Coping with Anomalies in Parallel Branch-and-Bound Algorithms

GUO-JIE LI AND BENJAMIN W. WAH

Abstract — **A general technique that can be used to solve a wide variety of discrete optimization problems is the branch-and-bound algorithm. We have adapted and extended branch-and-bound algorithms for parallel processing. The computational efficiency of these algorithms depends on the allowance function, the data structure, and the search strategies. Anomalies owing to parallelism may occur. In this correspondence, anomalies of parallel branch-and-bound algorithms using the same search strategy as the corresponding serial algorithms are studied. Sufficient conditions to guarantee no degradation in performance due to parallelism and necessary conditions for allowing parallelism to have a speedup greater than the number of processors are presented.**

Index Terms — **Anomalies, approximation, branch-and-bound algorithms, heuristic search, lower-bound test, parallel processing.**

I. Introduction

The search for solutions in a combinatorially large problem space is a major problem in artificial intelligence and operations research. Generally speaking, these 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 and evaluating queries for expert systems belong to this class. The second type is combinatorial extremum-search or optimization problems that are characterized by an objective function to be optimized and a set of constraints to be satisfied. Practical problems such as the traveling-salesman, warehouse-location, job-shop-scheduling, knapsack, vertex-cover, and integer-programming problems are examples in this class.

Exhaustive search is usually impractical and prohibitively expensive for solving large search problems, especially when the problem is NP-hard [1]. Studies on improving search efficiency is, thus, of considerable importance. Research has been conducted on designing a unified method for a wide variety of problems [5], [3], [16], the most general of which is the *branch-and-bound* (B&B) algorithm. This is a partitioning algorithm that decomposes a problem into smaller disjunctive subproblems and repeatedly decomposes until infeasibility is proved or a solution is found [8], [12]. Backtracking, dynamic programming [14], and AND–OR tree search [6] can be viewed as variations of B&B algorithms.

In implementing search algorithms for combinatorial searches, approximations and parallel processing are two major approaches to enhance their efficiency. Owing to the exponential nature of NP-hard problems, optimal solutions are usually infeasible to obtain. In practice, approximate solutions are acceptable alternatives. Our experimental results on the vertex-cover and 0-1 knapsack problems reveal that a linear reduction in accuracy results in an exponential reduction of the average computational time [22]. On the other hand, parallel processing is applicable when the problem is solvable in polynomial time, or when the problem is NP-hard but is solvable in polynomial time on the average [17], or the problem is heuristically solvable in polynomial time (such as game trees) [21]. It is impractical to use parallel processing to solve a problem with an exponential complexity because, in the worse case, an exponential number of processors must be used to solve the problem in polynomial time.

Manuscript received March 15, 1984; revised July 1, 1985. This work was supported by the National Science Foundation under Grant ECS81-05968. A more comprehensive version of this correspondence was presented at the 1984 International Conference on Parallel Processing.

The authors were with the School of Electrical Engineering, Purdue University, West Lafayette, IN. They are now with the Coordinated Science Laboratory and the Department of Electrical and Computer Engineering, University of Illinois, Urbana, IL 61801.

IEEE Log Number 8407306.

Analytical properties of *parallel approximate branch-and-bound* (PABB) algorithms have been rarely studied. In general, a k-fold speedup (or ratio of the execution time in the serial case to that of the parallel case) is expected when k processors are used. However, simulations have shown that the speedup for PABB algorithms using k processors can be: a) less than one (called a *detrimental anomaly*) [4], [7], [13]; b) greater than k (called an *acceleration anomaly*) [4], [7]; or c) between one and k (called a *deceleration anomaly*) [4], [18], [7], [13], [20]. It is desirable to discover conditions that preserve the acceleration anomalies, eliminate the detrimental anomalies, and minimize the deceleration anomalies.

One of the motivations of this correspondence is to improve Lai and Sahni's results on anomalies of parallel B&B algorithms [7]. First, Lai and Sahni have made an implicit assumption that all nonterminal nodes of the B&B tree have identical lower bounds, and hence the nonterminal nodes can be expanded in any order. Nearly all of their anomalies were discovered with an inconsistent selection strategy, namely, when n_1 processors were used, the expansion order of nonterminal nodes in a level was from left to right, but when n_2 processors were available, the expansion order was from right to left. We will show in Section III that all of their detrimental anomalies can be avoided by using a consistent selection order. Second, Lai and Sahni have only considered best-first searches and finding optimal solutions. However, our theoretical analysis and simulations have shown that anomalies are infrequent when optimal solutions are sought using best-first searches, while they are more frequent in approximate B&B algorithms with depth-first searches [9]. We will prove conditions that can avoid detrimental anomalies even when approximations are allowed. Since the anomalous behavior depends on the search strategies, we will investigate the anomalies with respect to the various search strategies separately. Lastly, Lai and Sahni have claimed that a near-linear speedup for parallel B&B algorithms with best-first search holds only for a "small" number of processors. On the contrary, we have shown that a near-linear speedup may hold for a large number of processors, and that the maximum number of processors to attain a near-linear speedup can be predicted [9].

The objective of this correspondence is to study conditions to cope with the anomalous behavior of B&B algorithms under approximations and parallel processing. Anomalies between the serial and parallel cases are studied with respect to the same search strategy. In general, anomalies should be studied with respect to the best serial algorithm and the best parallel algorithm (with possibly a different search strategy than that of the serial algorithm). However, conditions to resolve these anomalies would be problem dependent and may result in a large number of cases that cannot be enumerated. The conditions to resolve anomalies between n_1 and n_2 processors, $1 \le n_1 < n_2$, are different from those presented here and are shown elsewhere [11]. These results on resolving anomalies are useful for designers to understand the existence of anomalies and to modify existing algorithms to prevent detrimental anomalies and enhance acceleration anomalies.

