and k--"*deceleration anomaly* [3,5,10]. Similar anomalous behavior have been reported by others. For instance, the achievable speedup for AND/OR-tree searches is limited by a constant (5 to 6) independent of the number of processors used (parallel-aspiration search) or $\sqrt{k}$ with k processors (tree-splitting algorithm) [1]. So far, all known results of parallel tree searches showed that a near-linear speedup holds only for a small number of processors. It is desirable to discover conditions that preserve the acceleration anomalies, eliminate the detrimental anomalies and minimize the deceleration anomalies. The objectives of this paper are to provide conditions for achieving the maximum speedup and to find the appropriate parallel search strategy under which a near-linear speedup will hold for a considerable number of processors.

2. PARALLEL APPROXIMATE BRANCH-AND-BOUND ALGORITHMS

Many theoretical properties of serial B&B algorithms have been developed [2], and a brief discussion is given here. In this paper minimization problems are considered. Let P_i be a subproblem, i.e., a node in the state-space tree, and $f(P_i)$ be the value of the best solution obtained by evaluating all the subproblems decomposable from P_i. A lower bound, $g(P_i)$, is calculated for P_i when it is created. If a subproblem is a feasible solution with the best objective-function value so far, the solution value becomes the *incumbent* z. The incumbent represents the best solution obtained so far in the process. During the computation, P_i is terminated if:

$$g(P_i) \geq z \tag{1}$$

The approximate B&B algorithm is identical to the optimal algorithm except that the lower-bound test is modified to:

$$g(P_i) \geq \frac{z}{1+\epsilon} \qquad \epsilon \geq 0,\ z \geq 0 \tag{2}$$

where ϵ is an *allowance parameter*. The final incumbent value z_F obtained by the modified lower-bound test deviates from the optimal solution value, z_O, by:

$$\frac{z_F}{1+\epsilon} \leq z_O \leq z_F \tag{3}$$

Let **L** denotes the lower-bound cutoff test, that is, $P_j \mathbf{L} P_i$ means that P_j is a feasible solution and $f(P_j)/(1+\epsilon) \leq g(P_i)$, $\epsilon \geq 0$. For example, in Figure 1, $P_1 \mathbf{L} P_2$, since $91/1.1 < 85$, and similarly $P_4 \mathbf{L} P_1$. However, $P_4 \mathbf{L} P_2$ is false because $100/1.1 > 85$.

Ibaraki mapped breadth-first, depth-first and best-first searches into a general form called ***heuristic searches*** [2]. A heuristic function is used to define the order in which subprob-

Research was supported by National Science Foundation Grant ECS81-05968.

AAAI-84 National Conference on Artificial Intelligence.

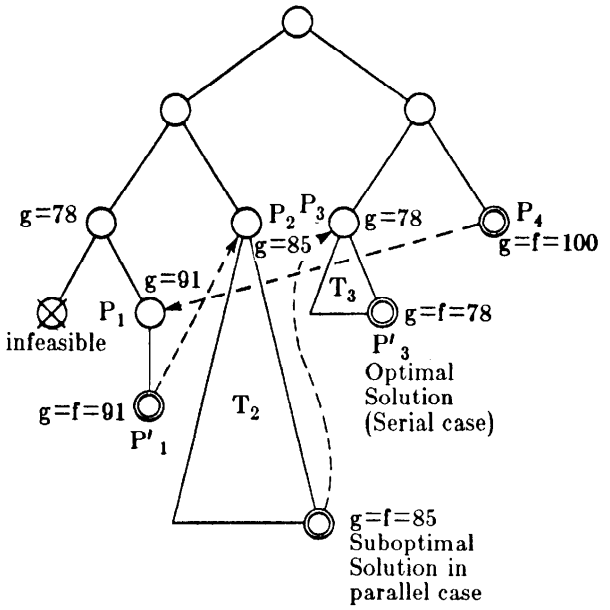

Figure 1. Example of a detrimental anomaly under a parallel depth-first search (ϵ=0.1).

lems are selected and decomposed. The algorithm always decomposes the subproblem with the minimum heuristic value. In a best-first search, the lower-bound values define the order of expansion, hence the lower-bound function can be taken as the heuristic function. In a breadth-first search, subproblems with the minimum level numbers are expanded first. The level number can, thus, be taken as the heuristic function. Lastly, in a depth-first search, subproblems with the maximum level numbers are expanded first. The negation of the level number can be taken as the heuristic function.

