COMPUTATIONAL EFFICIENCY OF PARALLEL APPROXIMATE BRANCH-AND-BOUND ALGORITHMS

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

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 dominance relation, the data structure, and the search strategies. Anomalies owing to parallelism may happen and occur frequently when approximations or dominance tests are used. In this paper, sufficient conditions to guarantee that parallelism will not degrade the performance are developed. Necessary conditions for allowing parallelism to have a speedup greater than the number of processors are also presented. Theoretical analysis and simulations show that a best-first search is robust for parallel processing.

KEYWORDS AND PHRASES: Anomalies, approximation, branch-and-bound algorithms, dominance criteria, heuristic search, lower-bound test, NP-hard problems, parallel processing.

1. INTRODUCTION
The search for solutions in a combinatorially large problem space is a major problem in artificial intelligence and operations research. Many practical and theoretical problems such as traveling salesman, warehouse location, job-shop scheduling, knapsack, vertex cover, integer programming, theorem proving, expert systems, and game playing can be solved by combinatorial searches. Exhaustive search is usually impractical and prohibitively expensive for solving large search problems, especially when the problem is NP-hard. Studies on improving search efficiency is, thus, of considerable importance. Research has been conducted on designing one unified method for a wide variety of problems. The most general technique is the branch-and-bound algorithm [2,8]. This is a partitioning algorithm that decomposes a problem into smaller subproblems and repeatedly decomposes until infeasibility is proved or a solution is found [8,10]. Backtracking, dynamic programming and AND-OR tree search can be viewed as variations of branch-and-bound algorithms.

From the viewpoint of implementation, approximations and parallel processing are two major approaches to enhance the efficiency of branch-and-bound algorithms. Approximations are useful for solving NP-hard problems because a linear reduction in accuracy results in an exponential reduction of the average computational time for solving some NP-hard problems [14]. Properties of serial approximate branch-and-bound algorithms have been studied by Ibaraki [1]. 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 [12], or the problem is heuristically solvable in polynomial time (such as game trees). It is not practical to use parallel processing to solve a problem with exponential complexity because an exponential number of processors must be used to solve the problem in polynomial time.

Analytical properties of parallel approximate branch-and-bound algorithms have been rarely studied. From experimental results on vertex-cover, 0-1 knapsack and integer-programming problems, the reduction in computational time of branch-and-bound algorithms with lower-bound tests and a best-first search is approximately linear on the average with respect to the number of processors [13,15]. In general, a k-fold speedup (ratio of the number of iterations 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 PAB algorithms using k processors can be (a) less than one—"detrimental anomaly" [5,7,11,17]; (b) greater than one—"acceleration anomaly" [5,6,17]; or (c) between one and k—"deceleration anomaly" [5,7,11,13]. It is desirable to discover conditions that preserve the acceleration anomalies, eliminate the detrimental anomalies and minimize the deceleration anomalies.

The objective of this paper is to study the computational efficiency and anomalous behavior of branch-and-bound algorithms under parallel processing and approximations. These results are helpful for the designers to understand the existence of anomalies and to modify existing algorithms so that detrimental anomalies can be prevented and acceleration anomalies can be enhanced. They can also be used to predict the maximum number of processors needed to attain a near-linear speedup.

2. PARALLEL APPROXIMATE BRANCH-AND-BOUND ALGORITHMS
Many theoretical properties of branch-and-bound algorithms have been developed by Kohler and Steiglitz [6] and Ibaraki [1,2,3]. A brief summary of these properties to solve a minimization problem are given in [15], and only the important properties that will be used in the following sections are stated here.

Branch-and-bound algorithms can be characterized by four constituents: a branching rule, a selection rule, one or more elimination rules 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 branch-and-bound algorithms.

The way in which \( P_0 \) is repeatedly decomposed into smaller subproblems can be represented as a finite rooted tree \( B = (P, E) \), where \( P \) is a set of subproblems and \( E \) is a set of edges. The root of the tree is \( P_0 \). If a subproblem \( P_i \) is obtained from \( P_j \) by decomposition, then \( (P_i, P_j) \in E \). The level number of a node is the number

This research was supported by National Science Foundation Grant ECS81-05068.