II. Parallel Approximate Branch-and-Bound Algorithms

Many theoretical properties of B&B algorithms have been developed by Kohler and Steiglitz [5] and Ibaraki [2], [3]. A brief summary of these properties to solve a minimization problem were discussed elsewhere [19], and only the important properties that will be used in the following sections are stated here.

B&B algorithms 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 problems into simpler subproblems and to appropriately order the search. The last two rules are used to eliminate generated subproblems that are not better than the ones already known. Appropriately ordering the search and restricting the region searched are the key ideas behind B&B algorithms.

The way in which P_0, the initial problem, is repeatedly decomposed into smaller subproblems can be represented as a finite rooted tree $\mathbf{B} = (\mathbf{P}, \mathbf{E})$ where $\mathbf{P}$ is a set of disjunctive subproblems, and $\mathbf{E}$ is a set of edges. The root of the tree is P_0. If a subproblem P_{i_j} is obtained from P_i by direct decomposition, then $(P_i, P_{i_j}) \in \mathbf{E}$. The *level number* of a node is the number of edges leading from the root to this node (the root is at Level 0). Let $f(P_i)$ be the value of the best solution obtained by evaluating all subproblems decomposable from P_i, P_{i_j} be the jth subproblem directly decomposable from P_i, and k_i be the total number of such subproblems ($k_i = |\{(P_i, x): (P_i, x) \in \mathbf{E}\}|$). Then f satisfies

$$f(P_i) = \min_{j=1,\cdots,k_i} \{f(P_{i_j})\}. \tag{1}$$

Each subproblem is characterized by a lower bound value that is computed from a lower bound function g. Let $\mathbf{T}$ be the set of all feasible solutions. The lower bound function satisfies the following properties:

$$1)\ g(P_i) \le f(P_i) \quad \text{for } P_i \in \mathbf{P} \quad (g \text{ is a lower bound estimate of } f); \tag{2}$$

$$2)\ g(P_i) = f(P_i) \quad \text{for } P_i \in \mathbf{T} \quad (g \text{ is exact when } P_i \text{ is feasible}); \tag{3}$$

$$3)\ g(P_i) \le g(P_{i_j}) \quad \text{for } (P_i, P_{i_j}) \in \mathbf{E} \quad (\text{lower bounds of descendents always increase}). \tag{4}$$

If a subproblem is a feasible solution with the best objective-function value so far, then the solution value becomes the *incumbent* z. The incumbent represents the best solution obtained so far in the process. Let $\mathbf{L}$ be the lower bound cutoff test. If a single solution is sought, the $P_j \mathbf{L} P_i$ means that P_j is a feasible solution and that $f(P_j) \le g(P_i)$. Using the best solution value obtained, P_i is terminated during the computation if

$$g(P_i) \ge z. \tag{5}$$

An approximate B&B algorithm is identical to the optimal B&B algorithm except that the lower bound test is modified. Hence, $P_j \mathbf{L} P_i$ if $f(P_j)/(1 + \epsilon) \le g(P_i)$, $\epsilon \ge 0$ where ϵ is an *allowance function* specifying the allowable deviation of a suboptimal solution value from the optimal solution value. Using the incumbent, P_i is terminated during the evaluation of the approximate B&B algorithm if

$$g(P_i) \ge \frac{z}{1+\epsilon} \quad \epsilon \ge 0,\ z \ge 0. \tag{6}$$

The final incumbent value z_F obtained by the modified lower bound test deviates from the optimal solution value z_O by

$$\frac{z_F}{1+\epsilon} \le z_O \le z_F. \tag{7}$$

For example, suppose it were decided that a deviation of 10 percent from the optimum was tolerable. If a feasible solution of 150 is obtained, all subproblems with lower bounds of 136.4 (or 150/(1 + 0.1)) or more can be terminated, since they cannot lead to a solution that deviates by more than 10 percent from 150. This technique significantly reduces the amount of intermediate storage and time needed to arrive at a suboptimal solution.

Ibaraki has mapped breadth-first, depth-first, and best-first searches into a general form called the *heuristic searches* [3], [15]. A heuristic function is defined to govern the order in which subproblems 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, and 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, and the level number can be treated as the heuristic function. Lastly, in a depth-first search, subproblems with the maximum level numbers are expanded first, and the negation of the level number can be taken as the heuristic function. If $\mathbf{U}$ is the current list of active subproblems in the process of expansion and h is the heuristic function, then the selection function for a subproblem to be expanded in a serial B&B algorithm is

$$S_s(\mathbf{U}) = \{P_{ji} \mid h(P_i) = \min_{P_j \in U}(h(P_j))\}. \tag{8}$$

B&B algorithms have inherent parallelism. Each of the four rules of a serial B&B algorithm can be enhanced by parallel processing.

1) Parallel Selection of Subproblems: In the parallel case, a set of subproblems less than or equal in size to the number of processors have to be selected for decomposition in each iteration. The selection problem is especially critical under a best-first search because a set of subproblems with the smallest lower bounds must be selected. The selection function in (8) becomes

$$s_P(\mathbf{U}) = \begin{cases} \{\mathbf{P}_{i_1}, \cdots, \mathbf{P}_{i_k}\} & \text{if } |\mathbf{U}| > k \\ \text{where } h(P_i) < h(P_j),\ P_i, P_j \in \mathbf{U} & \\ i \in \{i_1, \cdots, i_k\},\ j \notin \{i_1, \cdots, i_k\} & \\ \mathbf{U} & \text{if } |\mathbf{U}| \le k \end{cases}$$

where k is the number of processors. This returns the set of k (or less) subproblems with the smallest heuristic values from $\mathbf{U}$.

2) Parallel Branching: The selected subproblems can be decomposed in parallel.

