Journal of the Operations Research
Society of Japan
Vol. 25, No. 3, September 1982

THE POWER OF UPPER AND LOWER BOUNDING FUNCTIONS IN BRANCH-AND-BOUND ALGORITHMS

Toshihide Ibaraki
Kyoto University

(Received October 12, 1980; Revised January 27, 1982)

Abstract In a branch-and-bound algorithm, a partial problem P_i is terminated if the lower bound of the optimal value of P_i is greater (in case all optimal solutions are sought) or not smaller (in case a single optimal solution is sought) than the least upper bound on the optimal value of the original minimization problem P_0 currently available. Although it seems obvious that tighter lower bounding function and upper bounding function always improve the efficiency of a branch-and-bound algorithm, counterexamples can be easily constructed. In this paper, therefore, it is extensively studied when such improvement is guaranteed, for typical search strategies such as heuristic search, best-bound search and depth-first search. The model of branch-and-bound algorithms used for investigation is quite general in the sense that it allows the dominance test as well as the lower bound test mentioned above. The efficiency is measured by the number of partial problems decomposed in the execution of the algorithm.

1. Introduction

A branch-and-bound algorithm to solve a minimization problem P_0 is generally defined by (i) a *branching structure* $\mathscr{B}$ describing how P_0 is decomposed into partial problems of smaller and smaller sizes, (ii) *lower bound test* based on an *upper bounding function* u and a *lower bounding function* g (defined on the set of partial problems $\mathscr{P}$) that terminates those partial problems whose lower bounds are greater (in case all optimal solutions of P_0 are sought) or not smaller (in case a single optimal solution of P_0 is sought) than the least upper bound of the optimal value of P_0 known by then, (iii) *dominance test* based on a dominance relation D (a binary relation defined on $\mathscr{P}$) that terminates a partial problem P_i if another partial problem P_j generated by then is known to have a better solution, and (iv) a *search function* s specifying the

order of testing the generated partial problems. Four known types of search functions are treated in this paper; heuristic search function, best-bound search function, depth-first search function and breadth-first search function. Our view is therefore that, for a given branching structure $\mathscr{B}$, a branch-and-bound algorithm is essentially a lower bounding function g, an upper bounding function u, a dominance relation D and a search function s.

Although branch-and-bound is a wellknown principle for solving combinatorial optimization problems, only limited amount of research has been directed to clarify its general properties. Motivated by the pioneering work such as [8, 13, 18, 19], the author has investigated in earlier papers how the computational efficiency of a branch-and-bound algorithm (measured by the number of decomposed partial problems) depends on the accuracy of a search function s [9] and the strength of a dominance relation D [11]. Contrary to our intuitive understanding, it turned out that improvement in efficiency is theoretically guaranteed when a search function is improved or dominance test is strengthened only for certain restricted classes of branch-and-bound algorithms. Similar properties were also examined in [10] for approximate branch-and-bound algorithms which incorporate allowance functions specifying allowable deviation from the exact optimal value.

In this paper, we discuss how the efficiency depends on the tightness of an upper bounding function u and a lower bounding function g. The first result shown in Sections 3-4 is that tightening u and/or g does not always result in an improvement of efficiency. We see that this pathological phenomenon comes from the conflict between lower bound test and dominance test, which becomes possible under certain search functions. It is then examined what is necessary to guarantee an improvement in computational efficiency when u and/or g are tightened, for each case of the above four search functions. It turns out that the consistency assumption of D with respect to g plays a crucial role. Under this assumption we show that tightening g always results in improvement for most search functions, in Sections 3 and 4, and that tightening u always results in improvement for all search functions mentioned above, in Section 5. Without assuming the consistency, an improvement in efficiency is not guaranteed except for a few special cases. These special cases are also discussed in Sections 3-5. As will be noted later, some special cases of our results have been known in the literature such as [5, 8, 13, 15, 18].

2. Branch-and-Bound Algorithm

A formal deseription of a branch-and-bound algorithm A applied to a mini-

mization problem P_0 is given in this section, after introducing eight constituents of it. The justification may be found elsewhere [1, 2, 6, 8, 9, 13, 16, 17, 18, 20] and is not given here. Two types of branch-and-bound algorithms are considered throughout this paper: One is to obtain all optimal solutions of P_0 and the other is to obtain a single optimal solution of P_0. In most cases, however, proofs are given only to the case of all optimal solutions. The case of a single optimal solution can usually be treated similarly. Complete proofs may be found in [12].

A *finite rooted tree* $\mathscr{B} = (\mathscr{P}, \mathscr{E})$ with a set of nodes $\mathscr{P}$ and a set of arcs $\mathscr{E}$ represents how P_0 (represented by the root P_0 of $\mathscr{B}$) is decomposed into partial problems when all possible decompositions are executed; $(P_i, P_j) \in \mathscr{E}$ denotes that partial problem P_j is generated from P_i by a decomposition. $\mathscr{T}$ denotes the set of leaf nodes in $\mathscr{B}$. Terminologies such as *son*, *ancestor*, *descendant*, *depth* of P_i (denoted $d(P_i)$) are defined in a customary manner (e.g., [9]).

