The Power of Dominance Relations in Branch-and-Bound Algorithms

TOSHIHIDE IBARAKI

Kyoto University, Kyoto, Japan

ABSTRACT A dominance relation D is a binary relation defined on the set of partial problems generated in a branch-and-bound algorithm, such that P_iDP_j (where P_i and P_j are partial problems) implies that P_j can be excluded from consideration without loss of optimality of the given problem if P_i has already been generated when P_j is selected for the test The branch-and-bound computation is usually enhanced by adding the test based on a dominance relation

A dominance relation D' is said to be stronger than a dominance relation D if P_iDP_j always implies $P_iD'P_j$ Although it seems obvious that a stronger dominance relation makes the resulting algorithm more efficient, counterexamples can easily be constructed. In this paper, however, four classes of branch-and-bound algorithms are found in which a stronger dominance relation always gives a more efficient algorithm This indicates that the monotonicity property of dominance relations would be observed in a rather wide class of branch-and-bound algorithms, thus encouraging the designer of a branch-and-bound algorithm to find the strongest possible dominance relation.

KEY WORDS AND PHRASES: combinatorial optimization, branch-and-bound algorithms, dominance relations, heuristic search, best-bound search, breadth-first search, depth-first search

CR CATEGORIES 3.64, 5.39, 5.49

1. *Introduction*

It is known that the branch-and-bound principle is applicable to a wide variety of combinatorial optimization problems (e.g. [1, 12, 22, 24, 29]). The underlying idea is to decompose a given problem, which is difficult to solve directly, into several partial problems of smaller sizes. The decomposition may be repeatedly applied until tests applied to the generated partial problems reveal that each undecomposed problem is either solved or proved not to provide an optimal solution of the original problem.

The test of a partial problem is usually based on computing a lower bound on the minimum objective value (when a minimal solution is sought). It is concluded that a partial problem does not provide an optimal solution of the original problem if the computed lower bound is greater than the objective value of the best feasible solution currently available (i.e. the incumbent).

A generalization of the lower bound test is also possible if the available information on partial problems can be used to show that a partial problem P_j cannot provide a better feasible solution than that obtainable from another partial problem P_i. This relation is denoted P_iDP_j and called a dominance relation. A test based on a dominance relation is carried out as follows: Partial problem P_j is excluded from consideration if a partial problem P_i such that P_iDP_j has already been generated. A formal description of a branch-and-bound algorithm, including both types of tests, will be given in Section 3 after explaining each constituent of the algorithm in Section 2.

Theoretical treatment of dominance relations seems to have been initiated by Kohler

Author's address: Department of Applied Mathematics and Physics, Faculty of Engineering, Kyoto University, Kyoto, Japan

and Steiglitz [18]. It is natural, however, to consider that similar ideas have been heuristically used in many other branch-and-bound algorithms. Some dominance relations used in practical branch-and-bound algorithms will be discussed in Section 2.

It was defined in [18] that a dominance relation D' is stronger than a dominance relation D if P_iDP_j implies $P_iD'P_j$ (i.e. $D' \supset D$ if a dominance relation $D(D')$ is regarded as the set of pairs (P_i, P_j) such that $P_iDP_j(P_iD'P_j)$). Although it seems intuitively obvious that a stronger dominance relation makes a branch-and-bound algorithm more efficient, counterexamples to this conjecture can be easily constructed as shown in [18] and in Section 4 of this paper, if the whole class of branch-and-bound algorithms is considered.

This paper, however, discovers four subclasses of branch-and-bound algorithms in which a stronger dominance relation always results in a more efficient algorithm in terms of measures T and B, the numbers of partial problems decomposed before the algorithm terminates and before an optimal solution is obtained, respectively. This would suggest that the monotonicity property of the power of dominance relations can be observed in a rather wide class of branch-and-bound algorithms. Thus the designer of a branch-and-bound algorithm is encouraged to find the strongest possible dominance relation for the class of problems to be solved (provided of course that the time required to calculate D does not become predominant).

2. *Constituents of Branch-and-Bound Algorithms*

The eight constituents of a branch-and-bound algorithm, i.e. $\mathcal{B}, f, O, g, \mathcal{G}, s, h$, and D, are introduced in this section. The construction of a branch-and-bound algorithm from these constituents will be described in section 3. Since all but D were implicitly or explicitly explained in papers such as [1, 3, 12, 14, 18, 22, 24, 29], they are only briefly sketched here.

Assume that we are asked to obtain an optimal (minimal) solution (or all optimal solutions) of problem P_0. A finite rooted tree $\mathcal{B} = (\mathcal{P}, \mathcal{E})$, where $\mathcal{P}$ is a set of nodes with *root* $P_0 \in \mathcal{P}$, and $\mathcal{E}$ is a set of arcs, represents the decomposition process of P_0 (assuming that all possible decompositions are applied); node $P_i \in \mathcal{P}$ corresponds to partial problem P_i and $(P_i, P_j) \in \mathcal{E}$ denotes that P_j is generated from P_i by a decomposition.[1] The set of *bottom nodes* (leaf nodes) of $\mathcal{B}$ is denoted $\mathcal{T}$. The *depth* of P_i, denoted $d(P_i)$, is the length of the path from P_0 to P_i in $\mathcal{B}$. Note that only a small subset of $\mathcal{P}$ is usually generated and tested prior to termination of a branch-and-bound algorithm.

Let $f : \mathcal{P} \to E \cup \{\infty\}$, where E is the set of real numbers, denote the objective values of optimal solutions of partial problems (nodes). f satisfies

$$f(P_i) = \min\{f(P_{i_j}) \mid j = 1, 2, \ldots, k\} \tag{2.1}$$

if $(P_i, P_{i_j}) \in \mathcal{E}$, $j = 1, 2, \ldots, k$. The set of optimal solutions of P_i is denoted $O(P_i)$. $O(P_i)$ satisfies

$$O(P_i) = \cup \{O(P_{i_j}) \mid f(P_i) = f(P_{i_j}), j = 1, 2, \ldots, k\}. \tag{2.2}$$

Note that our final goal is to obtain $f(P_0)$ and $O(P_0)$ (or an element in $O(P_0)$). (2.1) implies that

$$f(P_i) \leq f(P_{i_j}) \text{ for } (P_i, P_{i_j}) \in \mathcal{E}. \tag{2.3}$$

$(\mathcal{B}, O, f)$ is called the *branching structure* of P_0. Note that $f(P_i)$ is not known until P_i is completely solved.

A *lower bounding function* $g : \mathcal{P} \to E \cup \{\infty\}$ satisfies the following conditions.[2]

[1] In this case, P_j is called a *son* of P_i. P_j is called a *descendant* of P_i (P_i is an *ancestor* of P_j) if $P_j = P_i$ or P_j is a son of a descendant of P_i. A descendant (ancestor) P_j of P_i is *proper* if $P_i \neq P_j$.

[2] In some formalization of branch-and-bound algorithms (e.g. [12]) it is assumed that $g(P_i)$ is also dependent on other factors such as the incumbent value z (i.e. a more accurate bound is computed if the initial tentative $g(P_i)$ is smaller than but close to z) and the available computer time. This aspect is not considered in our definition.

(a) $g(P_i) \leq f(P_i)$ for $P_i \in \mathcal{P}$.
(b) $g(P_i) = f(P_i)$ for $P_i \in \mathcal{T}$.
(c) $g(P_j) \geq g(P_i)$ for $(P_i, P_j) \in \mathcal{E}$.

In a branch-and-bound algorithm, $g(P_i)$ is computed when P_i is generated. $\mathcal{G}$ ($\subset \mathcal{P}$) denotes the set of partial problems which are incidentally solved or proved to be infeasible[3] in the course of computation of g. $\mathcal{G}$ satisfies the following conditions.

(A) $g(P_i) = f(P_i)$ for $P_i \in \mathcal{G}$.
(B) $\mathcal{G} \supset \mathcal{T}$.
(C) $P_i \in \mathcal{G}$ implies $P_j \in \mathcal{G}$ if $(P_i, P_j) \in \mathcal{E}$.

Now let Π be the family of independent[4] sets in $\mathcal{P}$. $s : \Pi \to \mathcal{P}$ is a *search function* if $s(\mathcal{A}) \in \mathcal{A}$ for $\mathcal{A} \in \Pi$. s determines the order in which partial problems are selected for test. s is a *heuristic search function* based on $h : \mathcal{P} \to E$, if

$$h(s(\mathcal{A})) = \min\{h(P_i) \mid P_i \in \mathcal{A}\} \tag{2.4}$$

holds for $\mathcal{A} \in \Pi$. In this case, s is denoted s_h. In particular, $s = s_g$ is called the *best-bound search function*. Let

$$N(\mathcal{A}) = \{P_i \in \mathcal{A} \mid d(P_i) = \max\{d(P_j) \mid P_j \in \mathcal{A}\}\}. \tag{2.5}$$

Then the *depth-first search function* based on h, denoted $\bar{s}_h$, satisfies

$$h(\bar{s}_h(\mathcal{A})) = \min\{h(P_i) \mid P_i \in N(\mathcal{A})\}, \quad \bar{s}_h(\mathcal{A}) \in N(\mathcal{A}). \tag{2.6}$$

Finally a heuristic search function s_h is also called a *breadth-first search function* if $d(s_h(\mathcal{A})) = \min\{d(P_i) \mid P_i \in \mathcal{A}\}$ holds. This is realized if $d(P_i) < d(P_j)$ implies $h(P_i) < h(P_j)$ for $P_i, P_j \in \mathcal{P}$. It is known that heuristic search is the most general among these since it includes the other three as special cases [14].

We now turn to the last constituent of a branch-and-bound algorithm, the dominance relation. A binary relation $D \subset \mathcal{P} \times \mathcal{P}$ ($(P_i, P_j) \in D$ is also denoted P_iDP_j) is called a *dominance relation* if D satisfies the following conditions.