0190-3918/84/0000/0473-01.00 © 1984 IEEE
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 the subproblems decomposable from \( P_i \), let \( P_j \) be the \( j \)th subproblem decomposable from \( P_i \) and let \( k_j \) be the number of such subproblems (i.e., \( k_j = \{ (P_{i,x}) : (P_{i,x}) \in E \} \)). Then \( f \) satisfies:

\[
f(P_j) = \min_{j=1,\ldots,k} \{ f(P_{i,j}) \}
\]

(1)

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

(a) \( g(P_i) \leq f(P_i) \) for \( P_i \in P \).

(b) \( g(P_j) = f(P_j) \) for \( P_j \in T \).

(c) \( g(P_j) \leq g(P_{i,j}) \) for \( (P_{i,j}) \in E \).

(lower bounds always increase)

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:

\[
g(P_i) \geq z
\]

(5)

The approximate branch-and-bound algorithm is the same as the optimal algorithm except that the lower-bound test is modified to:

\[
g(P_j) \geq \frac{z_p}{1+\epsilon} \quad \epsilon \geq 0, \ z \geq 0
\]

(6)

where \( \epsilon \) is an allowance function specifying the allowable deviation of a suboptimally valued from the exact optimal value. The final incumbent value \( z_p \) obtained by the modified lower-bound test deviates from the optimal solution value, \( z_0 \), by:

\[
\frac{z_p}{1+\epsilon} \leq z_0 \leq z_p
\]

(7)

Let \( L \) denotes the lower-bound cutoff test, that is, \( P_i \in L \) means that \( P_i \) is a feasible solution and \( f(P_i)/(1+\epsilon) \leq g(P_i), \ \epsilon \geq 0 \).

Branch-and-bound algorithms have inherent parallelism. Each of the four rules of serial branch-and-bound algorithms can be implemented by parallel processing.

(a) 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 minimum lower bounds must be selected. The selection function (Eq. 11) becomes:

\[
s_i(U) = \{ P_i \mid h(P_i) = \min_{P_j \in U} h(P_j) \}
\]

(11)

(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 \( k \).

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

(d) Parallel elimination test: The lower-bound test (Eq. 5 or 6) can be sped up by comparing lower bounds of multiple subproblems with the incumbent. The elimination of subproblems by dominance tests can also be carried out in parallel. However, the bounding function and the dominance test are problem-dependent, and software implementation is more flexible.
3. ANOMALIES ON PARALLEL BRANCH-AND-BOUND ALGORITHMS

In this section, some anomalies on parallel approximate branch-and-bound algorithms are illustrated. The parallel computation model for studying the anomalies consists of a set of processors connected by a network. There can be a single subproblem list for all the processors, or there are multiple lists, one for each processor. The distinction lies in the memory configuration. When all the processors are connected to a centralized memory, the subproblem list is global to all the processors. When each processor has a private memory, only the local subproblem list can be accessed. The workload is balanced through the network. It is assumed that the processors operate synchronously in executing the steps of the branch-and-bound algorithm.

Let $T^k([k,c])$ and $T^4([k,a])$ denote the number of iterations required for expanding a branch-and-bound tree using centralize and $k$ subproblem lists respectively, where $k$ is the total number of processors used, and $c$ is the allowance function. Since the subproblems are decomposed synchronously, the number of iterations is measured by the number of times that subproblems are decomposed in each processor.

For the branch-and-bound tree in Figure 1a, the order of nodes expanded using a depth-first search on a single processor is $P_0, P_1, P_2, \ldots$, all the nodes in $T_1$ that result in a feasible solution $P_0$ with value 8, $P_k$, and all the nodes in $T_k$ that result in the optimal solution $f(P_k)=5$. $P_1$ is terminated by the lower-bound test using $P_0$. In contrast, when two processors are used, $P_1$ and $P_2$ are expanded concurrently. After the expansion, $P_2$ is terminated as a result of dominance by $P_1$. Since $T_1$ is terminated, $P_1$ and $T_2$ must be expanded. If subtree $T_2$ is large, the combined time of expanding $T_1$ and $T_2$ using one processor can be smaller than the combined time of expanding $T_1$ and $T_2$ using two processors. This anomaly will be investigated in Section 3.2.