Branch-and-bound algorithms have inherent parallelism:

(a) *Parallel selection of subproblems*: A set of subproblems less than or equal to in size to the number of processors have to be selected for decomposition in each iteration. A selection function returns k subproblems with the minimum heuristic values from **U**, where k is number of processors and **U** is the active list of subproblems. The selection problem is especially critical under a best-first search because a set of subproblems with the minimum lower bounds must be selected.

(b) *Parallel branching*: The subproblems assigned to the processors can be decomposed in parallel. In order for the processors to be well utilized, the number of active subproblems should be greater than or equal to k.

(c) *Parallel termination test*: Multiple infeasible nodes can be eliminated in each iteration. Further, multiple feasible solutions may be generated, and the incumbent may have to be updated in parallel.

(d) *Parallel elimination test*: If the incumbent is accessible to all the processors, the lower-bound test (Eq's 1 or 2) can be carried out in parallel.

The above sources of parallelism has been studied in MANIP, a multiprocessor implementing PABB algorithms with a best-first search and lower-bound tests [10].

3. ANOMALIES ON PARALLELISM

In this section anomalies are studied under lower-bound elimination and termination rules. The results on anomalies with dominance tests are shown elsewhere [7]. For simplicity, only the search for a single optimal solution is considered here.

A synchronous model of PABB algorithm is used. The incumbent is stored in a global register that can be updated concurrently. Active subproblems can be stored in a centralized list or multiple lists. The distinction lies in the memory configuration. When all the processors are connected to a centralized memory, the subproblem list is global to the processors. When each processor has a private memory, only the local subproblem list can be accessed. The sequence of operations performed in an iteration are selection, branching, feasibility and elimination tests, and inserting newly generated subproblems into the list(s). Let $T^c(k,\epsilon)$ and $T^d(k,\epsilon)$ denote the number of iterations required for expanding a B&B tree using centralized and k subproblem lists respectively, where k is the number of processors used, and ϵ is the allowance parameter.

An example of a detrimental anomaly is illustrated in Figure 1. In a serial depth-first search, subtree T_2 is terminated owing to the lower-bound test of P_1': $f(P_1')/(1+\epsilon) \leq g(P_2)$ where $\epsilon = 0.1$. In a parallel depth-first search with two processors, a feasible solution, P_4, that terminates P_1 and P_1' is found in the second iteration. As mentioned before, P_2 is not eliminated by P_4. Consequently, subtree T_2 has to be expanded that will eventually terminate subtree T_3. If the size of T_2 is much larger than the size of T_3, the time it takes to expand T_2 using two processors will be longer than the time it takes to expand T_3 using one processor. Note that the above anomaly does not happen in a best-first search because subtree T_2 is not expanded in both the serial and the parallel cases.

An example of an acceleration anomaly is shown in Figure 2. When a single processor with a depth-first search is used, subtree T will be expanded since $f(P_1)/(1+\epsilon) > g(P_2)$ where $\epsilon = 0.1$. When two processors are used, P_2 and hence T will be terminated by lower-bound tests with P_3: $f(P_3)/(1+\epsilon) < g(P_2)$. If T is very large, an acceleration anomaly will occur.

4. GENERALIZED HEURISTIC SEARCHES

Recall that the selection function uses the heuristic values to define the order of node expansions. In this section we show that detrimental anomalies are caused by the ambiguity in the selection rule. A generalized heuristic search is proposed to eliminate detrimental anomalies in a single subproblem list.

Consider the serial depth-first search. The subproblems are maintained in a last-in-first-out list, and the subproblem with the maximum level number is expanded first. When multiple subproblems have identical level numbers (heuristic values), the subproblem chosen by the selection function depends on the order of insertion into the stack. Suppose the rightmost son is always expanded and inserted first. Then the leftmost son will be the subproblem inserted last and expanded first in the next iteration.