(i)$_a$ $P_iDP_j \wedge P_i \neq P_j$ imply $f(P_i) < f(P_j)$ and that P_i is not a proper descendant of P_j, in case all optimal solutions are sought,[5] or

(i)$_s$ P_iDP_j implies $f(P_i) \leq f(P_j)$, in case a single optimal solution is sought.[5]

(ii) D is a partial ordering, i.e. transitive ($P_iDP_j \wedge P_jDP_k \Rightarrow P_iDP_k$), reflexive ($P_iDP_i$), and antisymmetric ($P_iDP_j \wedge P_jDP_i \Rightarrow P_i = P_j$).

(iii) $P_iDP_j \wedge P_i \neq P_j$ imply that some descendant $P_{i'}$ of P_i satisfies $P_{i'}DP_{j'} \wedge P_{i'} \neq P_{j'}$ for any descendant $P_{j'}$ of P_j.

(iv)$_s$ In case a single optimal solution is sought, there exists no set of nodes ($\subset \mathcal{P}$),

$$P_{i_1}, P_{i_2}, \ldots, P_{i_{k+1}} \quad (k \geq 2 \text{ and } P_{i_1}, P_{i_2}, \ldots, P_{i_k} \text{ are distinct}), \tag{2.7}$$

generated in the branch-and-bound algorithm under consideration, such that (1) P_{i_t} is a proper descendant of $P_{i_{t+1}}$ or $P_{i_t}, P_{i_{t+1}}$ satisfy $P_{i_t}DP_{i_{t+1}} \wedge f(P_{i_t}) = f(P_{i_{t+1}})$, for $t = 1, 2, \ldots, k$, and (2) $P_{i_{k+1}} = P_{i_1}$ (i.e. a closed path). (For $k = 2$, this condition prohibits the case in which $P_iDP_j \wedge f(P_i) = f(P_j)$ holds for a proper descendant P_j of P_i.)

By condition (i) of D, it is obvious that P_j need not be solved if P_i is already generated and P_iDP_j holds; thus P_j can be terminated. A dominance relation D may be interpreted as an embodiment of the information on optimal solutions of partial problems obtainable without actually solving them (i.e. computing f).

The above definition of a dominance relation is different from the original one of [18] in that conditions (iii) and (iv)$_s$ are not assumed in [18], while one more condition, the consistency property with g (i.e. $P_iDP_j \Rightarrow g(P_i) \leq g(P_j)$), is assumed in [18]. Condition (iii) is satisfied by most dominance relations used in practice (see examples given below).

[3] If $P_i \in \mathcal{P}$ is infeasible, then $f(P_i) = \infty$ is assumed for convenience
[4] $\mathcal{A} \subset \mathcal{P}$ is *independent* if no P_i in $\mathcal{A}$ is a proper descendant of the others
[5] See the formal description of a branch-and-bound algorithm in Section 3

Condition $(iv)_s$ is necessary to prevent a deadlock in which all the nodes containing optimal solutions are terminated by dominance relation D, and as a result no optimal solution is obtained by the algorithm.[6] This condition, however, is not used in the subsequent discussion of this paper. The consistency property with g is also a quite natural assumption. It is assumed in many sections in the following.

Remark 2.1. Assume that a single optimal solution is sought. If $f(P_i) = f(P_j)$ is concluded for some reason, we can let either P_iDP_j or P_jDP_i. Note, however, that only one of them is possible by antisymmetry of condition (ii). A convenient way to make the resulting D a partial ordering is to let P_iDP_j if P_i is tested before P_j.

Some examples of dominance relations taken from the literature are given below. These examples assume that only a single optimal solution is sought (i.e. conditions $(i)_s$ and $(iv)_s$ hold). Many other dominance relations are also used in practical branch-and-bound algorithms (e.g. [2, 8, 10, 17, 23, 25-28, 31, 33]).

(A) *n-job two-machine mean finishing time flow-shop problem* [5, 16, 20]. Let a partial problem P specify a partial schedule on a subset $J(P)$ of n jobs, and let $F_{jk}(P)$ be the finishing time of job j ($\in J(P)$) on machine k (= 1 or 2) under the partial schedule specified by P. Then a dominance relation D may be defined as follows: P_sDP_t if and only if $J(P_s) = J(P_t)$, $\max_{j\in J(P_s)}F_{j2}(P_s) \leq \max_{j\in J(P_t)}F_{j2}(P_t)$ and $\sum_{j\in J(P_s)}F_{j2}(P_s) \leq \sum_{j\in J(P_t)} F_{j2}(P_t)$ (If equalities hold in the last two relations, we let P_sDP_t if P_s is tested before P_t, according to Remark 2.1.) D obviously satisfies conditions $(i)_s$ and (ii). It satisfies condition (iii) since, for a descendant $P_{t'}$ of P_t with $J(P_{t'}) = J(P_t) \cup S$, the partial problem $P_{s'}$, which is the schedule P_s followed by the set of jobs S scheduled in the same order as S of $P_{t'}$, satisfies $P_{s'}DP_{t'}$ as is easily shown. It is also possible to prove that condition $(iv)_s$ holds (though not given here since condition $(iv)_s$ is not used in our discussion).

(B) *n-job one-machine scheduling problem with deadlines* [32]. Each job j (= 1, 2, . . . , n) has a deadline d_j, penalty p_j, and processing time t_j, and pays penalty p_j if it is not completed by its deadline. Find a schedule on one machine which minimizes the total penalty. Let jobs be arranged such that $d_1 \leq d_2 \leq \cdots \leq d_n$. A partial problem P is defined by a positive integer $i(P)$ (satisfying $1 \leq i(P) \leq n$) and a subset $J(P) \subset \{1, 2, \ldots, i(P)\}$; P represents a partial schedule that first processes the jobs in $J(P)$ in increasing order and then processes the jobs not in $J(P)$ after their deadlines. Then it is possible to define that P_sDP_t if and only if $i(P_s) = i(P_t)$, $\sum_{j\in J(P_s)}t_j \leq \sum_{j\in J(P_t)}t_j$ and $\sum_{j\notin J(P_s),1\leq j\leq i(P_s)}p_j \leq \sum_{j\notin J(P_t),1\leq j\leq i(P_t)}p_j$. (If equalities hold in the last two relations, let P_sDP_t if P_s is tested before P_t.)

(C) *Problem of eight queens* [9]. A partial problem P represents a pattern of m (≤ 8) queens put on a chessboard of size 8×8. Then P_sDP_t if and only if the patterns represented by P_s and P_t are isomorphic and P_s is tested before P_t.

(D) *Shortest path problem* [15]. Find the shortest path from node 1 to node n on an n-node network with nonnegative arc lengths. A partial problem P represents path $\pi(P)$ from node 1 to node $i(P)$ ($1 \leq i(P) \leq n$). Then P_sDP_t if and only if $i(P_s) = i(P_t)$ and the length of $\pi(P_s)$ is not greater than that of $\pi(P_t)$ (P_s is tested before P_t if their lengths are equal).

It is not difficult to regard the Dijkstra algorithm for the shortest path problem [6, 7] as a branch-and-bound algorithm with the above dominance relation (D) [15], though it was not originally given in the framework of branch-and-bound. This point may be extended to most other dynamic programming algorithms based on the principle of optimality [4], as pointed out by [19, 27]. (In [19], the dynamic programming algorithm for the traveling salesman problem [13] is formulated as a branch-and-bound algorithm. [27] contains a further general discussion.) Thus it should be noted that a dominance relation adds another dimension of flexibility in designing a branch-and-bound algorithm, sometimes enabling us to exploit a special structure of a given problem to improve the efficiency of the resulting algorithm.

[6] Reference [18] uses a different mechanism to prevent a deadlock

3. *Branch-and-Bound Algorithms: General Description*

In this section the constituents discussed previously are assembled into a branch-and-bound algorithm. Two types of algorithms are given; the first one obtains all optimal solutions of a given problem P_0, while the second one obtains only a single optimal solution.

Branch-and-bound algorithm $A_a = ((\mathcal{B}, O, f), (\mathcal{G}, g), D, s)$. *All optimal solutions.*

Remark. $\mathcal{N} \subset \mathcal{P}$ denotes the set of the partial problems currently generated. A node in $\mathcal{N}$ is *active* if it is yet neither tested nor decomposed into smaller partial problems. $\mathcal{A}$ denotes the set of currently active nodes, which is always independent as easily proved. $\mathcal{O}$ denotes the set of the best feasible solutions currently available and is called the *incumbent*; z is its objective value (incumbent value). Upon termination, $\mathcal{O}$ stores $O(P_0)$ and z stores its objective value. It is assumed that $O(P_i)$ is calculated as a by-product of testing P_i in case $P_i \in \mathcal{G}$ holds.

$A1_a$ (Initialize)[7]: $\mathcal{A} \leftarrow \{P_0\}$, $\mathcal{N} \leftarrow \{P_0\}$, $z \leftarrow \infty$, $\mathcal{O} \leftarrow \emptyset$ (empty)

$A2_a$ (Search): If $\mathcal{A} = \emptyset$, go to $A9_a$; otherwise let $P_i \leftarrow s(\mathcal{A})$ and go to $A3_a$

$A3_a$ ($\mathcal{G}$ test). If $P_i \in \mathcal{G}$, go to $A7_a$, otherwise go to $A4_a$.

$A4_a$ (Lower bound test) If $g(P_i) > z$, go to $A8_a$, otherwise go to $A5_a$

$A5_a$ (Dominance test): If there exists $P_k (\neq P_i) \in \mathcal{N}$ such that $P_k D P_i$, go to $A8_a$; otherwise go to $A6_a$

$A6_a$ (Decompose): Generate sons $P_{i_1}, P_{i_2}, \ldots, P_{i_k}$ of P_i Return to $A2_a$ after letting $\mathcal{A} \leftarrow \mathcal{A} \cup \{P_{i_1}, P_{i_2}, \ldots, P_{i_k}\} - \{P_i\}$ and $\mathcal{N} \leftarrow \mathcal{N} \cup \{P_{i_1}, P_{i_2}, \ldots, P_{i_k}\}$.