An example illustrating a detrimental anomaly when approximations are allowed is shown in Figure 1b. In a parallel depth-first search, subtree $T_2$ is terminated owing to the lower-bound test of $P_0$: $f(P_0)/(1+c) \leq g(P_2)$ where $c = 0.1$. In a parallel depth-first search with two processors, a feasible solution $P_1$ is found in the second iteration, and nodes $P_1$ and $P_2$ are terminated owing to the lower-bound test of $P_2$. Consequently, subtree $T_2$ has to be expanded which will eventually terminate subtree $T_3$. If the size of $T_3$ is much larger than that of $T_2$, the time it takes to expand $T_3$ using two processors will be longer than the time it takes to expand $T_3$ using one processor. Detrimental anomalies with approximations are discussed in Section 3.3.

Figure 2a shows an acceleration anomaly with dominance and lower-bound tests under a depth-first search. In a parallel depth-first search, when two processors are used, $P_2$ is expanded in the second iteration, and nodes $P_1$ and $P_2$ are terminated. Therefore, only $T_3$ is expanded in the parallel case. A similar example on lower-bound tests with approximations is shown in Figure 2b. Acceleration anomalies will be discussed in Section 5.

Other anomalous examples can be created for breadth-first and best-first searches. However, the important consideration here is not in knowing that anomalies exist, but in understanding why these anomalies occur. Our objective is to find the sufficient conditions to ensure that $T^k([k,c]) \leq T^4([1,c])$ and $T^k([k,a]) \leq T^4([1,a]),$ as well as the necessary conditions for $T^k([k,c]) \leq T^4([1,c])/k$ and $T^k([k,a]) \leq T^4([1,a])/k$.

4. GENERALIZED HEURISTIC SEARCHES

Recall in Sections 2 that the selection function uses heuristic values to define the order of node expansions.
(a) with lower-bound and dominance tests ($\epsilon=0$).

(b) with lower-bound tests only ($\epsilon>0$).

Figure 2.
Examples of acceleration anomalies under a depth-first search.

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 node chosen depends on the order of insertion into the stack. Suppose the rightmost son of a parent node is always expanded and inserted first. Then the leftmost son will be 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 Figure 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 search function can choose any two of these nodes for expansion in the next iteration. 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 an unexpected behavior as compared to the serial case and is exactly the reason for the anomalies reported in [7].

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. The path number of a node in a tree is a sequence of $d+1$ integers that represent the path from the root to this node where $d$ is the maximum level number of the tree. The path number $E = e_0 e_1 ... e_d$ is defined recursively as follows. The root $P_0$ exists at level 0 and has a path number of 000...0. A node $P_j$ on level $j$ that is the $j$th son (counting from the left) of $P_i$ with path number $E_p = e_0 e_1 ... e_{j-1} 000...$ has path number $E_p = e_0 e_1 ... e_{j-1} j00...$. As an example, the path numbers for all the nodes in the tree of Figure 3 are shown.

To compare path numbers, the relations "$\geq$" and "$\leq$" must be defined. A path number $E_1 = e_1 e_2 ...$ is less than another path number $E_2 = e_1 e_2 ...$ if there exists $0 < i < d$ such that $e_i = e_i$ and $e_j < e_j$. The path numbers are equal if $e_i = e_i$ for $0 < i < d$. 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 of a branch-and-bound algorithm, 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.

\[
\begin{align*}
\text{level number, path number} & \quad \text{for breadth-first search} \\
\text{(path number)} & \quad \text{for depth-first search} \\
\text{(lower bound, level number, path number)} & \quad \text{or} \\
\text{(lower bound, -level number, path number)} & \quad \text{for best-first search}
\end{align*}
\]
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) $h(P_i) > h(P_j)$ if $P_i, P_j \in P$ and $P_i \neq P_j$.
(b) $h(P_i) \leq c_i h(P_j)$ if $P_j$ is a descendant of $P_i$.

All heuristic values in the active list are distinct. (14)

5. SUFFICIENT CONDITIONS TO ELIMINATE DETRIMENTAL ANOMALIES IN A SINGLE SUBPROBLEM LIST

5.1 Parallel Branch-and-Bound with Lower-Bound Tests Only

In this section, we show that any heuristic search with a heuristic function that satisfies Eqs. 14 and 15 can guarantee that $T(k, 0) \leq T(1, 0)$ when only lower-bound tests with $c = 0$ are used in a system with a centralized subproblem list. First, we define a basic node as the node with the smallest heuristic value in each iteration. All the nodes expanded in a serial algorithm are basic nodes. Let $\Phi^1$ and $\Phi^2$ be the sets of nodes in the branch-and-bound tree expanded using one and $k$ processors respectively. To show that $T(k, 0) \leq T(1, 0)$, it is necessary to prove: (a) at least one node that belongs to $\Phi^1$ (the basic node) is expanded in each iteration of the parallel search; and (b) that once all the nodes in $\Phi^1$ 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 expanded or terminated when $P_i$ is expanded.