In a parallel depth-first search with a single subproblem list, the mere extension of the serial algorithm may cause an anomalous behavior. For example, the order of expansion in a serial depth-first search for the tree in Figure 3 is A, B, D, I, J, E, etc. When two processors are used, nodes B and C are expanded in the second iteration that result in nodes D, E, F,

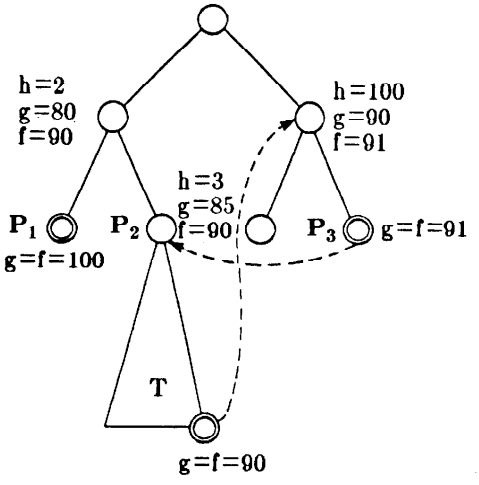

Figure 2. Example of an acceleration anomaly under a parallel depth-first search (ϵ=0.1).

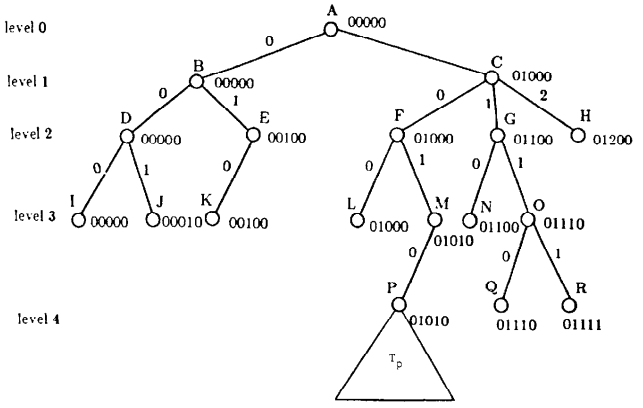

Figure 3. The path numbers of a tree.

G and H. Since these nodes have identical level numbers, any two of these nodes can be chosen for expansion in the next iteration by the conventional heuristic function discussed in Section 2. Suppose the nodes are inserted in the order E, D, H, G and F. Then nodes F and G will be selected and expanded in the third iteration. This may cause a detrimental anomaly if subtree T_p is large. In fact, this is exactly the reason for the anomalies reported by Lai and Sahni [5].

To solve this problem, we must define distinct heuristic values for the nodes so that there is no ambiguity on the nodes to be chosen by the selection function. In this paper a path number is used to uniquely identify a node in a tree. The *path number* of a node is a sequence of d+1 integers that represent the path from the root to this node where d is the maximum number of levels of the tree. The path number $E = e_0e_1e_2...e_d$ is defined recursively as follows. The root P_0 exists at level 0 and has a path number of 000...0. A node P_{i_j} on level ℓ which is the j-th son (counting from the left) of P_i with path number $E_{P_i} = e_0e_1...e_{\ell-1}000...$ has path number $E_{P_{i_j}} = e_0e_1...e_{\ell-1}j00...$. As an example, the path numbers for the nodes in the tree of Figure 3 are shown.

To compare path numbers, the relations ">" and "=" must be defined. A path number $E_1 = e_1^1e_2^1 \cdots$ is less than another path number $E_2 = e_1^2e_2^2 \cdots$ $(E_1 < E_2)$ if there exists $0 \le j \le d$ such that $e_i^1 = e_i^2$, $0 \le i < j$, and $e_j^1 < e_j^2$. The path numbers are equal if $e_i^1 = e_i^2$ for $0 \le i \le d$. For example, the path number 01000 is less than 01010. Note that nodes can have equal path numbers if they have the ancestor-descendant relationship. Since these nodes never coexist simultaneously in the list of active subproblems, the subproblems in the active list always have distinct path numbers.