$A7_a$ (Improve): Go to $A8_a$ after letting

$$\mathcal{O} \leftarrow \begin{cases} \mathcal{O} & \text{if } z < f(P_i)\ (= g(P_i)), \\ \mathcal{O} \cup O(P_i) & \text{if } z = f(P_i), \\ O(P_i) & \text{if } z > f(P_i); \end{cases}$$
$$z \leftarrow \min[z, f(P_i)].$$

$A8_a$ (Terminate P_i)[8]: Return to $A2_a$ after letting $\mathcal{A} \leftarrow \mathcal{A} - \{P_i\}$.

$A9_a$ (Halt): Halt $O(P_0) = \mathcal{O}$ and $f(P_0) = z$; P_0 is infeasible if $\mathcal{O} = \emptyset$.

The finiteness and the validity of the above procedure are not proved here since proofs for similar procedures may be found in survey papers such as [3, 18, 22, 24, 29]. Note that z decreases monotonically from its initial value $z = \infty$ to the final value $z = f(P_0)$, as the computation proceeds.

At this point it may be interesting to note[9] that the lower bound test can be regarded as a special case of the dominance test (with a certain modification); step $A4_a$ is equivalent to step $A5_a$ with $\mathcal{N}$ in the statement replaced by $\mathcal{N} - \mathcal{A}$ and with dominance relation D defined by $P_k D P_i$ if $P_k \in \mathcal{G}$ and $g(P_k)(= f(P_k)) < g(P_i)$.

The above algorithm is slightly modified if only a single optimal solution of P_0 is sought.

Branch-and-bound algorithm $A_s = ((\mathcal{B}, O, f), (\mathcal{G}, g), D, s)$. *Single optimal solution.*

Remark. It is assumed that an element in $O(P_i)$ is calculated as a by-product of testing P_i in case $P_i \in \mathcal{G}$ holds.

$A1_s$, $A2_s$, $A3_s$: Same as $A1_a$, $A2_a$, $A3_a$ with $A\iota_a$ in their statements replaced by $A\iota_s$, respectively.

$A4_s$: If $g(P_i) \geq z$, go to $A8_s$, otherwise go to $A5_s$

$A5_s$, $A6_s$: Same as $A5_a$, $A6_a$ with $A\iota_a$ replaced by $A\iota_s$, respectively.

$A7_s$: Go to $A8_s$ after letting

$$\mathcal{O} \leftarrow \begin{cases} \mathcal{O} & \text{if } z \leq f(P_i), \\ \{x\}, & \text{where } x \in O(P_i), \text{ otherwise,} \end{cases}$$
$$z \leftarrow \min[z, f(P_i)].$$

---

[7] $\leftarrow$ stands for the assignment operation represented by := in Algol.

[8] P_i is called *terminated* if $P_i \in \mathcal{G}$ in $A3_a$, $g(P_i) > z$ in $A4_a$, or $P_k D P_i$ for some $P_k \in \mathcal{N}$ in $A5_a$

[9] Due to W.H Kohler (private communication)

$A8_s$, $A9_s$: Same as $A8_a$, $A9_a$ with At_a replaced by At_s and with $O(P_0) = \mathcal{O}$ replaced by $\mathcal{O}(P_0)$ in A9.

Throughout this paper, the efficiency of a branch-and-bound algorithm A is measured by the following two parameters.

$T(A)$: The number of nodes decomposed in $A6_a$ ($A6_s$) prior to termination $A9_a$ ($A9_s$).

$B(A)$: The number of nodes decomposed in $A6_a$ ($A6_s$) prior to the last modification of $\mathcal{O}$ occurred in $A7_a$ ($A7_s$).

$T(A)$ is closely related to the total computation time of A and has been one of the most popular measures (e.g. [18, 30]). $B(A)$ is related to the computation time required until all optimal solutions (a single optimal solution if $A = A_s$) are stored in $\mathcal{O}$. This is important in practical applications in which the computation may be cut off prior to termination $A9_a$ (or $A9_s$) due to the insufficiency of the available computer time. Obviously it is desirable to design a branch-and-bound algorithm with smaller $T(A)$ and $B(A)$.

It should be noted here that $T(A)$ (or $B(A)$) does not always reflect the exact computation time actually required since $T(A)$ may be made small at the cost of increasing the time required for testing each partial problem. To know the behavior of $T(A)$ and $B(A)$ at least provides a useful guideline for designing an efficient branch-and-bound algorithm, however, since the time required for testing a partial problem can usually be estimated more accurately than the number of nodes $T(A)$ or $B(A)$. Furthermore, the actual computation time does not seem to be a measure which is theoretically tractable.

In the subsequent discussion, subscripts a and s are added, e.g. A_a, A_s, At_s, $T_a(A)$, and $B_a(A)$, to distinguish algorithms for all optimal solutions and a single optimal solution, respectively. Conversely, no subscript is added if it is not necessary to distinguish them; A refers to either algorithm A_a or algorithm A_s, $T(A)$ to either $T_a(A)$ or $T_s(A)$, and so forth.

4. *Power of Dominance Relations*

Let D and D' be dominance relations on $\mathcal{P}$. If

$$D' \supset D \quad (\text{i.e. } P_iDP_j \text{ implies } P_iD'P_j),$$

D' is said to be *stronger* than D. In view of the motivation for introducing dominance relations, it seems intuitively obvious that a stronger dominance relation makes the resulting branch-and-bound algorithm more efficient: Branch-and-bound algorithms $A = ((\mathcal{B}, O, f), (\mathcal{G}, g), D, s)$ and $A' = ((\mathcal{B}, O, f), (\mathcal{G}, g), D', s)$ with $D' \supset D$ always satisfy $T(A') \leq T(A)$ and $B(A') \leq B(A)$. This monotonicity property of dominance relations, however, does not generally hold (as first observed in [18] under a somewhat different assumption on dominance relations). In this section it is shown that the monotonicity property is not generally observed for branch-and-bound algorithms with heuristic search functions (note that heuristic search is the most general among search strategies as mentioned in Section 2). However, such monotonicity is guaranteed if a search function belongs to one of the following four special classes: Class of heuristic search functions with some additional property (Section 5), class of best-bound search functions with D satisfying the consistency property with g (Section 6), class of breadth-first search functions on some restricted branching structures (Section 7), and class of depth-first search functions with minor modifications (Section 8).

THEOREM 4.1. *Let $A = ((\mathcal{B}, O, f), (\mathcal{G}, g), D, s_h)$ and $A' = ((\mathcal{B}, O, f), (\mathcal{G}, g), D', s_h)$ be branch-and-bound algorithms with a heuristic search function s_h, and let $D' \supset D$. This does not generally imply $T_a(A') \leq T_a(A)$, $T_s(A') \leq T_s(A)$, $B_a(A') \leq B_a(A)$, or $B_s(A') \leq B_s(A)$.*

PROOF. Consider the branching structure $(\mathcal{B}, O, f)$ and $h : \mathcal{P} \to E$ shown in Figure 1 ($O(P_i)$ is not indicated since it is not relevant to T or B). For simplicity, it is assumed that

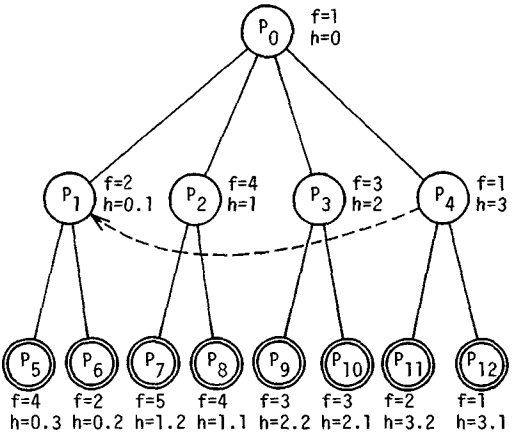


FIG. 1. Branching structure $(\mathcal{B}, O, f)$ and h used in the proof of Theorem 4.1 (A broken arrow indicates dominance relation D'. It is assumed that $g = f$ and $\mathcal{G} = \mathcal{T}$.)

$\mathcal{G} = \mathcal{T}$ and $g(P_i) = f(P_i)$ for each $P_i \in \mathcal{P}$. Let D' and D be given by

$$D = \{(P_i, P_i) \mid P_i \in \mathcal{P}\} \quad (D \text{ is the identity relation}),$$
$$D' = D \cup \{(P_4, P_1)\} \cup \{(P_i, P_j) \mid P_i, P_j \in \mathcal{G}, f(P_i) < f(P_j)\}.$$

In other words, D' is the identity relation augmented with (P_4, P_1) (indicated in Figure 1 by a broken arrow) and those defined on nodes in $\mathcal{G}$. Obviously $D' \supset D$. Computation processes of A and A' are illustrated in Figure 2 (a) and (b), respectively, in which node numbers indicate the order of testing the generated nodes, and z indicates the incumbent value when the corresponding node is selected in step A2. Note that P_1 of Figure 2 (b) is terminated in step A5 by dominance relation $P_4 D' P_1$. From Figure 2 (a, b) it follows that

$$T_a(A) = T_s(A) = B_a(A) = B_s(A) = 3,$$
$$T_a(A') = T_s(A') = B_a(A') = B_s(A') = 4. \qquad \text{Q.E.D.}$$

Note that s_h in the above A and A' is a depth-first search function. Thus the monotonicity of dominance relations does not hold even if branch-and-bound algorithms are restricted to those with depth-first search functions.