3) Parallel Termination Tests: Multiple infeasible nodes can be eliminated in each iteration. Further, multiple feasible solutions may be generated in an iteration, and the incumbent has to be updated in parallel to find a new incumbent.

4) Parallel Elimination Tests: The lower bound test [(5) or (6)] can be carried out in parallel by comparing lower bounds of multiple subproblems with the incumbent. However, the bounding functions are problem-dependent, and software implementation may be more flexible.

For simplicity, the case of searching a single optimal (or suboptimal) solution is discussed in this correspondence. The case in which all solutions are sought can be analyzed similarly. The parallel computational model used here consists of a set of processors connected to a shared memory. There is a single subproblem list shared by all processors. It is assumed that the processors operate synchronously in executing the steps of the PABB algorithm. In each iteration of the algorithm, multiple subproblems are selected and decomposed. The newly generated subproblems are tested for feasibility (and the incumbent updated if necessary), eliminated by lower bound tests, and inserted into the active list if not eliminated. In this model, eliminations are performed after branching instead of after selection as in Ibaraki's algorithm [2]. This reduces the memory space required for storing the active subproblems.

The shared memory in the proposed computational model may seem to be a bottleneck of the system. However, this model can be transformed into a second model in which all processors have a private memory and are connected by a ring network. The behavior of the PABB algorithm in the transformed model has been shown to be very close to that of the original model, except that very little interprocessor communication and interference are involved [19], [20]. Since subproblems are decomposed synchronously and the bulk of the overhead is on branching operations, the number of iterations, which is the number of times that subproblems are decomposed in each processor, is an adequate measure in both the serial and parallel models. The *speedup* is thus measured by the ratio of number of iterations in the serial case to that of the parallel case.

III. Anomalies of PABB Algorithms

In this section, some anomalies of the PABB algorithm are illustrated. Let $T(k, \epsilon)$ be the number of iterations required for expanding a B&B tree to find the first optimal (or suboptimal) solution, where k is the number of processors and ϵ is the allowance function. Once the optimal solution is found, the time to drain the remaining subproblems from the active list is not accounted for here.

Fig. 1 shows an example of a detrimental anomaly when approximations are allowed. In a serial depth-first search, subtree T_2 is terminated owing to the lower bound test of P_1': $f(P_1')/(1+\epsilon) \le g(P_2)$ where $\epsilon = 0.1$. In a parallel depth-first search with two processors, a feasible solution P_4 is found first, and nodes P_1 and P_1' are terminated owing to the lower bound test of P_4. Consequently, subtree T_2 has to be expanded, which will eventually prune subtree T_3. If the size of T_2 is much larger than that of T_3, the time taken to expand T_2 using two processors will be longer than that taken to expand T_3 using one processor. Strategies to handle these anomalies will be discussed in Section V-B.

Fig. 2 shows an example of an acceleration anomaly when a depth-first search with approximations is used. When a single processor is used, subtree T has to be expanded. However, when two processors are used, P_2 is expanded in the second iteration, and the feasible solution P_4 is found. Therefore, node P_3 and subtree T will be eliminated by lower-bound tests with P_4. If subtree T is large, then the speedup of using two processors over one processor will be much greater than two. Acceleration anomalies will be discussed in Section VI.

Many examples to illustrate anomalies can be created for the various combinations of search strategies and allowance functions. However, the important consideration here is *not in knowing that anomalies exist, but in understanding why these anomalies occur.* Furthermore, it is desirable to preserve the acceleration anomalies and to avoid the detrimental anomalies. Our objective is to find the sufficient conditions to ensure that $T(k, \epsilon) \le T(1, \epsilon)$, as well as the necessary conditions for $T(k, \epsilon) \le T(1, \epsilon)/k$. The necessary conditions to eliminate detrimental anomalies and the sufficient conditions to preserve acceleration anomalies are not evaluated because they depend on the sequence of nodes expanded and the size of the resulting subtrees. Besides being impossible to enumerate due to the large number of possible combinations, these conditions are problem dependent and cannot be generalized to all problems.

IV. Generalized Heuristic Searches

Recall from Section II that the selection function uses heuristic values to define the order of node expansions. It was mentioned that breadth-first, depth-first, and best-first searches are special cases of heuristic searches. These searches are potentially anomalous when parallel expansions are allowed.

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 (or heuristic values), the node chosen by the selection function depends on the order of insertion into the stack. If the rightmost child of a parent node is inserted first, then the leftmost child will be the node inserted last and expanded first in the next iteration.

In a parallel depth-first search, 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 Fig. 3 is A, B, D, I, J, E, etc. When two processors are used, nodes B and C are decomposed to nodes D, E, F, G, and H in the second iteration. Since these nodes have identical level numbers, the selection function can choose any two of these nodes in the next iteration. Suppose that they 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 an unexpected behavior as compared to the serial case. A similar example can also be developed for the best-first search when the lower bounds of nodes are identical.

The ambiguous selection of nodes for expansion is exactly the reason for anomalies reported by Lai and Sahni [7]. In their proof of Theorem 1, which states that detrimental anomalies can always exist when a larger number of processors are used, the nodes selected for expansion are different when a different number of processors are used. This change of selection order in their Theorem 1