The path number is now included in the heuristic function. The primary key is still the lower-bound value or the level number. The secondary or ternary key is the path number and is used to break ties in the primary key.

$$h(P_i)=\begin{cases}\text{(level number, path number)} & \text{breadth-first search}\\ \text{(path number)} & \text{depth-first search}\\ \text{(lower bound, level number, path number)} & \\ \text{or (lower bound, path number)} & \text{best-first search}\end{cases} \quad (4)$$

For a best-first search, two alternatives are defined that search in a breadth-first or depth-first fashion for nodes with identical lower bounds. The heuristic functions defined above belong to a general class of heuristic functions that satisfy the following properties:

(a) $h(P_i) \neq h(P_j)$ if $P_i \neq P_j$, $P_i, P_j \in \mathbf{U}$
(all heuristic values in the active list are distinct) (5)

(b) $h(P_i) \le h(P_j)$ if P_j is a descendant of P_i
(heuristic values do not decrease) (6)

In general, any heuristic function with a tie-breaking rule that satisfy Eq's 5 and 6 will not lead to detrimental anomalies. Due to space limitation, the results are stated without proof in the following theorems. The proofs can be found in [7].

Theorem 1: Let $\epsilon=0$, i.e., an exact optimal solution is sought. $T^c(k,0) \le T^c(1,0)$ holds for parallel heuristic searches of a single optimal solution in a centralized list using any heuristic function that satisfies Eq's 5 and 6.

When approximations are allowed, detrimental anomalies cannot always be avoided for depth-first searches even though path numbers or other tie-breaking rules are used (see Figure 1). The reason for the anomaly is that lower-bound tests under approximation, **L**, are not transitive. That is, $P_i \mathbf{L} P_j$ and $P_j \mathbf{L} P_k$ do not imply $P_i \mathbf{L} P_k$, since $f(P_i)/(1+\epsilon) \le g(P_j)$ and $f(P_j)/(1+\epsilon) \le g(P_k)$ implies $f(P_i)/(1+\epsilon)^2 \le g(P_k)$ rather than $f(P_i)/(1+\epsilon) \le g(P_k)$. In this case detrimental anomalies can be avoided for best-first or breadth-first searches only.

Theorem 2: $T^c(k,\epsilon) \le T^c(1,\epsilon)$, $\epsilon>0$, holds for parallel best-first or breadth-first searches for a single optimal solution when a heuristic function satisfying Eq's 5 and 6 is used.

Since the lower-bound function is used as the heuristic function in best-first searches, Eq's 5 and 6 are automatically satisfied if all the lower-bound values are distinct. Otherwise, path numbers must be used to break ties in the lower bounds. In Section 7 a more general condition will be given for best-first searches. For depth-first searches, the conditions of Theorem 2 are not sufficient, and the following condition is needed. For any feasible solution P_i, all nodes whose heuristic values are less than $h(P_i)$ cannot be eliminated by the lower-bound test due to P_i, that is, $f(P_i)/(1+\epsilon) \le g(P_j)$ implies that $h(P_i) < h(P_j)$ for any P_j. Generally, this condition is too strong and cannot be satisfied in practice.

5. NECESSARY CONDITIONS TO ENSURE ACCELERATION ANOMALIES IN A SINGLE SUBPROBLEM LIST

When an exact optimal solution is sought, acceleration anomalies may occur if a depth-first search is used or some nodes have identical heuristic values. This is characterized by the incomplete consistency between the heuristic and the lower-bound functions. A heuristic function, h, is said to be *not completely consistent* with g if there exist two nodes P_i and P_j such that $h(P_i) > h(P_j)$ and $g(P_i) \le g(P_j)$.

Theorem 3: Let $\epsilon = 0$. Assume that a single optimal solution is sought. The necessary condition for $T^c(k,0) < T^c(1,0)/k$ is that the heuristic function is not completely consistent with g.