5. *Nonmisleading Heuristic Search*

A heuristic function $h : \mathcal{P} \to E$ is called *nonmisleading* if

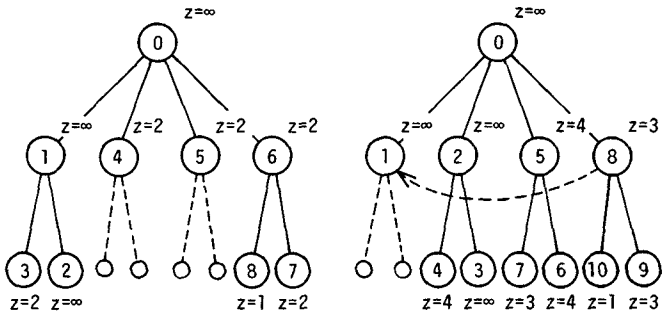
$$h(P_i) < h(P_j) \text{ implies } f(P_i) \leq f(P_j) \text{ for } P_i, P_j \in \mathcal{P}. \tag{5.1}$$

It is known that a branch-and-bound algorithm with a nonmisleading heuristic function is most efficient [11, 14]. Although it is not reasonable to assume that such a heuristic function is easily obtainable (since it requires the complete knowledge of f), it is considered as a theoretical goal when we design a heuristic function for a branch-and-bound algorithm. In addition, it is shown in [14] that a heuristic function which is close to nonmisleading always makes the performance of the resulting algorithm close to the most efficient one. Thus the investigation of the effect of dominance relations on algorithms with nonmisleading heuristic functions would help one to understand the behavior of branch-and-bound algorithms which are very successfully designed by using almost nonmisleading heuristic functions.

In the following we assume for simplicity (but without loss of generality) that

$$h(P_i) \neq h(P_j) \text{ for } P_i \neq P_j \tag{5.2}$$
(by using an appropriate tie breaking rule if necessary),

$$h(P_i) > h(P_j) \quad \text{if } P_i \text{ is a proper descendant of } P_j. \tag{5.3}$$



(a) Computation process by A (b) Computation process by A'

FIG. 2 Computation processes by branch-and-bound algorithms A and A' used in the proof of Theorem 4 1

(Given a heuristic function h' it is possible to prove the existence of a heuristic function h such that h satisfies (5.2) and (5.3) and $s_h = s_{h'}$ [14].)

LEMMA 5.1. *Let $A = ((\mathcal{B}, O, f), (\mathcal{G}, g), D, s_h)$ be a branch-and-bound algorithm, and let $\mathcal{S} = P_{i_1}P_{i_2} \ldots P_{i_s}$ (where $P_{i_1} = P_0$) be the sequence of the generated nodes arranged in the order of selection in step A2. Then $h(P_{i_k}) < h(P_{i_l})$ holds for $k < l$ ($\leq s$). In addition, $f(P_{i_k}) \leq f(P_{i_l})$ holds for $k < l$ ($\leq s$) if h is nonmisleading.*

PROOF. Assume that P_i and P_j are generated in A and $h(P_i) < h(P_j)$. Then any ancestor P_k of P_i satisfies $h(P_k) < h(P_i) < h(P_j)$ by (5.3). This shows that P_k and hence P_i are already generated when P_j is selected. Thus P_i is selected prior to P_j since $h(P_i) < h(P_j)$, proving the first half. The second half is immediate from the definition of a nonmisleading heuristic function. Q.E.D.

Before proving the next theorem, one more definition is introduced. A dominance relation D is *consistent with* g if

(v) P_iDP_j implies (1) $g(P_i) < g(P_j)$, or (2) $g(P_i) = g(P_j)$ and P_i is selected before P_j.

THEOREM 5.2. *Let $A = ((\mathcal{B}, O, f), (\mathcal{G}, g), D, s_h)$ and $A' = ((\mathcal{B}, O, f), (\mathcal{G}, g), D', s_h)$ be branch-and-bound algorithms based on a nonmisleading heuristic function h. If $D' \supset D$ and D' is consistent with g, then it holds that $T_a(A') \leq T_a(A)$ and $B_a(A') \leq B_a(A)$.*

PROOF. Let $\mathcal{S} = P_{i_1}P_{i_2} \ldots P_{i_s}$ and $\mathcal{S}' = P_{j_1}P_{j_2} \ldots P_{j_t}$ be the sequences of nodes selected in A and A', respectively. We show that $\mathcal{S}'$ is a subsequence of $\mathcal{S}$ when all optimal solutions are sought. Let $\mathcal{A}(P_i)$ and $\mathcal{A}'(P_i)$ denote the sets of active nodes, and $z(P_i)$ and $z'(P_i)$ denote the incumbent values, when P_i is selected in A and A' respectively.

To use induction, first note that $P_{i_1} = P_{j_1}$ ($= P_0$) and $\mathcal{A}(P_{i_2}) = \mathcal{A}'(P_{j_2})$. Then assume that $\mathcal{S}'_q = P_{j_1}P_{j_2} \ldots P_{j_q}$ ($q < t$; if $q = t$, the proof is done) is a subsequence of $\mathcal{S}_p = P_{i_1}P_{i_2} \ldots P_{i_p}$ ($p < s$) and $\mathcal{A}'(P_{j_{q+1}}) \subset \mathcal{A}(P_{i_{p+1}})$. Two cases are possible.

(a) $P_{i_{p+1}} \neq P_{j_{q+1}}$. Then $P_{i_{p+1}} \in \mathcal{A}(P_{i_{p+1}})$ but $P_{i_{p+1}} \notin \mathcal{A}'(P_{j_{q+1}})$ since $h(P_{i_{p+1}}) < h(P_j)$ for each P_j ($\neq P_{i_{p+1}}$) $\in \mathcal{A}'(P_{j_{q+1}})$ ($\subset \mathcal{A}(P_{i_{p+1}})$) by the definition of heuristic search. $P_{i_{p+1}}$ may then be terminated in $A3_a$, $A4_a$, $A5_a$, or decomposed in $A6_a$. In either case, we have two sequences, $\mathcal{S}'_q = P_{j_1}P_{j_2} \ldots P_{j_q}$ and $\mathcal{S}_{p+1} = P_{i_1}P_{i_2} \ldots P_{i_p}P_{i_{p+1}}$, such that $\mathcal{S}'_q$ is a subsequence of $\mathcal{S}_{p+1}$ and $\mathcal{A}'(P_{j_{q+1}}) \subset \mathcal{A}(P_{i_{p+2}})$.

(b) $P_{i_{p+1}} = P_{j_{q+1}}$. First note that either both $P_{i_{p+1}}$ and $P_{j_{q+1}}$ are terminated or none of them are terminated in step $A3_a$ ($\mathcal{G}$ test).

To consider step $A4_a$ (lower bound test), note that Lemma 5.1 implies that the first incumbent P_k ($\in \mathcal{G}$) obtained in A or A' satisfies $f(P_k) = f(P_0)$. z or z' (initially ∞) is then set to $f(P_0)$ and keeps the same value thereafter. From this observation $z(P_{i_{p+1}}) = z'(P_{j_{q+1}})$ is proved as follows. First if $z(P_{i_{p+1}}) = \infty$ then $z'(P_{j_{q+1}}) = \infty$ since $\mathcal{S}'_q$ is a subsequence of $\mathcal{S}_p$. Second if $z(P_{i_{p+1}}) = f(P_0)$ then some P_{i_v} ($1 \leq v \leq p$) in $\mathcal{S}_p$ satisfies $P_{i_v} \in \mathcal{G}$ and $f(P_{i_v}) = f(P_0)$. Since any ancestor P_{i_u} of P_{i_v} satisfies $P_{i_u} \notin \mathcal{G}$, $g(P_{i_u}) \leq f(P_{i_u}) = f(P_0)$ holds and

no $P_a \in \mathcal{P}$ satisfies $P_a D P_{i_u}$ (see condition (i)$_a$ of a dominance relation), P_{i_u} is not terminated in step A3$_a$, A4$_a$, or A5$_a$ of A'. Thus $\mathcal{A}'(P_{j_{q+1}}) \subset \mathcal{A}(P_{i_{p+1}})$ (i.e. $P_{i_u}, P_{i_v} \notin \mathcal{A}'(P_{j_{q+1}})$) implies that P_{i_v} has also been selected in $\mathcal{S}'_q$. Thus $z'(P_{j_{q+1}}) = f(P_0)$. $z(P_{i_{p+1}}) = z(P_{i_{q+1}})$ and $g(P_{i_{p+1}}) = g(P_{j_{q+1}})$ (since $P_{i_{p+1}} = P_{j_{q+1}}$) then implies that $P_{i_{p+1}}$ is terminated in step A4$_a$ of A if and only if $P_{j_{q+1}}$ is terminated in step A4$_a$ of A'.