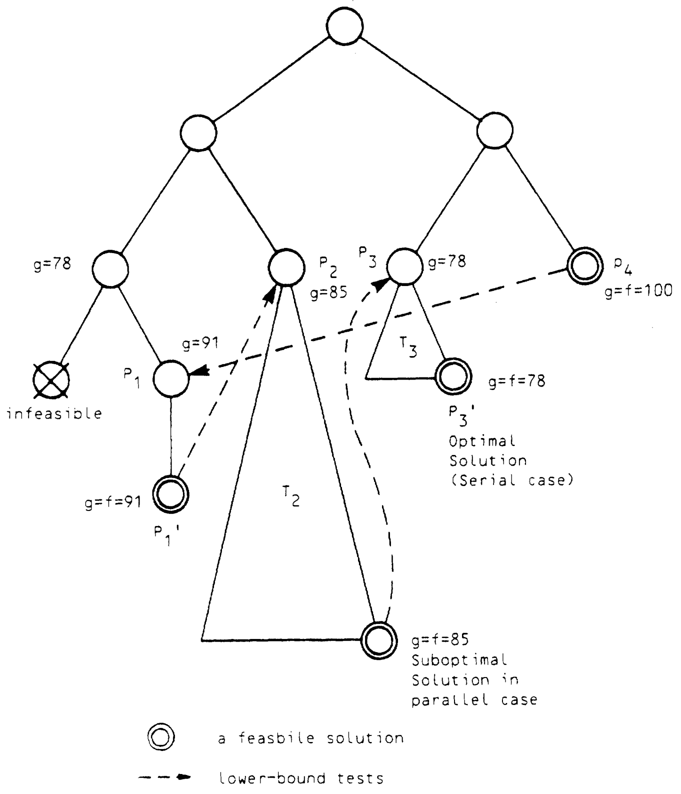

Fig. 1. An example of a detrimental anomaly under a depth-first search with approximations ($\epsilon = 0.1$).

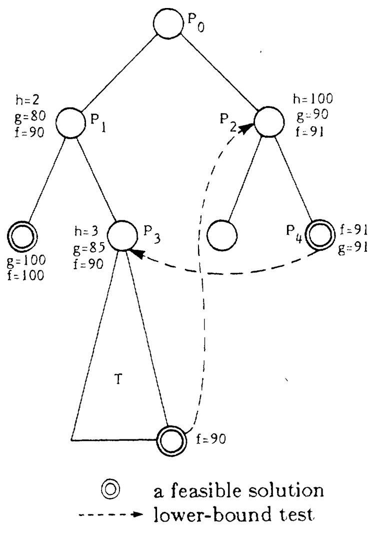

Fig. 2. An example of an acceleration anomaly under a depth-first or best-first search with approximations ($\epsilon = 0.1$).

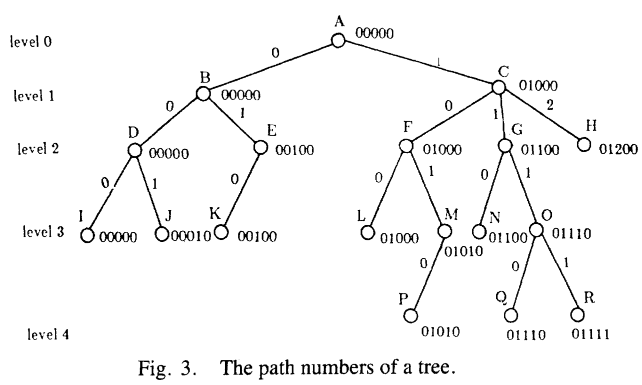

Fig. 3. The path numbers of a tree.

(and almost all their other theorems) is based on the implicit assumption that nodes have identical lower bounds. In this case, the lower bounds are not useful in guiding the selection of subproblems. To have an accurate comparison when different number of processors are used, a consistent selection strategy must be used.

To resolve the ambiguity of the selection of subproblems, distinct heuristic values must be defined for the nodes. In this correspondence, a path number is proposed to uniquely identify a node. The *path number* of a node in a tree is a sequence of $d + 1$ integers representing the path from the root to this node, where d is the maximum number of levels of the tree. The path number $E = e_0e_1e_2 \cdots e_d$ is defined, recursively, as follows. The root P_0 exists at Level 0 and has a path number of $E_0 = 000 \cdots 0$. A node P_{i_j} on Level l, which is the jth child (counting from the left) of P_i with path number $E_i = e_0e_1 \cdots e_{l-1}000 \cdots$, has path number $E_{i_j} = e_0e_1 \cdots e_{l-1}j00 \cdots$. As an example, the path numbers of all nodes in the tree of Fig. 3 are shown outside the nodes.

To compare path numbers, the relations '>' and '=' must be defined. A path number $E_x = e_1^1e_2^1 \cdots$ is less than another path number $E_y = e_1^2e_2^2 \cdots$ ($E_x < E_y$) 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. According to our definition of path numbers, nodes can have equal path numbers if they have the ancestor–descendant relationship. Since these nodes never coexist simultaneously in the active list of subproblems of a B&B algorithm, the subproblems in the active list always have unique 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.

$$h(P_i) = \begin{cases} \text{(level number, path number)} & \text{breadth-first search} \\ \text{(path number)} & \text{depth-first search} \\ \text{(lower bound, level number,} & \\ \text{ path number) or} & \\ \text{(lower bound, path number)} & \text{best-first search} \end{cases} \quad (10)$$

where the level number, path number, and lower bound are defined for P_i. For a best-first search, nodes with identical lower bounds can be searched in a breadth-first or depth-first fashion.

The heuristic functions defined above belong to a general class of heuristic functions satisfying the following properties:

$$h(P_i) \ne h(P_j) \quad \text{if } P_i \ne P_j,\ P_i, P_j \in \mathbf{P} \quad \text{(all heuristic values are distinct)} \quad (11)$$

$$h(P_i) \le h(P_j) \quad \text{if } P_j \text{ is a descendant of } P_i \quad \text{(heuristic values do not decrease)}. \quad (12)$$

In general, the results developed in the following sections are applicable to any unambiguous heuristic function satisfying (11) and (12), and are not restricted to the use of path numbers. For example, the lower bound can be used as the secondary key and the path number as the ternary key in a breadth-first search.

By using the heuristic function defined in (10), the study of anomalies for depth-first, best-first, and breadth-first searches can be unified. The results developed in the following sections apply to all these search strategies.