For a breadth-first search, no acceleration anomaly will occur if the heuristic function defined in Eq. 4 is used. For a best-first search, acceleration anomalies may exist if the level number is not used in the heuristic function. It is important to note that the condition in Theorem 3 is not necessary when approximate solutions are sought. An example showing the existence of an acceleration anomaly when h is completely consistent with g is shown in Figure 2. A looser necessary condition is that h is not completely consistent with the lower-bound test with approximation, that is, there exist P_i and P_j such that $h(P_i) > h(P_j)$ and $P_i \mathbf{L} P_j$.

6. MULTIPLE SUBPROBLEM LISTS

When there are multiple subproblem lists, one for each processor, a node with the minimum heuristic value is selected for decomposition from each local list. This node may not belong to the global set of active nodes with the minimum heuristic values; however, the node with the minimum heuristic value will always be expanded by a processor as long as the nodes are selected in a consistent order when there are ties. Since it is easy to maintain the incumbent in a global data register, the behavior of multiple lists is analogous to that of a centralized list. However, the performance of using multiple lists is usually worse than that of a single subproblem list [10].

So far, we have shown conditions to avoid detrimental anomalies and to preserve acceleration anomalies under lower-bound tests only. The results are summarized in Table 1. The corresponding results when dominance tests are used will not be shown here due to space limitation [7].

Table 1. Summary of results for the elimination of detrimental anomalies and the preservation of acceleration anomalies in parallel B&B algorithms with lower-bound tests.

Allowance parameter	Sub-problem lists	Search strategies	Suff. cond. to eliminate Detrimental Anomaly	Necessary cond. for Acceleration Anomaly
$\epsilon=0$	single	all	I	II
	multiple		no anomaly	
$\epsilon>0$	single or multiple	breadth-first or best-first	I	exists
		depth-first	anomaly	

Conditions: I: heuristic function satisfies Eq's 5 and 6.
II: h is not completely consistent with g.
anomaly: the sufficient conditions are impractical.
exists: the necessary conditions are too loose.

7. ROBUSTNESS OF PARALLEL BEST-FIRST SEARCHES

The preceding sections have shown that best-first searches are more robust for parallel processing in the sense of avoiding detrimental anomalies and preserving acceleration anomalies. In this section we shown that best-first searches are more robust as far as deceleration anomalies are concerned.

Figure 4 shows the computational efficiency of a parallel optimal B&B algorithm using a best-first or a depth-first search for solving knapsack problems in which the weights, w(i), are chosen randomly between 0 and 100 and the profits are set to be p(i)=w(i)+10. The assignment used is intended to increase the complexity of the problem. In the simulations each processor has a local memory. Load balancing is incorporated so that an idle processor with an empty subproblem list can get a subproblem from its neighbor. It is observed that the speedup is sensitive to T(1,0), and the speedup is better for best-first searches. For instance, when 64 processors are used, the average speedup is 48.8 for best-first searches and 27.9 for depth-first searches. Moreover, it should be noted that the generalized heuristic search presented in Section 4 cannot guarantee $T(k_1,0) < T(k_2,0)$, $k_1 > k_2 > 1$, for depth-first and breadth-first searches. Similar results were observed for vertex-cover problems.

The following theorem gives the performance bound of parallel best-first searches. The maximum number of processors within which a near-linear speedup is guaranteed can be predicted.

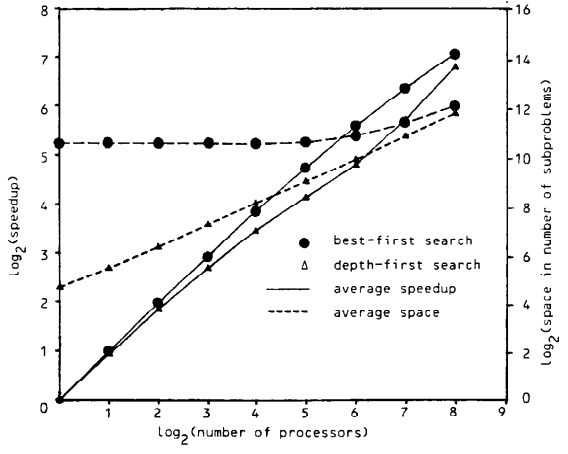

Figure 4. Average speedups and space requirements of parallel optimal B&B algorithms for 10 knapsack problems with 35 objects (average T(1,0)=15180 for best-first searches; average T(1,0)=15197 for depth-first searches).