Next we turn to step A5$_a$ (dominance test). Assume that $P_{i_{p+1}}$ is terminated in A5$_a$ of A, i.e. $P_{i_r} D P_{i_{p+1}}$ holds for some $r \leq p$ (note that $P_{i_r} D P_{i_{p+1}} \Rightarrow f(P_{i_r}) < f(P_{i_{p+1}}) \Rightarrow h(P_{i_r}) < h(P_{i_{p+1}}) \Rightarrow r < p+1$ (by Lemma 5.1)). If $P_{i_r} = P_{j_d}$ for some $d \leq q$, we have $P_{j_d} D' P_{j_{q+1}}$ by $D' \supset D$. Thus $P_{j_{q+1}}$ is also terminated in A5$_a$ of A'. On the other hand, if P_{i_r} is not in $\mathcal{S}'_q$, a proper ancestor P_{j_v} ($v \leq q$) of P_{i_r} (in $\mathcal{B}$) must have been tested and terminated in step A5$_a$ of A'. (See Figure 3.) (Note that $P_{j_v} = P_{i_a}$ for some $a < r$ since $\mathcal{S}'_q$ is a subsequence of $\mathcal{S}_p$. P_{i_a} was not terminated in A since a proper descendent P_{i_r} is generated. Thus P_{j_v} ($= P_{i_a}$) is not terminated in step A3$_a$ or step A4$_a$ of A'.) This shows that there exists P_{j_w} in $\mathcal{S}'_q$ such that $P_{j_w} D' P_{j_v}$ as shown in Figure 3. P_{j_w} has a descendent P_{j_u} satisfying $P_{j_u} D' P_{i_r}$ by condition (iii). First assume that P_{j_u} is generated in A'. Then $P_{j_u} D' P_{j_{q+1}}$ follows from transitivity (see condition (ii)), and P_{j_u} has already been generated when $P_{j_{q+1}}$ is tested since $P_{j_u} D' P_{j_{q+1}} \Rightarrow f(P_{j_u}) < f(P_{j_{q+1}}) \Rightarrow h(P_{j_u}) < h(P_{j_{q+1}})$. Thus $P_{j_{q+1}}$ is also terminated in A'. On the other hand, if P_{j_u} is not generated in A' (i.e. a proper ancestor is terminated in A5$_a$ of A' since, if a proper ancestor P_{j_y} of P_{j_u} is terminated in A4$_a$, we have $z'(P_{j_{q+1}}) \leq z'(P_{j_y}) < g(P_{j_y}) \leq g(P_{j_u}) \leq g(P_{j_{q+1}})$ (by the consistency of D with g) and $P_{j_{q+1}}$ is also terminated in A4$_a$, a contradiction), repeat the same argument. We will have a sequence of nodes $P_{k_1}(= P_{j_{q+1}})$, $P_{k_2}(= P_{i_r})$, $P_{k_3}(= P_{j_v})$, P_{k_4}, . . . such that . . , $P_{k_4} D' P_{k_3} \wedge P_{k_4} \neq P_{k_3}$, $P_{k_3} D' P_{k_2} \wedge P_{k_3} \neq P_{k_2}$, $P_{k_2} D' P_{k_1} \wedge P_{k_2} \neq P_{k_1}$. Note that all nodes $P_{k_1}, P_{k_2}, \ldots$ are distinct since otherwise D' is not a partial ordering. Therefore, this process does not continue indefinitely since $\mathcal{B}$ has only finite nodes, showing again that $P_{j_{q+1}}$ is terminated in A'.

The above argument proves that $P_{j_{q+1}}$ is terminated in A' if $P_{i_{p+1}}$ is terminated in A. Thus we have $\mathcal{S}'_{q+1} = P_{j_1} P_{j_2} \ldots P_{j_q} P_{j_{q+1}}$ and $\mathcal{S}_{p+1} = P_{i_1} P_{i_2} \ldots P_{i_p} P_{i_{p+1}}$ such that $\mathcal{S}'_{q+1}$ is a subsequence of $\mathcal{S}_{p+1}$ and $\mathcal{A}'(P_{j_{q+2}}) \subset \mathcal{A}(P_{i_{p+2}})$.

By repeating this induction step (a) or (b), we will eventually reach $\mathcal{S}'$ and $\mathcal{S}$ such that $\mathcal{S}'$ is a subsequence of $\mathcal{S}$. This proves $T_a(A') \leq T_a(A)$ and $B_a(A') \leq B_a(A)$. Q.E.D.

THEOREM 5.3. *Let A and A' be defined as in Theorem 5.2. If $D' \supset D$ and D' is consistent with g, then it holds that*

$$T_s(A') \leq T_s(A) + |\mathcal{J} \cap \mathcal{K}|, \quad B_s(A') \leq B_s(A) + |\mathcal{J} \cap \mathcal{K}|,$$

where

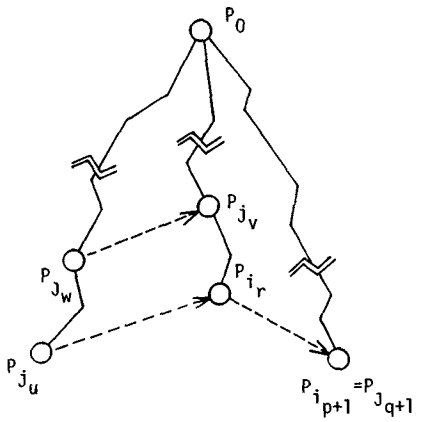


FIG. 3. Relations of nodes used in the proof of Theorem 5.2. (Broken arrows indicate dominance relation D'.)

$$\mathcal{J} = \{P_i \in \mathcal{P} \mid f(P_i) = f(P_0) \wedge P_i \notin \mathcal{G}\},^{10} \tag{5.4}$$

$$\mathcal{K} = \{P_i \in \mathcal{P} \mid g(P_i) = f(P_0)\}.^{10} \tag{5.5}$$

PROOF. When a single optimal solution is sought by A or A', the proof of Theorem 5.2 should be slightly changed since P_iDP_j implies only $f(P_i) \leq f(P_j)$ (rather than $f(P_i) < f(P_j)$) and step $A4_s$ is active if $g(P_i) \geq z$ (rather than $g(P_i) > z$). Thus a node P_j with $f(P_j) = f(P_0)$ may possibly be terminated in step $A5_s$ or in step $A4_s$ if P_iDP_j (or $P_iD'P_j$) or $g(P_j) = f(P_j) = z(= f(P_0))$ holds, respectively.

To see how those nodes P_i with $f(P_i) = f(P_0)$ are treated in A or A', first note that they are located in the initial portions of $\mathcal{S}$ and $\mathcal{S}'$, respectively, by Lemma 5.1. Denote such portions consisting of nodes P_i with $f(P_i) = f(P_0)$ by $\mathcal{S}_I$ and $\mathcal{S}'_I$, respectively. Let $\mathcal{S}^* = P_{k_1}P_{k_2} \ldots P_{k_a}$ be the sequence of all nodes $P_k \in \mathcal{P}$ with $f(P_k) = f(P_0)$ arranged in the increasing order of h. By Lemma 5.1, $\mathcal{S}_I$ and $\mathcal{S}'_I$ are subsequences of $\mathcal{S}^*$. Let P_{k_b} be the first node satisfying

$$P_{k_b} \in \mathcal{G} \text{ and } P_{k_b} \text{ appears in } \mathcal{S}'_I. \tag{5.6}$$

Then there exists P_{k_c} $(c \leq b)$ such that

$$P_{k_c} \in \mathcal{G} \text{ and } P_{k_c} \text{ appears in } \mathcal{S}_I, \tag{5.7}$$

as proved below. If (5.7) is false, $z = \infty$ holds in A until some $P_{k_d} \in \mathcal{G}$ such that $d > b$ is selected. Thus a proper ancestor of each P_{k_c} satisfying $P_{k_c} \in \mathcal{G}$ and $c \leq b$ must have been terminated in step $A5_s$ of A. By considering $D' \supset D$ and the case of $c = b$, this implies that a proper ancestor of P_{k_b} is also terminated in A' (apply an argument similar to the last half of case (b) in the proof of Theorem 5.2). This is a contradiction to (5.6).

From (5.6) and (5.7), it is possible for a node P_i with $f(P_i) = f(P_0)$ to be decomposed in A' but terminated in A (by step $A4_s$) if

$$f(P_i) = f(P_0) = g(P_i) \text{ and } P_i \notin \mathcal{G}, \tag{5.8}$$

since the incumbent value z in A may possibly be set to $f(P_0)$ earlier than z' in A'. Once $z(P_{i_u}) = z'(P_{j_v}) = f(P_0)$ holds for $P_{i_u} = P_{j_v}$ (in $\mathcal{S}$ and $\mathcal{S}'$, respectively), however, an argument similar to the proof of Theorem 5.2 can be applied to the rest portions of $\mathcal{S}$ and $\mathcal{S}'$; any P_i (in that portion of $\mathcal{S}'$) decomposed in A' is also decomposed in A. Consequently any P_i which is decomposed in A' but not decomposed in A satisfies (5.8). This proves $T_s(A') \leq T_s(A) + |\mathcal{J} \cap \mathcal{K}|$ and $B_s(A') \leq B_s(A) + |\mathcal{J} \cap \mathcal{K}|$. Q.E.D.

It should be noted that $|\mathcal{J} \cap \mathcal{K}|$ is usually very small.

COROLLARY 5.4. *Assume that there exists exactly one path in $\mathcal{B}$ from P_0 to $P_j \in \mathcal{G}$ such that $f(P_j) = f(P_0)$. Then $T_s(A') \leq T_s(A)$, $B_s(A') \leq B_s(A)$ holds in Theorem* 5.3.

PROOF. Obvious since $\mathcal{S}_I = \mathcal{S}'_I = \mathcal{S}^* = P_{k_1}P_{k_2} \ldots P_{k_e}$, where $P_{k_1} = P_0$, $P_{k_e} = P_j$ corresponds to such a path. Q.E.D.

6. *Best-Bound Search*

In this section it is shown that the monotonicity property of dominance relations is also observed for branch-and-bound algorithms using best-bound search functions. Two lemmas are first proved. Essentially the same property as Lemma 6.1 was also proved in [21].

LEMMA 6.1. *Let $A = ((\mathcal{B}, O, f), (\mathcal{G}, g), D, s_g)$ be a branch-and-bound algorithm with best-bound search. Let $\mathcal{S} = P_{i_1}P_{i_2} \ldots P_{i_s}$ be the sequence of nodes arranged in the order of selection in step A2. Then $g(P_{i_k}) \leq g(P_{i_l})$ holds for $k < l$ $(\leq s)$.*

PROOF. Similar to the proof of Lemma 5.1, with h replaced by g. (Also note condition (c) of g in Section 2). Q.E.D.