V. Sufficient Conditions to Eliminate Detrimental Anomalies

A. Parallel B&B Algorithms with Lower Bound Tests Only

In this section, we show that any heuristic search with an unambiguous heuristic function can guarantee that $T(k, \epsilon) \le T(1, \epsilon)$ when only lower bound tests with $\epsilon = 0$ are used. A *basic node* is the node with the smallest heuristic value in each iteration. Let Φ^1 and Φ^k be the sets of nodes expanded in the B&B tree using one and k processors, respectively. To show that $T(k, 0) \le T(1, 0)$, it suffices to prove a) that at least one node belonging to Φ^1 is expanded in each iteration of the parallel search; and b) that once all the nodes in Φ^1 are expanded or terminated, the parallel heuristic search must terminate. The proof requires the following property on basic nodes.

Lemma 1: Let P_i be a basic node, then for any node P_j such that $h(P_j) < h(P_i)$, P_j must be either expanded or terminated when P_i is expanded.

Proof: Suppose that in the current active list, $\mathbf{U}$, $P_i \in \mathbf{U}$ is a basic node. Assume that there exists a node P_j such that $h(P_j) < h(P_i)$ and that P_j has not been expanded or terminated when P_i is expanded. Since P_i has the minimum heuristic value among the active nodes in $\mathbf{U}$, P_j must not be active at that time. That is, P_j is a descendant of some node P_k, $P_k \in \mathbf{U}$, and $h(P_i) < h(P_k)$. By (12), $h(P_i) < h(P_k) \le h(P_j)$, which contradicts the assumption that $h(P_j) < h(P_i)$. □

The following theorem proves that any unambiguous heuristic functions satisfying (11) and (12) are sufficient to eliminate detrimental anomalies.

Theorem 1: Let $\epsilon = 0$, i.e., an exact optimal solution is sought. $T(k, 0) \le T(1, 0)$ holds for any parallel heuristic search with a heuristic function satisfying (11) and (12).

Proof: The proof is by contradiction. Suppose there exists a basic node P_{i_1} in the parallel search such that $P_{i_1} \notin \Phi^1$ and that $P_{i_1} \in \Phi^k$ (see Fig. 4). This means that either P_{i_1} or its an ancestor is terminated by a lower bound test in the serial case. Hence, there must exist a feasible solution $P_{i_2} \in \Phi^1$ such that $f(P_{i_2}) \le g(P_{i_1})$ and that P_{i_2} has not been obtained when P_{i_1} is expanded in the parallel case. It implies that a proper ancestor $P_{i_3} \in \Phi^1$ of P_{i_2} exists in the serial case such that $h(P_{i_3}) \le h(P_{i_1})$, and that P_{i_2} is obtained before P_{i_1} and terminates P_{i_1}. Since P_{i_1} is a basic node in the parallel search, $h(P_{i_3}) < h(P_{i_1})$, and P_{i_3} has not been expanded when P_{i_1} is expanded in the parallel case, P_{i_3} must be terminated according to Lemma 1. For the parallel search, there must exist a feasible solution $P_{i_4} \in \Phi^k$ such that $f(P_{i_4}) \le g(P_{i_3})$, and that P_{i_4} has not been obtained when P_{i_3} is expanded in the serial case. Two cases are possible:

First, P_{i_4} is not generated when P_{i_3} is expanded in the serial case, i.e., a proper ancestor $P_{i_5} \in \Phi^k$ of P_{i_4} exists when P_{i_3} is active and $h(P_{i_5}) > h(P_{i_3})$. According to the properties of lower bound functions [(2), (3), and (4)], we have $f(P_{i_4}) \le g(P_{i_3}) \le f(P_{i_3}) \le f(P_{i_2}) \le g(P_{i_1})$. Moreover, in the parallel case, P_{i_4} should be obtained before P_{i_1} is expanded (otherwise, P_{i_3} would not be terminated by P_{i_4}). Hence P_{i_1} has to be terminated by P_{i_4} in the parallel algorithm, which contradicts the assumption that $P_{i_1} \in \Phi^k$.

Second, $h(P_{i_5}) < h(P_{i_3})$, and P_{i_5}, as well as its descendant P_{i_4}, have been terminated in the serial case and not in the parallel case. We can then apply the above argument again to P_{i_5} and eventually obtain a sequence of nodes $P_{i_1}, P_{i_2}, \cdots, P_{i_m}$ as depicted in Fig. 4, in which P_{i_m} is not terminated by any lower bound test. There are three possibilities:

1) The first node P_{i_m} occurs in the serial case [Fig. 4(a)]. Since $P_{i_{m-1}}$ is a feasible solution, we have: $g(P_{i_m}) \le f(P_{i_{m-1}}) \le g(P_{i_{m-2}}) \le f(P_{i_{m-2}}) \le g(P_{i_{m-3}}) \le \cdots \le f(P_{i_1}) \le g(P_{i_1})$. Further, since $h(P_{i_k+2}) < h(P_{i_k})$ (otherwise, P_{i_k} could not have been terminated by $P_{i_{k+1}}$ in the serial case) and since $h(P_{i_{k+4}}) < h(P_{i_{k+2}})$ (by the same argument as $h(P_{i_5}) < h(P_{i_3})$), we have $h(P_{i_{k+4}}) < h(P_{i_k})$. Repeating this, we get $h(P_{i_m}) < h(P_{i_1})$. By Lemma 1, P_{i_m} must be expanded in the parallel case and terminates P_{i_1}, which contradicts that $P_{i_1} \in \Phi^k$.

2) The first node P_{i_m} occurs in the parallel case, Fig. 4(b). Similar to the argument for P_{i_4} discussed before, we can explain that P_{i_m-1} has been obtained when P_{i_1} is selected. Therefore, P_{i_1} must be terminated in the parallel case, which contradicts that $P_{i_1} \in \Phi^k$.