Proof: Suppose in the current active list, $P_i \in U$ is a basic node. Assume that there exists a node $P_j$ such that $h(P_j) < h(P_i)$ and $P_j$ has not been expanded or terminated when $P_i$ is expanded. Since $P_j$ has the minimum heuristic value among the set of currently active nodes $U$, $P_j$ must be active at that time. That is, $P_j$ is a descendant of some node $P_k$, $P_k \in U$, and $h(P_k) < h(P_j)$. By Eq. 15, $h(P_k) < h(P_j) \leq h(P_i)$ which contradicts the assumption that $h(P_j) < h(P_i)$.

The following theorem proves that any selection function with properties given in Eqs. 14 and 15 are sufficient to eliminate the detrimental anomalies.

Theorem 1: Let $\varepsilon = 0$, D = I (identity relation), i.e., an exact optimal solution is sought and the dominance tests are not active. $T(k, 0) \leq T(1, 0)$ for parallel heuristic searches of a single optimal solution in a centralized list using any heuristic function with properties defined in Eqs. 14 and 15.

Proof: The proof is by contradiction. Suppose there exists a basic node $P_i$, such that $P_i \in \Phi^1$ and $P_i \in \Phi^2$ (Figure 4). This means that $P_i$ is terminated by a lower-bound test in the serial case, and there must exist a feasible solution $P_j \in \Phi^1$ such that $f(P_j) \leq g(P_i)$ and $P_j$ has not been obtained when $P_i$ is expanded in the parallel case. It implies that a proper ancestor $P_j \in \Phi^1$ of $P_i$ exists such that $h(P_j) < h(P_i)$, and $P_j$ is obtained before $P_i$ and terminates $P_j$. Since $P_j$ is a basic node in the parallel search, $h(P_j) < h(P_i)$ and $P_j$ has not been expanded when $P_i$ is expanded in the parallel case, $P_j$ must be terminated according to Lemma 1. For the parallel search, there must exist a feasible solution $P_j \in \Phi^2$ such that $f(P_j) \leq g(P_i)$, and $P_j$ has not been obtained when $P_j$ is expanded in the serial case. Two cases are possible:

First, $P_i$ is not generated when $P_j$ is expanded in the serial case, i.e., a proper ancestor $P_k \in \Phi^1$ of $P_i$ exists such that $h(P_k) > h(P_i)$. According to the properties of lower-bound functions (Eqs. 2, 3 and 4), we have $f(P_j) \leq g(P_i) \leq f(P_j) \leq g(P_j)$. Moreover, in the parallel case, $P_j$ has been obtained when $P_i$ is generated because $P_i$ is terminated by $P_j$. Since $h(P_j) < h(P_i)$, $P_j$ must be generated before $P_i$ by Lemma 1. Consequently, $P_j$ is available when $P_i$ is expanded. Hence, $P_i$ must be generated in the parallel algorithm which contradicts the assumption that $P_j \in \Phi^1$.

Second, $h(P_j) < h(P_i)$, and $P_k$, as well as its descendant $P_i$, have been terminated in the serial case and not in the parallel case. We can then apply the above argument again to $P_k$, and eventually obtain a sequence of nodes $P_i, P_j, ..., P_m$ as depicted in Figure 4. There are three possibilities:

(a) The first node $P_i$ occurs in the serial case (Figure 4a). Since $h(P_i) < h(P_j)$, otherwise $P_i$ cannot be expanded by $P_j$, in the serial case, and since $h(P_i) < h(P_i)$ by the same argument as $h(P_j) < h(P_i)$, we have $h(P_i) < h(P_i)$. Repeating this, we get $h(P_i) < h(P_i)$. By Lemma 1, $P_i$ must be expanded in the parallel case and terminates $P_i$, which contradicts that $P_j \in \Phi^1$.