LEMMA 6.2. *A node $P_i \in \mathcal{P}$ may be decomposed in $A = ((\mathcal{B}, O, f), (\mathcal{G}, g), D, s_g)$ only if $P_i \in \mathcal{F} - \mathcal{G}$ holds, where*

[10] Note that $f(P_0)$ is the optimal value of the original problem P_0

$$\mathcal{F} = \{P_i \in \mathcal{P} \mid g(P_i) \leq f(P_0)\} . \tag{6.1}$$

PROOF. It is obvious from Lemma 6.1 that the first $P_i \in \mathcal{G}$ selected in A satisfies $f(P_i) = g(P_i) = f(P_0)$. Thus a node P_j with $P_j \in \mathcal{G}$ or $g(P_j) > f(P_0)$ is always terminated in step A3 or step A4, since $z = f(P_0)$ holds by Lemma 6.1 when P_j with $g(P_j) > f(P_0)$ is selected. Q.E.D.

Note that Lemmas 6.1 and 6.2 hold for both cases of all optimal solutions and a single optimal solution.

THEOREM 6.3. *Let* $A = ((\mathcal{B}, O, f), (\mathcal{G}, g), D, s_g)$ *and* $A' = ((\mathcal{B}, O, f), (\mathcal{G}, g), D', s_g)$ *be branch-and-bound algorithms with best-bound search, where* D' *satisfies condition* (v). *If* $D' \supset D$, *then*

$$T_a(A') \leq T_a(A),\ B_a(A') \leq B_a(A),$$
$$T_s(A') \leq T_s(A) + |\mathcal{K} - \mathcal{G}|,\ B_s(A') \leq B_s(A) + |\mathcal{K} - \mathcal{G}| .$$

($\mathcal{K}$ *was defined in* (5.5).)

PROOF. First consider the case of all optimal solutions. By Lemma 6.2, assume that $P_i \in \mathcal{F} - \mathcal{G}$ is not decomposed in A. Since $g(P_i) \leq f(P_0) \leq z$, step $A4_a$ (lower bound test) is not active for P_i and hence P_i must be terminated in step $A5_a$ (dominance test), i.e. there exist $P_k \in \mathcal{N}(P_i)$ such that $P_k D P_i$, where $\mathcal{N}(P_i)$ denotes the set of the generated nodes when P_i is selected in step $A2_a$. Now assume that the same P_i is decomposed in A'. If $P_k \in \mathcal{N}'(P_i)$, where $\mathcal{N}'$ is similarly defined, then $P_k D' P_i$ by $D' \supset D$, a contradiction. Thus let $P_k \notin \mathcal{N}'(P_i)$. Then a proper ancestor P_j of P_k must have been terminated in $A5_a$ of A' by $P_l D' P_j$ for some $P_l \in \mathcal{N}'(P_j)$. (Note step $A4_a$ is not active for P_j since $g(P_j) \leq g(P_k) \leq f(P_0)$.) By condition (iii), P_l has a descendant P_t such that $P_t D' P_k$ (and hence $P_t D' P_i$), where $P_t \in \mathcal{N}'(P_k)$ by condition (v), if P_t is generated in A'. Thus P_i is terminated in $A5_a$ of A', again a contradiction. If a proper ancestor of P_t is terminated in A', repeat the same argument; this process does not continue indefinitely since $\mathcal{B}$ has only finite nodes. Consequently it is proved that P_i can be decomposed in A' only if it is decomposed in A. This and Lemmas 6.1 and 6.2 show that $\mathcal{S}'$ is a subsequence of $\mathcal{S}$, where $\mathcal{S}(\mathcal{S}')$ is the subsequence of nodes arranged in the order of selection in $A2_a$ of $A(A')$.[11] Thus $T_a(A') \leq T_a(A)$ and $B_a(A') \leq B_a(A)$.

When only a single optimal solution is sought, a slight modification similar to the proof of Theorem 5.3 is necessary. A stronger dominance relation may tend to delay the time of obtaining the first incumbent solution $P_j \in \mathcal{G}$. Thus a node P_i may be terminated in $A4_s$ of A, but not in $A4_s$ of A' if

$$g(P_i) = f(P_0), \quad P_i \notin \mathcal{G} \tag{6.2}$$

and $z(P_i) = f(P_0)$ hold. The number of such nodes is at most $|\mathcal{K} - \mathcal{G}|$. This proves the results for T_s and B_s. Q.E.D.

$|\mathcal{K} - \mathcal{G}|$ is usually very small. It is actually zero, if for example $g(P_j) \neq g(P_i)$ holds for $P_i, P_j \in \mathcal{P} - \mathcal{G}$.

7. *Breadth-First Search*

In this section we consider a special class of branching structures such that

$$P_i \in \mathcal{T} \text{ (i.e. } P_i \text{ is a bottom node)} \Leftrightarrow d(P_i) = n \tag{7.1}$$

for some positive integer n, and assume that

$$\mathcal{G} = \mathcal{T}.^{12} \tag{7.2}$$

We further assume that D satisfies the following condition in addition to conditions (i)–(iv) of Section 2.

[11] Here we assume that the same tie-breaking rule is used to define s_g in A and A', when $g(P_i) = g(P_j)$ holds for $P_i \neq P_j$.

[12] This condition can be relaxed to: $P_i \in \mathcal{G} - \mathcal{T}$ implies $f(P_i) = \infty$ (i.e. P_i is infeasible).

(vi) P_iDP_j implies $d(P_i) \leq d(P_j)$.

The above conditions are satisfied in many practical problems. For example, dominance relations (A) and (B) discussed in Section 2 satisfy condition (vi).

It is not difficult to show that if a branch-and-bound algorithm with breadth-first search is applied to a problem satisfying (7.1) and (7.2), the lower bound test (step A4) is not effective since $z(P_i) = \infty$ holds for all $P_i \in \mathcal{P} - \mathcal{G}$.

This leads to the next theorem.

THEOREM 7.1 *Let* $A = ((\mathcal{B}, O, f), (\mathcal{G}, g), D, s_h)$ *and* $A' = ((\mathcal{B}, O, f), (\mathcal{G}, g), D', s_h)$ *be branch-and-bound algorithms where* s_h *is a breadth-first search function. Let* $\mathcal{T}$, $\mathcal{G}$ *satisfy* (7.1), (7.2), *and let* D *satisfy condition (vi). If* $D' \supset D$, *then*

$$T_a(A') \leq T_a(A),\ B_a(A') \leq B_a(A),\ T_s(A') \leq T_s(A),\ \text{and}\ B_s(A') \leq B_s(A). \quad (7.3)$$

PROOF. First note that $T_a(A) = B_a(A) = T_s(A) = B_s(A)$ and $T_a(A') = B_a(A') = T_s(A') = B_s(A')$ hold by assumption (7.1) and (7.2). Assume that a partial problem $P_i \in \mathcal{P} - \mathcal{T}$ is not decomposed in A. Since neither step A3 (by $P_i \notin \mathcal{G}$) nor step A4 (by the comment given above) is effective for P_i, P_i must have been terminated in step A5 (dominance test), i.e. P_kDP_i holds for some $P_k \in \mathcal{N}(P_i)$. Then by an argument similar to the end of part (b) in the proof of Theorem 5.2 (or the proof of Theorem 6.3), it can be proved that $P_tD'P_i$ holds for some $P_t \in \mathcal{N}'(P_i)$ (condition (vi) is required to show this); thus P_i is also terminated in step A5 of A'. Consequently only a subset of the nodes decomposed in A is also decomposed in A'. This proves (7.3), since breadth-first search applied to a problem satisfying (7.1) and (7.2) first selects the nodes in $\mathcal{P} - \mathcal{G}$ and then the nodes in $\mathcal{G}$ (note that no node in $\mathcal{G}$ is decomposed in A or A'). Q.E.D.

Although a branch-and-bound algorithm with breadth-first search is not efficient without dominance test (since the lower bound test is almost useless even if (7.1), (7.2) are not assumed; see also [14]), it should be emphasized that the algorithm can be very efficient if a very strong dominance relation is available. The algorithm for scheduling problems proposed by Sahni [32] is such an example.

8. *Depth-First Search*

As stated after Theorem 4.1, the monotonicity property of dominance relations does not hold for branch-and-bound algorithms with depth-first search. The monotonicity can be recovered, however, if step A5 of the branch-and-bound algorithm described in Section 3 and the definition of a dominance relation given in Section 2 are slightly modified.

Modified A5 (Dominance test)· If there exists $P_k (\neq P_i) \in \mathcal{N} - \mathcal{A}$ such that P_kDP_i, go to A8, otherwise go to A6.

Namely, $P_k \in \mathcal{N}$ in the original $A5_a$ and $A5_s$ is replaced by $P_k \in \mathcal{N} - \mathcal{A}$ (= the set of nodes which have been generated and tested).

We further assume that D satisfies the following condition in addition to conditions (i)–(iv) of Section 2.

(vii) P_iDP_j and $P_i \neq P_j$ imply that P_j is not a descendant of P_i.

Before proceeding to the main result, two lemmas concerning properties of depth-first search are given.

LEMMA 8.1. *Let* $A = ((\mathcal{B}, O, f), (\mathcal{G}, g), D, \bar{s}_h)$ *be a branch-and-bound algorithm with a depth-first search function based on* $h : \mathcal{P} \to E$ *(either A5 or the modified A5 is used), and let* $\mathcal{S} = P_{i_1}P_{i_2} \ldots P_{i_s}$ *be the sequence of nodes arranged in the order of selection in step A2 of A. Assume that* P_{i_q} *is the first node which satisfies* $p < q$ *and is not a descendant of* P_{i_p}. *Then all proper descendants of* P_{i_p} *eventually generated in A are located in* $\mathcal{S}$ *between* P_{i_p} *and* P_{i_q}.

PROOF. Denote by $\mathcal{A}(P_{i_p})$ the set of active nodes when P_{i_p} is selected. Then $P_{i_q} \in \mathcal{A}(P_{i_p})$ since P_{i_q} is not a descendant of P_{i_p}. In other words, P_{i_p} had the higher priority

than P_{i_q} in the selection by depth-first search. This means $d(P_{i_q}) \leq d(P_{i_p})$. Since any proper descendant P_j of P_{i_p} satisfies $d(P_j) > d(P_{i_p})$ $(\geq d(P_{i_q}))$, P_j is selected prior to P_{i_q} according to depth-first search. Q.E.D.