3) There is a cycle of cutoffs such that $P_{i_m-1}\mathbf{L}P_{i_m-2}$, $P_{i_m-3}\mathbf{L}P_{i_m-4}, \cdots, P_{i_k+1}\mathbf{L}P_{i_k}$, and $P_{i_k-1}\mathbf{L}P_{i_m}$ where $\mathbf{L}$ denotes a lower bound cutoff test [Figure 4(c)]. By transitivity, we have $f(P_{i_m}) \le f(P_{i_{m-1}}) \le f(P_{i_{k-1}}) \le f(P_{i_m})$, which implies that $f(P_{i_m}) = f(P_{i_m-1}) = f(P_{i_k-1})$. The heuristic value of all nodes of the cycle are less than $h(P_{i_1})$, so a feasible solution has been obtained before P_{i_1} is selected. Thus, P_{i_1} must be terminated in the parallel case, which contradicts that $P_{i_1} \in \Phi^k$.

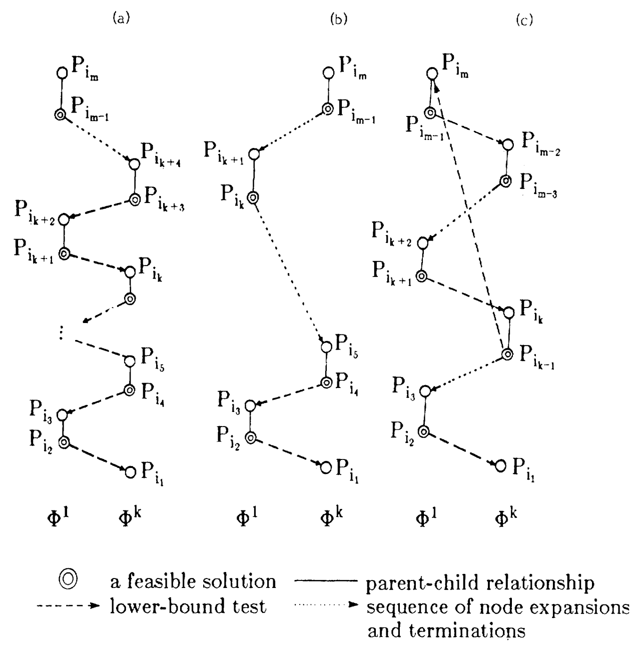
(a)
(b)
(c)
Φ^1 Φ^k
◎ a feasible solution
——— parent-child relationship
- - - -► lower-bound test
·········► sequence of node expansions and terminations

Fig. 4. Proof of Theorem 1.

So far, we have proved that at least one node in Φ^1 is expanded in each iteration of a parallel heuristic search. Since approximation is not allowed, the optimal-solution node cannot be eliminated by lower bound tests. Hence during a parallel heuristic search, once all nodes in Φ^1 are either expanded or terminated, the optimal-solution node must be found. The remaining unexpanded nodes do not belong to Φ^1 because their lower bounds are greater than the optimal solution. The parallel heuristic search is thus completed at this time. □

The above theorem shows that detrimental anomalies can be avoided for depth-first, breadth-first, and best-first searches with $\epsilon = 0$ by augmenting the heuristic function with an unambiguous function. As a special case, for a best-first search in which all nodes have distinct lower bounds, the node with the smallest lower bound can always be selected from the priority queue. In this case, the path numbers do not have to be used, and no detrimental anomaly will occur. Due to space limitation, the performance bounds of the parallel best-first search will not be shown here [9], [11].

B. Parallel B&B Algorithms with Approximations

When parallel approximate B&B algorithms are considered, Theorem 1 is no longer valid all the time (see the example in Fig. 1). The reason for the detrimental anomaly is that $\mathbf{L}$, the lower bound tests under approximation, 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 $f(P_i)/(1+\epsilon) \le g(P_k)$. Somewhat surprisingly, it is possible that Φ^1 and Φ^k are almost disjoint, and most of the nodes in Φ^1 are not expanded in the parallel case. The following theorem shows that detrimental anomalies can be avoided for a best-first search.

Theorem 2: $T(k, \epsilon) \le T(1, \epsilon)$, $\epsilon > 0$, holds for best-first searches if the heuristic function satisfies (11) and (12).

Proof: The key idea of this proof is to show that a detrimental anomaly cannot occur although transitivity of lower bound tests is not valid here. Suppose that there exists a basic node P_{i_1} in Φ^k and not in Φ^1. There are two cases as described in the proof of Theorem 1 (see Fig. 4).

First, assume that $h(P_{i_5}) > h(P_{i_3})$. Since the relations $P_{i_4}\mathbf{L}P_{i_3}$ and $P_{i_2}\mathbf{L}P_{i_1}$ exist, and P_{i_3} is selected before P_{i_1} in the serial case (for a best-first search), it is true that $g(P_{i_3}) \le g(P_{i_1})$. This implies that $f(P_{i_4})/(1+\epsilon) \le g(P_{i_1})$. P_{i_4} can be shown to be available before P_{i_1} is expanded by the same argument as in the proof of Theorem 1. Hence, P_{i_1} must be eliminated by P_{i_4} in the parallel case, a contradiction!

Second, assume that $h(P_{i_5}) < h(P_{i_3})$. The argument here is similar to the proof of Theorem 1 except that the lower bound test to be used is $f(P_i)/(1+\epsilon) \le g(P_j)$ and not $f(P_i) \le g(P_j)$. □

For depth-first searches, the condition of Theorem 2 is not sufficient. Fig. 1 is an example of a detrimental anomaly caused by approximations. In breadth-first searches, since the sequences of feasible solutions with respect to serial and parallel cases are the same, and the minimal feasible solution is selected as the new incumbent if more than one feasible solution are obtained in an iteration of a parallel search, approximations will not result in detrimental anomalies when the condition in Theorem 2 is satisfied.