(b) The first node $P_i$ occurs in the parallel case (Figure 4b). Since $P_i$ is a feasible solution, we have $f(P_i) \leq g(P_i) \leq g(P_i) \leq \cdots \leq g(P_i)$, similar to the argument for $P_i$, discussed in the first case, we can explain that $P_i$ has been obtained when $P_i$ is selected. Therefore, $P_i$ is terminated in the parallel case which contradicts that $P_i \in \Phi^1$.

(c) There is a cycle of cutoffs such that $P_{i-1}, P_i, P_{i-2}, ..., P_i, P_{i+1}$, and $P_i, P_i$, where $P_i$ denotes a lower-bound cutoff test (Figure 4c). By transitivity, we have $f(P_i) \leq g(P_i) \leq g(P_i) \leq g(P_i)$, which implies that $f(P_i) = f(P_i) = f(P_i)$. The heuristic value of all the nodes of the cycle are less than $h(P_i)$, so a feasible solution has been obtained.

![Figure 4. Proof of Theorem 1.](image-url)
before $P_1$ is selected. Thus $P_i$ must be terminated in the parallel case.

So far we have proved that at least one node of $\Phi^1$ is expanded in each iteration of a parallel heuristic search. It also implies that after all the nodes of $\Phi^1$ are expanded or terminated in the parallel heuristic search, no active node remains.

The above theorem shows that detrimental anomalies can be avoided for depth-first, breadth-first and best-first searches with $\epsilon = 0$ and $D = 1$ by augmenting the heuristic values with the path numbers. As a special case, for a best-first search in which all the lower bounds are distinct, the nodes with the smallest lower bounds are always selected through the priority queue. Hence the path numbers do not have to be used, and no detrimental anomaly will occur. A more general condition will be shown in Section 8.

5.2 Parallel Branch-and-Bound with Lower-Bound and Dominance Tests

When lower-bound and dominance tests are used together, it is important to note that dominance tests are not transitive with lower-bound tests. That is, $P_i \leq P_j$ and $P_j \leq P_k$ do not imply $P_i \leq P_k$. Similarly, $P_i \leq P_j$ and $P_i \leq P_j$ do not imply $P_i \leq P_j$. In both cases, only $|f(P_j)| \leq |f(P_k)|$ can be deduced. Recall that when $P_i \leq P_j$, it implies $|f(P_i)| \leq |f(P_j)|$. (Eq. 8) The converse is not true because otherwise the number of active nodes can always be reduced to one.

Dominance tests together with lower-bound tests are more apt to enhance the parallel branch-and-bound algorithms than if lower-bound tests were used alone; however, detrimental anomalies are more difficult to prevent. The sufficient conditions developed here ensure that $T^*(k, 0) \leq T^*(1, 0)$ are stronger than those conditions developed in the last section. These conditions are defined for special classes of heuristic functions and dominance relations. A heuristic function $h$ is said to be consistent with the lower-bound function $g$ if $h(P_i) < h(P_j)$ implies that $g(P_i) \leq g(P_j)$ for all $P_i, P_j \in \mathcal{P}$. A dominance relation $D$ is said to be consistent with the heuristic function $h$ if $P_i \leq P_j$ implies that $h(P_i) \leq h(P_j)$ for all $P_i, P_j \in \mathcal{P}$. The following theorem proves that these conditions are sufficient.

Theorem 2: Let $\epsilon = 0, D = 1$. $T^*(k, 0) \leq T^*(1, 0)$ holds for dominance searches that satisfy Eq's 11 and 15 and the following conditions: (a) heuristic function $h$ is consistent with lower-bound function $g$, and (b) Dominance relation $D$ is consistent with $h$.

Due to space limitations, the proofs of Theorem 2 and the following theorems are omitted. All of the proofs can be found in [8].