LEMMA 8.2. *Let A, P_{i_p}, P_{i_q} be defined as in Lemma 8.1, except that A uses the modified A5 and D satisfies condition (vii) given above. Denote the incumbent value when P_i is selected by $z(P_i)$. Then*

$$z(P_{i_q}) = min[z(P_{i_p}), f(P_{i_p})] . \tag{8.1}$$

PROOF. To prove by induction, assume that the lemma is true for any subsequence of $\mathcal{S}$, $P_{i_1}P_{i_2} \ldots P_{i_t}$, with $t < w$. For $w = 2$, this is trivially true. Furthermore, if P_{i_w} is a descendant of any P_{i_t}, $t < w$, the lemma is immediately extended to the sequence $P_{i_1}P_{i_2} \ldots P_{i_w}$. So assume that P_{i_p} $(1 \leq p \leq t)$ and $P_{i_w} = P_{i_q}$ satisfy the lemma statement. Note that

$$z(P_{i_q}) \leq z(P_{i_p}), \quad z(P_{i_q}) \geq \min[z(P_{i_p}), f(P_{i_p})] \tag{8.2}$$

follow from Lemma 8.1. Let $P_{j_1}, P_{j_2}, \ldots, P_{j_r}$ be the descendants of P_{i_p} such that $P_{j_k} \in \mathcal{G}$ and $f(P_{j_k}) = f(P_{i_p})$, $k = 1, 2, \ldots, r$. By Lemma 8.1, either P_{j_k} is selected prior to P_{i_q} or a proper ancestor of P_{j_k} is terminated for some reason. If one of $P_{j_1}, \ldots, P_{j_r}$ is actually selected, then $z(P_{i_q}) \leq f(P_{i_p})$ (see steps A3 and A7 of A) and hence $z(P_{i_q}) = \min[z(P_{i_p}), f(P_{i_p})]$ by (8.2).

Assume then that each P_{j_k} $(k = 1, 2, \ldots, r)$ has a proper ancestor which is terminated. If a proper ancestor P_a of P_{j_k} (note that P_a is a descendant of P_{i_p}) is terminated by A4 (lower bound test), then $z(P_a) \leq g(P_a) \leq f(P_a) = f(P_{i_p})$. Thus $z(P_{i_q}) \leq z(P_a) \leq f(P_{i_p})$ and (8.1) is proved from (8.2).

Finally assume that P_a is terminated in the modified A5 (dominance test). This implies that there exist P_b such that P_bDP_a and P_b is selected prior to P_a, and hence prior to P_{i_p}. (When all optimal solutions are sought, P_b is not a descendant of P_{i_p} since $f(P_b) < f(P_a) = f(P_{i_p})$ by condition $(i)_a$ of D. Thus P_b is selected prior to P_{i_p} by Lemma 8.1. When a single optimal solution is sought, P_b may be a descendant of P_{i_p} satisfying $f(P_b) = f(P_a) = f(P_{i_p})$. (Note that P_b is not an ancestor of P_a by condition (vii).) In this case P_b is already decomposed since $P_b \in \mathcal{N} - \mathcal{A}$. Thus there exists a proper descendant P_c of P_b such that P_c is a proper ancestor of some P_{j_u} $(1 \leq u \leq r)$ and terminated in the modified A5. Regarding P_c as P_a, apply the same argument as above. Repeating this, we will eventually have P_a and P_b as described above.) Relative positions of P_a, P_b, P_{i_p}, P_{i_q}, P_{j_k} are illustrated in Figure 4. Note that P_b is not an ancestor of P_{i_p} by condition (vii) and P_bDP_a. Thus $z(P_{i_p}) \leq \min[z(P_b), f(P_b)]$ (by induction hypothesis) $\leq f(P_b) \leq f(P_a) = f(P_{i_p})$. Consequently (8.1) follows from (8.2). Q.E.D.

THEOREM 8.3. *Let $A = ((\mathcal{B}, O, f), (\mathcal{G}, g), D, \bar{s}_h)$ and $A' = ((\mathcal{B}, O, f), (\mathcal{G}, g), D', \bar{s}_h)$ be branch-and-bound algorithms with a depth-first search function based on $h : \mathcal{P} \to E$, where A and A' use the modified step A5, and D, D' satisfy condition (vii). Then if $D' \supset D$ and D' is consistent with g, it follows that*

$$T_a(A') \leq T_a(A),\ B_a(A') \leq B_a(A),\quad T_s(A') \leq T_s(A),\ B_s(A') \leq B_s(A).$$

PROOF. Let $\mathcal{S} = P_{i_1}P_{i_2} \ldots P_{i_3}$, $\mathcal{S}' = P_{j_1}P_{j_2} \ldots P_{j_t}$ be the sequences of nodes arranged in the order of selection in A and A', respectively. It will be shown by induction that $\mathcal{S}'$ is a subsequence of $\mathcal{S}$. First note that $P_{i_1} = P_{j_1}$ $(= P_0)$, $\mathcal{A}(P_{i_1}) = \mathcal{A}'(P_{j_1})$ $(= \{P_0\})$ and $z(P_{i_1}) = z'(P_{j_1})$ $(= \infty)$ hold, where $\mathcal{A}(P_i)$ and $z(P_i)$ is the set of active nodes and the incumbent value when P_i is selected in A; $\mathcal{A}'(P_i)$ and $z'(P_i)$ are similarly defined for A'.

Now assume that $\mathcal{S}'_b = P_{j_1}P_{j_2} \ldots P_{j_b}$ $(b < t$ since otherwise the proof is done) is a subsequence of $\mathcal{S}_a = P_{i_1}P_{i_2} \ldots P_{i_a}$, and that $P_{j_b} = P_{i_a}$, $\mathcal{A}(P_{i_a}) = \mathcal{A}'(P_{j_b})$ and $z(P_{i_a}) = z'(P_{i_b})$. Obviously the induction can proceed one step if either P_{i_a} and P_{j_b} are both terminated or both are decomposed. Thus assume that exactly one of P_{i_a} and P_{j_b} is terminated. By assumption that $P_{i_a} = P_{j_b}$ and $z(P_{i_a}) = z(P_{j_b})$, such a case occurs only if

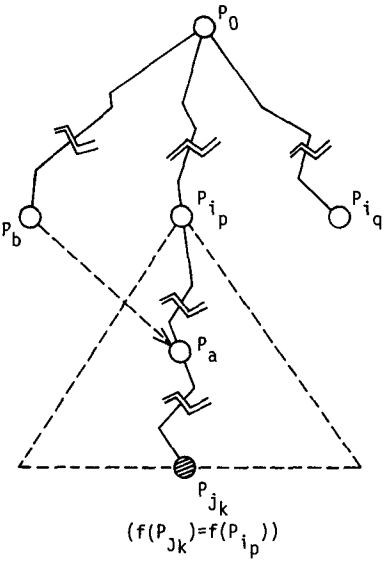


FIG. 4. Relative position of nodes used in the proof of Lemma 8.2

one of P_{i_a} and P_{j_b} is terminated in the modified step A5. First assume that P_{i_a} is terminated in A by $P_{i_p}DP_{i_a}$ for some P_{i_p} $(p < a)$. If $P_{i_p} = P_{j_q}$ holds for some $q < b$, then $P_{j_q}D'P_{j_b}$ and hence P_{j_b} is also terminated, a contradiction. On the other hand, if such P_{j_q} does not exist, a proper ancestor P_{j_r} of P_{j_q} $(r < b)$ must have been terminated in the modified step A5. Thus there exist P_{j_u} $(u < r < b)$ in $\mathcal{S}'$ such that $P_{j_u}D'P_{j_r}$. This implies by condition (iii) that a descendant P_{j_v} of P_{j_u} satisfies $P_{j_v}D'P_{j_q}$ (and hence $P_{j_v}D'P_{j_b}$). If P_{j_v} is generated in A', it is selected prior to P_{j_b} since P_{j_r} (satisfying $r < b$) is not a descendant of P_{j_u} by condition (vii) and hence selected after P_{j_v} by Lemma 8.1. Thus P_{j_b} is terminated in A' by $P_{j_v}D'P_{j_b}$. If on the other hand a proper ancestor of P_{j_v} is terminated in the modified step A5 (since, if it is terminated in A4, the consistency of D' with g implies that P_{j_b} is also terminated in A4, a contradiction) apply the same argument; after a finite number of iterations it is shown that some P_{j_w} $(w < b)$ satisfies $P_{j_w}D'P_{j_b}$. Consequently, P_{j_b} is terminated in A' if P_{i_a} is terminated in A.

Finally consider the case in which P_{i_a} is not terminated but P_{j_b} is terminated in the modified step A5. Then proper descendants of P_{i_a} are located in $\mathcal{S}$ between P_{i_a} and the first node, denoted P_{i_k}, which is not a descendant of P_{i_a}, by Lemma 8.1. Then $P_{i_k} = P_{j_{b+1}}$ since $\mathcal{A}(P_{i_a}) = \mathcal{A}'(P_{j_b})$ by assumption (note A and A' have the same search function $\bar{s}_h$), and

$$\mathcal{A}(P_{i_k}) = \mathcal{A}(P_{i_a}) - \{P_{i_a}\} = \mathcal{A}'(P_{j_b}) - \{P_{j_b}\} = \mathcal{A}'(P_{j_{b+1}}).$$

$z(P_{i_k}) = z'(P_{j_{b+1}})$ is proved as follows. First note that P_{j_q} satisfying $q < b$ and $P_{j_q}D'P_{j_b}$ is not an ancestor of P_{j_b} (condition (vii)) and $f(P_{j_q}) \leq f(P_{j_b})(= f(P_{i_a}))$ (condition $(\text{i})_a$ or $(\text{i})_s$ of a dominance relation). Thus $z'(P_{j_b}) \leq f(P_{j_q})$ (by Lemma 8.2) $\leq f(P_{j_b})$ and hence

$$z'(P_{j_{b+1}}) = \min[z'(P_{j_b}), f(P_{j_b})] = z'(P_{j_b})$$

by Lemma 8.2. Since P_{j_q} is also located before P_{i_a} in $\mathcal{S}_a$ (since $\mathcal{S}'_b$ is a subsequence of $\mathcal{S}_a$), it holds similarly that

$$z(P_{i_k}) = \min[z(P_{i_a}), f(P_{i_a})] = z(P_{i_a}) = z'(P_{j_b}).$$

This proves $z(P_{i_k}) = z'(P_{j_{b+1}})$.