VI. Necessary Conditions for Acceleration Anomalies

In this section, the necessary conditions for $T(k, 0) < T(1, 0)/k$ are developed. One condition is based on the complete consistency of heuristic functions. A heuristic function h is said to be *consistent* (respectively, *completely consistent*) with the lower bound function g if $h(P_i) < h(P_j)$ implies that $g(P_i) \le g(P_j)$ (respectively, $g(P_i) < g(P_j)$) for all $P_i, P_j \in \mathbf{P}$. 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 $g(P_i) \le g(P_j)$. Note that if there are nodes with equal lower bounds, then the heuristic function for a serial best-first search is consistent, but not completely consistent, with the lower bound function.

Theorem 3: The necessary condition for $T(k, 0) < T(1, 0)/k$ is that the heuristic function h is not completely consistent with g.

Proof: An acceleration anomaly does not exist if $\Phi^1 \subset \Phi^k$ because at least $\lceil |\Phi^1|/k \rceil$ iterations are needed to expand the nodes in Φ^1. Hence, the proof is based on the assumption that a node $P_{i_1} \in \Phi^1$ exists and that $P_{i_1} \notin \Phi^k$. This means that P_{i_1} is terminated by a lower bound test in the parallel case. That is, there is a feasible solution node $P_{i_2} \in \Phi^k$ such that $f(P_{i_2}) \le g(P_{i_1})$, and P_{i_2} does not exist in the serial case when P_{i_1} is expanded. Let P_{i_3} be the immediate parent of P_{i_2}. Referring to Fig. 5, two cases are possible.

First, $h(P_{i_3}) > h(P_{i_1})$. This means that P_{i_3} has not been generated when P_{i_1} is selected in the serial case. Since $P_{i_2}\mathbf{L}P_{i_1}$, thus $g(P_{i_3}) \le g(P_{i_2}) = f(P_{i_2}) \le g(P_{i_1})$, which implies that h is not completely consistent with g. Second, $h(P_{i_3}) < h(P_{i_1})$. In order for P_{i_1} to exist in the serial case, $P_{i_4} \in \Phi^1$ must exist such that $P_{i_4}\mathbf{L}P_{i_3}$ and that $h(P_{i_4}) < h(P_{i_3})$. By the transitivity of lower bound tests, $P_{i_4}\mathbf{L}P_{i_1}$. This contradicts the assumption that $P_{i_1} \in \Phi^1$. □

The significance of Theorems 1 and 3 is in showing that acceleration anomalies may exist and that detrimental anomalies can be prevented for depth-first searches when no approximation is allowed and an unambiguous heuristic selection function is used. For a best-first search without approximation, detrimental anomalies can be prevented by using an unambiguous heuristic selection function; however, acceleration anomalies may exist when there are nonsolution nodes of the B&B tree with lower bounds equal to the optimal solution value, since these nodes have heuristic values that are not completely consistent with their lower bound values. For a breadth-first search without approximation, no acceleration anomaly occurs because at least one node belonging to Φ^1 must be expanded in each iteration of a parallel breadth-first search.

It is important to note that the conditions in Theorem 3 are not necessary when approximate solutions are sought. An example in Fig. 2 shows the existence of an acceleration anomaly when $\epsilon = 0.1$ and h is completely consistent with g (since a best-first search is used and all lower bounds are distinct). In this case, the additional necessary condition is that the lower bound test with approximation is inconsistent with h; that is, there exist P_i and P_j such that $h(P_i) > h(P_j)$ and that $P_i\mathbf{L}P_j$. Clearly, this necessary condition is weak and can be satisfied easily.

VII. Conclusion

Branch-and-bound algorithms belong to a general class of algorithms that can solve a wide variety of combinatorial-search problems. Two major ways to improve the search efficiency are guiding the search by a heuristic function (or selection rule) and narrowing the search space (or elimination rules). Anomalies due to paral-

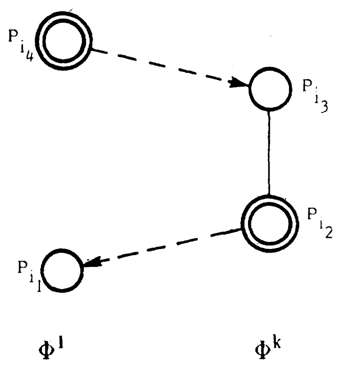

Fig. 5. Proof of Theorem 3 (the notation used is similar to those of Fig. 4).

lelism may occur because of the identical heuristic values of different subproblems and the inconsistencies between the selection and elimination rules.

In this correspondence anomalies between the serial and parallel cases for the same search strategy are studied. A method is proposed to assign a distinct heuristic value to each node and to uniquely identify a node in the B&B tree. It has been proved that detrimental anomalies can be prevented for a best-first or breadth-first search by using an unambiguous heuristic function, even when approximations are allowed. For a depth-first search, detrimental anomalies can only be prevented when an exact optimal solution is sought. On the other hand, acceleration anomalies may occur when one of the following conditions is true: 1) a depth-first search is used; 2) a best-first search is used, and some nonsolution nodes have lower bounds equal to the optimal solution value; or 3) a suboptimal solution is sought.

Although our results have been proved with respect to a system in which all subproblems are maintained in a single list, they apply to a system in which multiple subproblem lists are used. When there are multiple lists, one for each processor, a subproblem with the minimum heuristic value is selected from each local list for decomposition. This subproblem may not belong to the global set of active subproblems with the minimum heuristic values, but the subproblem with the minimum heuristic value will always be expanded by one of the processors. Further, when multiple lists are used, it is not difficult to maintain a global incumbent in a global data register [19], [20]. Hence the behavior of using multiple lists is analogous to that of a centralized list.