One possible way in which the two consistency requirements in Theorem 2 can be satisfied would be: (a) to use a best-first search, (b) to require that all the lower bounds be distinct or to use path numbers to break the tie for identical lower bounds, and (c) to use a dominance relation that is consistent with $g$, i.e., $P_i \leq P_j$ implies that $|f(P_i)| \leq |f(P_j)|$. The first requirement implies that the heuristic function is consistent with the lower-bound function. The last two requirements are true in order for the dominance relation to be consistent with the heuristic function. The requirement on the consistency of the dominance relation with respect to $g$ is satisfied in many practical problems. For instance, for the 0-1 knapsack problem, the upper bound of the maximization problem can be computed by arranging the objects in decreasing profit-to-weight ratios and allocating as many objects as possible into the knapsack while allowing fractional allocation of last object packed. A dominance relation is consistent with the upper bound function for the knapsack problem if it is defined as follows. For $P_i$ and $P_j$ defined on a given subset of objects, $P_i \leq P_j$ if (1) the profit of $P_i$ is larger than that of $P_j$ and the total weight of $P_i$ is less than that of $P_j$; or (2) the profit and weight of $P_i$ are equal to the corresponding profit and weight of $P_j$ and the path number of $P_i$ is less than that of $P_j$. Likewise, the hybrid dynamic-programming/branch-and-bound algorithm for solving the traveling-salesman problem has a dominance relation that is consistent with $g$. So are the dominance relations for the shortest-path problem [3], the n-job two-machine mean-finish-time flowshop problem and the n-job one-machine scheduling problem with deadlines [SAH76]. Note that the example in Figure 1a does not satisfy the requirement that $D$ is consistent with $h$.

5.3 Parallel Branch-and-Bound Algorithms with Approximations

When parallel approximate branch-and-bound algorithms are considered, Theorems 1 and 2 are no longer valid (see the example in Figure 1b). The reason for the anomaly is that the lower-bound tests under approximation, $L$, are not transitive. That is, $P_i \leq P_j$ and $P_j \leq P_k$ do not imply $P_i \leq P_k$, since $|f(P_j)|/(1 + |e|) \leq |f(P_k)|$ implies $|f(P_i)|/(1 + |e|) \leq |f(P_k)|$ rather than $|f(P_i)|/(1 + |e|) \leq |f(P_k)|$. Somewhat surprisingly, it is possible that $\Phi^1$ and $\Phi^k$ are almost disjoint, and most of the nodes in $\Phi^1$ are not expanded in the parallel case. Detrimental anomalies can be avoided for a best-first or a breadth-first search under the same conditions as stated in Theorem 2.

Theorem 3: $T^*(k, \epsilon) \leq T^*(1, \epsilon), \epsilon > 0$, holds for parallel best-first searches for a single optimal solution when: (a) heuristic function, $h$, satisfies Eq's 14 and 15; (b) dominance relation, $D$, is consistent with $h$.

Condition (a) in the above theorem is satisfied if all the lower-bound values are distinct. Otherwise, the identical lower bounds must be augmented by the path numbers to break the tie. For depth-first searches, the conditions of Theorem 3 are not sufficient, and it is hard to avoid the detrimental anomalies in this case.

6. NECESSARY CONDITIONS FOR ACCELERATION ANOMALIES IN A SINGLE SUBPROBLEM LIST

In this section, the necessary conditions for $T^*(1, 0) \leq T^*(1, 0)/x$ are developed. One of these conditions is based on the complement of the special class of dominance relations defined in Section 5.2. A dominance relation, $D$, is said to be inconsistent with $h$ if two nodes $P_i$ and $P_j$ exist such that $P_i \leq P_j$ and $h(P_i) > h(P_j)$. Another condition is based on the complete consistency of heuristic 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 $g(P_i) \leq g(P_j)$. Note that this is not exactly the complement of the consistency of heuristic functions defined in the last section. To illustrate the difference, if $g(P_i) = g(P_j)$ is allowed, then the heuristic function for a best-first search is consistent, but not completely consistent, with the lower-bound function.

Theorem 4: Let $\epsilon = 0$ and assume that a single optimal solution is sought. The necessary condition for $T^*(k, 0) \leq T^*(1, 0)/x$ is either that (a) the heuristic function is not completely consistent with $g$, or (b) the dominance relation is inconsistent with $h$.

478
For instance, the example in Figure 2a satisfies the conditions of Theorem 4. According to Theorems 2 and 4, we can conclude that if dominance tests are active, both detrimental and acceleration anomalies exist when a breadth-first or a depth-first search is used. However, it is possible that acceleration anomalies exist and detrimental anomalies can be prevented for first-best searches (owing to the difference between consistency and complete consistency). It is important to note that the conditions in Theorem 2 are not necessary for approximate solutions to be sought, i.e., acceleration anomalies may occur even though \( h \) is complete consistent with \( g \) and the dominance tests are consistent with \( h \). In summary, acceleration anomalies may occur in one of the following cases: (a) when using a breadth-first or a depth-first search; (b) when some nodes have identical lower bound; (c) when the dominance relation is inconsistent with the heuristic function; (d) when a suboptimal solution is sought; or (e) when using multiple lists of subproblems (discussed in the next section).