Theorem 4: For a parallel best-first search with k processors, $\epsilon=0$, and $g(P_i)\neq f^*$ if P_i is not an optimal-solution node (f^* is the optimal-solution value),

$$\frac{T(1,0)}{k} \leq T(k,0) \leq \frac{T(1,0)}{k} + \frac{k-1}{k}\ell \qquad (7)$$

where ℓ is the maximum number of levels of the B&B tree to be searched. Since the performance is not affected by using single or multiple subproblem lists, the superscript in T is dropped.

Since ℓ is a polynomial function of (usually equal to) the problem size while T(1,0) is an exponential function of the problem size for NP-hard problems, the first term on the R.H.S. of Eq. 7 is much greater than the second term as long as the problem size is large enough. Eq. 7 implies that the near-linear speedup can be maintained within a considerable range of the number of processors for best-first searches. As an example, if $\ell=50$, $T(1,0)=10^6$ (for a typical traveling-salesman problem), and k=1000, then $T(1000,0)\leq 1049$. This means that almost linear speedup can be attained with 1000 processors. Furthermore, it can be shown that there is always a monotonic increase in performance for all $1 \leq k_1 < k_2 \leq \sqrt{T(1,0)/\ell}$. For this example, there will not be any detrimental anomaly for any combinations of $1 \leq k_1 < k_2 \leq 141$ if the assumptions of Theorem 4 are satisfied.

Before ending this paper, it is worth saying a few words about the space required by parallel B&B algorithms. In the serial case, the space required by a best-first search is usually more than that required by a depth-first search. Somewhat surprisingly, the simulation results on 0-1 knapsack problems show that the space required by parallel best-first searches is not increased significantly (may also be decreased) until the number of processors is so large that a near-linear speedup is not possible. In contrast, the space required by parallel depth-first searches is almost proportional to the number of processors (Figure 4). Note that the space efficiency is problem-dependent. For vertex-cover problems, the space required by parallel best-first searches is not increased significantly regardless of the number of processors used.

REFERENCES

[1] Finkel, R., "Parallelism in Alpha-Beta Search," *Artificial Intelligence*, (1982) 89-106.

[2] Ibaraki, T., "Theoretical Comparisons of Search Strategies in Branch-and-Bound Algorithms," *Int'l Jr. of Comp. and Info. Sci.*, 5:4 (1976) 315-344.

[3] Imai, M., T. Fukumura and Y. Yoshida, "A Parallelized Branch-and-Bound Algorithm Implementation and Efficiency," *Systems, Computers, Controls*, 10:3 (1979) 62-70.

[4] Kumar, V., and L. Kanal, "A General Branch-and-Bound Formulation for understanding and synthesizing AND/OR Tree-search Procedures," *Artificial Intelligence*, 14 (1983) 179-197.

[5] Lai, T.H. and S. Sahni, "Anomalies in Parallel Branch-and-Bound Algorithms," in *Proc. 1983 Int'l Conf. on Parallel Processing*, Bellaire, Michigan, Aug. 1983, pp. 183-190.

[6] Lawler, E. L., and D. W. Wood, "Branch-and-Bound Methods: A Survey," *Operations Research*, 14 (1966) 699-719.

[7] Li, G.-J., and B. W. Wah, "Computational Efficiency of Parallel Approximate Branch-and-Bound Algorithms," Tech. Report TR-84-6, School of Electrical Engineering, Purdue University, West Lafayette, Indiana, March 1984; a shorter version appears in *Proc. 1984 Int'l Conf. on Parallel Processing*, Bellaire, Michigan, Aug. 1984.

[8] Pearl, J., *Heuristics*, Addison-Wesley, 1984.

[9] Smith, D. R., "Random Trees and the Analysis of Branch-and-Bound Procedures," *Journal of the ACM*, 31:1 (1984) 163-188.

[10] Wah, B. W. and Y. W. Ma, "MANIP - A Multicomputer Architecture for solving Combinatorial Extremum-Search Problems," *IEEE Trans. on Computers*, C-33:5, (1984) 377-390.