Due to the limitation of space, deceleration anomalies, anomalies due to dominance relations, and anomalies when k_1 and k_2, $1 \leq k_1 < k_2$, processors are used are not discussed here. These results can be found elsewhere [9]–[11].

References

[1] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness.* San Francisco, CA: W. H. Freeman, 1979.
[2] T. Ibaraki, "Computational efficiency of approximate branch-and-bound algorithms," *Math. Oper. Res.*, vol. 1, no. 3, pp. 287–298, 1976.
[3] ——, "Theoretical comparisons of search strategies in branch-and-bound algorithms," *Int. J. Comput. Inform. Sci.*, vol. 5, no. 4, pp. 315–344, 1976.
[4] M. Imai, T. Fukumura, and Y. Yoshida, "A parallelized branch-and-bound algorithm implementation and efficiency," *Syst. Comput. Contr.*, vol. 10, no. 3, pp. 62–70, 1979.
[5] W. Kohler and K. Steiglitz, "Characterization and theoretical comparison of branch-and-bound algorithms for permutation problems," *J. Ass. Comput. Mach.*, vol. 21, no. 1, pp. 140–156, 1974.
[6] V. Kumar and L. Kanal, "A general branch-and-bound formulation for understanding and synthesizing AND/OR tree-search procedures," *Artif. Intell.*, vol. 14, pp. 179–197, 1983.
[7] T. H. Lai and S. Sahni, "Anomalies in parallel branch-and-bound algorithms," *Commun. Ass. Comput. Mach.*, vol. 27, no. 6, pp. 594–602, 1984.
[8] E. L. Lawler and D. W. Wood, "Branch-and-bound methods: A survey," *Operat. Res.*, vol. 14, pp. 699–719, 1966.
[9] G. -J. Li and B. W. Wah, "Computational efficiency of parallel approximate branch-and-bound algorithms," in *Proc. 1984 Int. Conf. Parallel Processing,* Aug. 1984, pp. 473–480.
[10] ——, "Computational efficiency of parallel approximate branch-and-bound algorithms," School Elec. Eng., Purdue Univ., West Lafayette, IN, Tech. Rep. TR-84-6, Feb. 1984.
[11] G. -J. Li, "Parallel processing of combinatorial search problems," Ph.D. dissertation, Purdue Univ., West Lafayette, IN, Dec. 1985.
[12] L. Mitten, "Branch-and-bound methods: General formulation and properties," *Operat. Res.*, vol. 18, pp. 24–34, 1970.
[13] J. Mohan, "Experience with two parallel programs solving the traveling-salesman problem," in *Proc. 1983 Int. Conf. Parallel Processing,* pp. 191–193, 1983.
[14] T. L. Morin, "Branch-and-bound strategic for dynamic programming," *Operat. Res.*, vol. 24, no. 4, 1976.
[15] J. Pearl, *Heuristics: Intelligent Search Strategies for Computer Problem Solving.* Reading, MA: Addison-Wesley, 1984.
[16] Y. Sekiguchi, "A unifying framework of combinatorial optimization algorithms; tree programming and its validity," *J. Operat. Res. Japan,* vol. 24, Mar. 1981.
[17] D. R. Smith, "Random trees and the analysis of branch-and-bound procedures," *J. Ass. Comput. Mach.*, vol. 31, pp. 163–188, Jan. 1984.
[18] B. W. Wah and Y. W. Ma, "MANIP—A parallel computer system for implementing branch-and-bound algorithms," in *Proc. 8th Ann. Int. Symp. Comput. Architect.*, 1982, pp. 239–262.
[19] ——, "The architecture of MANIP—A multicomputer architecture for solving combinatorial extremum-search problems," *IEEE Trans. Comput.*, vol. C-33, pp. 377–390, May 1984.
[20] B. W. Wah, G. -J. Li, and C. F. Yu, "The Status of MANIP—A multicomputer architecture for solving combinatorial extremum-search problems," in *Proc. 11th Ann. Int. Symp. Comput. Architect.*, June 1984, pp. 56–63.
[21] ——, "Multiprocessing of combinatorial search problems," *IEEE Computer,* pp. 93–108, June 1985.
[22] B. W. Wah and C. -F. Yu, "Probabilistic modeling of branch-and-bound algorithms under a best-first search," *IEEE Trans. Software Eng.*, vol. SE-11, pp. 922–934, Sept. 1985.

Performance of Unbuffered Shuffle-Exchange Networks

MANOJ KUMAR AND J. R. JUMP

Abstract—**The throughput of unbuffered shuffle-exchange networks (also known as delta networks) is related to the arrival rate by a quadratic recurrence relation. Lower and upper bounds on the solution of this recurrence relation are derived in this paper. Two approaches for improving the throughput of unbuffered delta networks are investigated. The first approach combines multiple delta subnetworks of size $N \times N$ each, in parallel, to obtain a network of size $N \times N$. Three policies used to distribute the incoming packets between the subnetworks are discussed and the relative effect of each on the throughput is compared. The second approach replaces each link of the simple delta networks by K parallel links (K equals $2, 4, \cdots,$). The throughput of such networks is analyzed and one possible implementation for crossbar switches that could be used in these networks is discussed. The throughput of such networks with four parallel links is almost equal to the throughput of crossbars.**

Manuscript received July 31, 1984; revised July 10, 1985. This work was supported by the National Science Foundation under Grant MCS 80-01667. An earlier version of this paper was presented at the 1983 International Conference on Parallel Processing, Ann Arbor, MI, August 1983.

M. Kumar was with Rice University, Houston, TX. He is now with the IBM Corporation, T. J. Watson Research Center, Yorktown Heights, NY 10598.
J. R. Jump is with the Department of Electrical Engineering, Rice University, Houston, TX 77251.

IEEE Log Number 8608657.