7. MULTIPLE SUBPROBLEM LISTS

When there are multiple subproblem lists, one for each processor, a node is selected from each local list with the minimum heuristic value for decomposition. This node may not belong to the global set of active nodes with the minimum heuristic values, but the node with the minimum heuristic value will always be expanded by one processor as long as the order of insertion into a subproblem list is the same for the serial and the parallel searches. 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. Hence, when dominance tests are inactive (\( D = 1 \)), no detrimental anomaly occurs. However, the performance of using multiple lists is usually worse than that of a single subproblem list [16].

When dominance tests are active, these tests can be restricted to the local set of current dominating nodes or they can be performed on all the current dominating nodes. If the dominance tests are carried out on all the current dominating nodes, it is necessary to broadcast newly generated nodes to all the processors and to balance the sizes of the multiple sets of current dominating nodes at the end of each iteration. The behavior is similar to that of a centralized list. On the other hand, if the dominance tests are performed on the local set of current dominating nodes, then it is possible that \( P_i, D_P \), and \( P_j \) is not terminated because the two nodes are stored in different processors. As a result, both detrimental and acceleration anomalies may occur. For instance, a detrimental anomaly may happen because a subtree rooted at \( P_i \) is pruned in the serial case and is not pruned in the parallel case. However, if \( P_i \) has a feasible solution \( P_j \) such that \( P_i, L_P \), and \( P_j \) are located in the same processor, then the subtree rooted at \( P_i \) will be pruned in the parallel case because \( P_i, D_P \) is inactive and not in the serial case because \( P_i \) is terminated by \( P_i, D_P \). This may introduce an acceleration anomaly. It is found that anomalies happen more frequently when dominance tests are performed on local current dominating nodes.

8. ROBUSTNESS OF PARALLEL BEST-FIRST SEARCHES

The results in the previous sections are useful for preventing detrimental anomalies and preserving acceleration anomalies. However, they do not apply when deceleration anomalies are concerned. Deceleration anomalies are usually unavoidable and diminishing returns occur when the number of processors is large. So far, all known results showed that for a depth-first search, a near-linear speedup holds only for a small number of processors. In addition, the generalized heuristic search mentioned in Section 4 cannot guarantee that \( T(k_2) \leq T(k_1) \), \( k_1 \leq k_2 \), even with \( r = 0 \) and inactive dominance tests. The results in this section indicate that a best-first search with lower-bound tests only has the best behavior as far as deceleration anomalies and relative performance are concerned. Since the performance is not affected by using single or multiple subproblem lists, the superscript in \( T \) is dropped.

Theorem 5: For a parallel best-first search with \( k \) processors, \( r = 0 \), \( D = 1 \) and \( g(P_i) \neq \emptyset \) if \( P_i \) is not an optimal-solution node,

\[
\frac{\eta}{k} \leq T(k, 0) \leq \frac{T(1, 0) + k - 1}{k} \eta
\]

where \( \eta \) is the maximum number of levels of the branch-and-bound tree to be searched.

In most applications, the height of the branch-and-bound tree is a polynomial function of the problem size. Hence, the second term on the right hand side of Eq. 16 is much smaller than the first term. and a near-linear speedup holds for best-first searches in a considerable range of the number of processors. As an example, if \( \eta = 50 \), \( T(1, 0) = 10^5 \) (for a typical traveling-salesman problem), and \( k = 1000 \), then \( 1000 \leq T(1000, 0) \leq 1049 \). This means that almost linear speedup can be attained with one thousand processors.

From Theorem 5, it is also easy to determine the maximum number of processors that must be used to have a near-linear speedup. Assume that the speedup required is \( T(1, 0)/\eta k \geq T(k, 0) \), \( 0 < \eta < 1 \). From Eq. 16,

\[
T(k, 0) \leq \frac{T(1, 0)}{k} + \frac{k - 1}{k} \eta
\]

This results in:

\[
k \leq \frac{T(1, 0)}{\eta} \eta + 1 \tag{17}
\]

For instance, if \( \eta = 0.9 \), \( \eta = 50 \), and \( T(1, 0) = 10^5 \), then \( k \leq 2223 \). That is, a minimum of 0.9 speedup is obtained if 2223 or less processors are used.