As a result, we now have two sequences

$$\mathcal{S}_k = P_{i_1}P_{i_2} \ldots P_{i_a} \ldots P_{i_k} \text{ and } \mathcal{S}'_{b+1} = P_{j_1}P_{j_2} \ldots P_{j_b}P_{j_{b+1}}$$

such that $\mathcal{S}'_{b+1}$ is a subsequence of $\mathcal{S}_k$, $P_{i_k} = P_{j_{b+1}}$, $\mathcal{A}(P_{i_k}) = \mathcal{A}'(P_{j_{b+1}})$, and $z(P_{i_k}) = z'(P_{j_{b+1}})$. This shows by induction that $\mathcal{S}'$ is a subsequence of $\mathcal{S}$, and $T_a(A') \leq T_a(A)$, $T_s(A') \leq T_s(A)$, and $B_a(A') \leq B_a(A)$. (Note that this proof is applicable to both cases of all optimal solutions and a simple optimal solution.)

To show $B_s(A') \leq B_s(A)$, let P_{i_r} be the first node in $\mathcal{S}$ such that $P_{i_r} \in \mathcal{G}$ and $f(P_{i_r}) = f(P_0)$. Then $P_{i_r} = P_{j_s}$ holds for some P_{j_s} in $\mathcal{S}'$ (note that this proves $B_s(A') \leq B_s(A)$) since no proper ancestor P_{j_v} of P_{j_s} is terminated in A' as proved next: (a) $z'(P_{j_v}) = z(P_{j_v}) \geq z(P_{j_s}) = z(P_{i_r}) > f(P_0) = f(P_{i_r}) \geq g(P_{i_r}) = g(P_{j_s}) \geq g(P_{j_v})$ holds by assumption on P_{i_r} $(= P_{j_s})$, and hence steps A3, A4 are not active for P_{j_v} in A'; (b) there exists no P_j satisfying $f(P_j) < f(P_{j_r})$ $(= f(P_0))$ and hence $P_j D' P_{j_v}$ is possible only if $f(P_j) = f(P_{j_v})$ and P_j is not a proper ancestor of P_{j_v} by conditions $(i)_s$ and (vii) of a dominance relation. Then a descendent P_{j_u} of P_j satisfying $P_{j_u} \in \mathcal{G}, f(P_{j_u}) = f(P_0)$ must have been selected before P_{j_v} by Lemma 8.2, a contradiction to the fact that $\mathcal{S}'$ is a subsequence of $\mathcal{S}$ and P_{i_r} $(= P_{j_s})$ is the first node in $\mathcal{S}$ such that $P_{i_r} \in \mathcal{G}$ and $f(P_{i_r}) = f(P_0)$. This proves that step A5 is not active for P_{j_v} in A'. Q E.D.

9. *Conclusion*

In this paper we have found the following four subclasses of branch-and-bound algorithms in which a stronger dominance relation always results in a more efficient algorithm in terms of measures T and B: (1) those using heuristic search with nonmisleading heuristic functions; (2) those using best-bound search; (3) those using breadth-first search, where branching structures satisfy restrictions (7.1) and (7.2); and (4) those using depth-first search. Note that some restrictions on D are added in all cases, and step A5 of the algorithm is slightly modified in (4).

These results would indicate that a stronger dominance relation usually provides a more efficient algorithm for most of the branch-and-bound algorithms practically encountered, though it is not always true as shown in Theorem 4.1.

ACKNOWLEDGMENTS. The author wishes to thank Professors H. Mine and T. Hasegawa of Kyoto University for their support. He is also indebted to Professor W.H. Kohler of the University of Massachusetts and to two anonymous reviewers for their helpful comments. Some of the references involving the use of practical dominance relations were pointed out to the author by one of the reviewers.

REFERENCES

1 AGIN, N Optimum seeking with branch and bound *Manage. Sci. 13* (1966), B176–B185
2 AHRENS, J H , AND FINKE, G Merging and sorting applied to the zero-one knapsack problem *Oper Res 23* (1975), 1099–1109.
3 BALAS, E A note on the branch-and-bound principle *Oper Res. 16* (1968), 442–445
4 BELLMAN, R E *Dynamic Programming* Princeton U Press, Princeton, N J , 1957.
5 CONWAY, R W , MAXWELL, W L., AND MILLER, L W *Theory of Scheduling* Addison-Wesley, Reading, Mass , 1967
6 DIJKSTRA, E W A note on two problems in connexion with graphs *Numer. Math. 1* (1959), 269–271
7 DREYFUS, S E. An appraisal of some shortest path algorithms. *Oper Res. 17* (1969), 394–411
8 ELMAGHRABY, S E The one-machine sequencing problem with delay cost *J Indust Eng. 19* (1968), 105–108
9 FILLMORE, J P , AND WILLIAMSON, S G On backtracking· A combinatorial description of the algorithm *SIAM J Compt 3* (1974), 41–55
10 FOX, B L. Discrete optimization via marginal analysis *Manage Sci. 13* (1966), 210–216
11 FOX, B L , AND SCHRAGE, L E The value of various strategies in branch-and-bound. Res Rep , U of Chicago, Chicago, Ill., 1972
12 GEOFFRION, A.M , AND MARSTEN, R E Integer programming algorithms: A framework and state-of-the-art survey. *Manage. Sci. 18* (1972), 465–491.
13 HELD, M , AND KARP, R. A dynamic programming approach to sequencing problems *J. SIAM 10* (1962), 196–210
14 IBARAKI, T Theoretical comparisons of search strategies in branch-and-bound algorithms *Int J Comptr and Inform Sci 5* (1976), 315–344

15. IBARAKI, T On the computational efficiency of branch-and-bound algorithms. Working Paper, Dep. Applied Math and Physics, Kyoto U , Kyoto, Japan, 1975 (To appear in *J. Oper. Res Soc. Japan*)
16 IGNALL, E., AND SCHRAGE, L E Application of the branch and bound technique to some flow-shop sequencing problems *Oper Res. 13* (1965), 400–412.
17. INGARGIOLA, G , AND KORSH, J F An algorithm for the solution of 0-1 loading problems. *Oper. Res 23* (1975), 1110–1119
18 KOHLER, W.H , AND STEIGLITZ, K. Characterization and theoretical comparison of branch-and-bound algorithms for permutation problems. *J. ACM 21*, 1 (Jan. 1974), 140–156.
19 KOHLER, W.H Exact and approximate algorithms for permutation problems Ph D. Diss., Princeton U , Princeton, N J , 1972
20 KOHLER, W H., AND STEIGLITZ, K Exact, approximate, and guaranteed accuracy algorithms for the flow-shop problem $n/2/F/\bar{F}$ *J ACM 22*, 1 (Jan. 1975), 106–114
21. KOWALSKI, R. Search strategies for theorem proving. *Machine Intelligence 5*, B. Meltzer and D. Michie, Eds., American Elsevier, New York, 1970, pp 181–201
22. LAWLER, E.L., AND WOOD, D.E. Branch-and-bound methods· A survey. *Oper. Res 14* (1966), 699–719.
23. MINE, H , IBARAKI, T , AND KISE, H. Algorithms for a scheduling problem (in Japanese) *J Oper. Res. Soc. Japan 18* (1974), 23–37.
24 MITTEN, L.G Branch-and-bound methods· General formulation and properties *Oper. Res. 18* (1970), 24–34
25. MITTEN, L G , AND TSOU, C.A Efficient solution procedures for certain scheduling and sequencing problems. Proc. Symp. Theory of Scheduling and Its Applications, S E. Elmaghraby, Ed., Lecture Notes in Econ. and Math. Systems, Vol. 86, Springer-Verlag, Berlin, 1973.
26. MORIN, T.L., AND MARSTEN, R E An algorithm for nonlinear knapsack problems. Tech. Rep No 95, Oper. Res. Ctr , M I T., Cambridge, Mass , 1974
27 MORIN, T L , AND MARSTEN, R E. Branch-and-bound strategies for dynamic programming. Working Paper No 750–74, Sloan School of Management, M I.T , Cambridge, Mass., 1975.
28. NEMHAUSER, G.L., AND ULLMAN, Z. Discrete dynamic programming and capital allocation. *Manage. Sci. 15* (1969), 494–505
29 NILSSON, N J *Problem-Solving Methods in Artificial Intelligence*. McGraw-Hill, New York, 1971.
30 POHL, I First results on the effect of error in heuristic search. *Machine Intelligence 5*, B Meltzer and D Michie, Eds., American Elsevier, New York, 1970, p 219–236
31. PROSCHAN, F., AND BRAY, T A Optimal redundancy under multiple constraints *Oper Res. 13* (1965), 800–814.
32 SAHNI, S.K Algorithms for scheduling independent tasks. *J ACM 23*, 1 (Jan 1976), 116–127.
33. WEINGARTNER, H.M., AND NESS, D N. Methods of solution of the multidimensional 0/1 knapsack problem. *Oper. Res 15* (1967), 83–103.

RECEIVED JUNE 1975; REVISED MARCH 1976