Lastly, we can obtain the sufficient condition for ensuring the monotonic increase in computational efficiency of parallel branch-and-bound algorithms with a best-first search. If the assumptions of Theorem 5 are satisfied, \( T(k_2, 0) \leq T(k_1, 0) \), \( k_1 < k_2 \), then:

\[
T(k_2, 0) \geq \frac{k_1 k_2 - 1}{k_2 - k_1} T(1, 0)
\]

For \( k_2 > k_1 \), since \( k_2 > k_1 \), \( k_2 \geq k_1 + 1 \), then the theorem is true if \( T(1, 0) = 10^5 \). In other words, there is always a monotonic increase in performance for all \( 1 \leq k_1 < k_2 \). For example, if \( k = 50 \), \( T(1, 0) = 10^5 \), there will be no risk of detrimental anomalies for any combinations of \( 1 \leq k_1 < k_2 \leq 141 \).

Theorem 5 assumes that \( g(P_i) \neq \emptyset \) unless \( P_i \) is an optimal-solution node. When this assumption is relaxed, acceleration anomalies may occur. Simulation results on 0-1 knapsack and vertex-cover problems revealed that the speedup of a parallel best-first search is much better than that of a parallel depth-first search when the number of processors is large [9].

9. CONCLUSION

Branch-and-bound algorithms can be used to solve a wide variety of combinatorial-search problems.
Anomalies may occur when parallelism and approximation are applied owing to the ambiguities of heuristic functions and inconsistencies between the selection rule and the elimination rules. A method of using the path numbers is proposed in this paper to assign a distinct heuristic value to each node and to uniquely order the search in the branch-and-bound tree. Other conditions on consistencies of dominance relations and heuristic functions are also developed. A summary of the results proved in this paper are shown in Table 1. Note that these results are developed for finding a single solution. If all the solutions are sought, the corresponding conditions can be investigated similarly.

A best-first search is found to be a good search strategy that can guarantee little degradation in performance with increasing degrees of parallelism. It is more suitable for parallel processing than breadth-first or depth-first searches due to the large number of active subproblems. More general results on the relative performance between 3 and 2 processors also show that best-first searches are more robust as far as deceleration anomalies are concerned. A near-linear speedup with respect to the number of processors has been observed for parallel best-first searches [13,15]. Owing to the large number of active subproblems in a best-first search, the effective mapping of active subproblems onto the secondary storage is an important problem. It was found that a direct implementation of the algorithm on a memory hierarchy resulted in an intolerable amount of disk accesses. A modified best-first search that can overlap all the secondary-storage accesses with computations has been proposed and studied [16,18].

Table 1. Summary of results for the elimination of detrimental anomalies and the preservation of acceleration anomalies in parallel branch-and-bound algorithms.

<table>
<thead>
<tr>
<th>Allowance function</th>
<th>Dominance relation</th>
<th>Subproblem lists</th>
<th>Search strategies</th>
<th>Suff. cond. to eliminate detrimental Anomaly</th>
<th>Necessary cond. for acceleration Anomaly</th>
</tr>
</thead>
<tbody>
<tr>
<td>D=1</td>
<td>Single</td>
<td>all</td>
<td>IV</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>multiple</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D&gt;1</td>
<td>single</td>
<td>all</td>
<td>I, II, III, IV</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td></td>
<td>multiple</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D&gt;1</td>
<td>single or breadth-first</td>
<td></td>
<td>I</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>multiple or best-first</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D&gt;1</td>
<td>breadth-first</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>depth-first</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D&gt;1</td>
<td>single</td>
<td>breadth-first</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>multiple</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>all</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Conditions:
I: heuristic function satisfies Eq's 14 and 15.
II: h is consistent with g, i.e., g[Pj]<h[Pj] implies g[Pj]<g[Pj].
III: D is consistent with h, i.e., 3jPj D[Pj] implies h[Pj]<h[Pj].
IV: h is not completely consistent with g.
V: D is inconsistent with h.
* anomaly: the sufficient conditions are impractical.
+ exists: the necessary conditions are too loose.
+*: multiple subproblem lists and only local dominance tests are used.

REFERENCES