Let $f: \mathscr{P} \to E \cup \{\infty\}$ denote the *optimal values of nodes* (partial problems), where E is the set of real numbers. $f(P_i) = \infty$ if P_i is *infeasible*. f satisfies

$$f(P_i) = \min \{f(P_j) \mid (P_i, P_j) \in \mathscr{E}\} \tag{2.1}$$

and hence

$$f(P_i) \leq f(P_j) \text{ for } (P_i, P_j) \in \mathscr{E}. \tag{2.2}$$

$O(P_i)$ denotes the *set of optimal solutions* of $P_i \in \mathscr{P}$. It satisfies

$$O(P_i) = \{O(P_j) \mid f(P_j) = f(P_i), (P_i, P_j) \in \mathscr{E}\}.$$

$(\mathscr{B}, O, f)$ (O is sometimes omitted) is called the *branching structure* of P_0.

In executing a branch-and-bound algorithm, $f(P_i)$ is usually not known but a *lower bounding function* $g(P_i)$ is evaluated for each generated P_i. $g: \mathscr{P} \to E \cup \{\infty\}$ satisfies

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

$\mathscr{G}$ denotes the set of nodes P_i for which $g(P_i) = f(P_i)$ is known (and $O(P_i)$ is obtained) or $O(P_i) \cap O(P_0) = \phi$ is concluded, in the computation process of g. It satisfies

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

Note that condition (A) is assumed for simplicity even if $P_i \in \mathscr{G}$ is concluded

due to $O(P_i) \cap O(P_0) = \phi$, since in this case the value $g(P_i)$ is not relevant to the computation process.

At this point, let us derive $\mathscr{B}$ and g of a typical branch-and-bound algorithm for the (mixed) integer programming problem, as an example. Assume that the algorithm uses the decomposition scheme proposed by Dakin [3] and the LP (linear programming) lower bound (e.g., [6] for general description). Then a partial problem P_i (including the case of $P_i = P_0$) is decomposed into two partial problems by adding constraints

$$x_k \leq K \text{ and } x_k \geq K + 1 \tag{2.3}$$

respectively to the original constraint of P_i, where x_k is an integer variable selected for decomposition (called a *branching variable*) and K is a nonnegative integer such that x_k assumes a value between K and $K + 1$ in the optimal solution of the LP problem corresponding to P_i (i.e., obtained from P_i by removing the integrality condition on variables). Thus all partial problems are again integer programming problems. We see now that the resulting branching structure is a finite binary tree (finite under the assumption that all integer variables are bounded both from below and above). $f(P_i)$ is the optimal value of integer programming problem P_i, $O(P_i)$ is the set of optimal solutions of P_i, and $g(P_i)$ is the optimal value of the LP problem corresponding to P_i. We say that $P_i \in \mathscr{T}$ if all integer variables are fixed by additional constraints of type (2. 3), and $P_i \in \mathscr{G}$ if the LP optimal solution happens to be an integer solution or the LP problem turns out to be infeasible. Obviously these $\mathscr{B}$, f, O, $\mathscr{G}$, g, $\mathscr{T}$ satisfy the above conditions.

In many practical cases, a good feasible solution of each partial problem P_i is obtained by simple computation. This gives rise to an *upper bounding function* $u: \mathscr{P} \to E \cup \{\infty\}$ satisfying

(I) $u(P_i) \geq f(P_i)$ for $P_i \in \mathscr{P}$

(II) $u(P_i) = f(P_i)$ for $P_i \in \mathscr{G}$.

$u(P_i) = \infty$ denotes that no feasible solution is obtained for P_i or that the computation of $u(P_i)$ is not attempted. $u = \infty$ stands for that $u(P_i)$ is never computed, and $u = u(P_0)$ stands for that u is computed only for the original problem P_0. Note however that condition (II) is assumed even in these cases.

A *dominance relation* D is also used to test partial problems. D is a *partial ordering* on $\mathscr{P}$ satisfying the following conditions.

When all optimal solutions of P_0 are sought:

(i)$_a$ $P_i D P_j \wedge P_i \neq P_j$ implies $f(P_i) < f(P_j)$ (including $\infty < \infty$).

(ii)$_a$ $P_i D P_j \wedge P_i \neq P_j$ implies that, for each descendant $P_{\bar{j}}$ of P_j, there

exists a descendant $P_{\bar{i}}$ of P_i satisfying $P_{\bar{i}} DP_{\bar{j}}$.

When a single optimal solution of P_0 is sought:

(i)$_s$ $P_i DP_j \wedge P_i \neq P_j$ implies $f(P_i) \leq f(P_j)$ and that P_j is not an ancestor of P_i.

(ii)$_s$ $P_i DP_j \wedge P_i \neq P_j$ implies, for each descendant $P_{\bar{j}}$ of P_j, there exists a descendant $P_{\bar{i}}$ of P_i satisfying $P_{\bar{i}} DP_{\bar{j}}$,.

(iii)$_s$ There exists no sequence of nodes $P_{i_1}, P_{i_2}, \ldots, P_{i_{k+1}}$ ($k \geq 2$ and $P_{i_1}, P_{i_2}, \ldots, P_{i_k}$ are distinct) generated during computation, such that P_{i_s} is a proper descendant of $P_{i_{s+1}}$ or $P_{i_s} DP_{i_{s+1}} \wedge f(P_{i_s}) = f(P_{i_{s+1}})$ for $s=1,2,\ldots,k$, and $P_{i_{k+1}} = P_{i_1}$. (This condition is not used in this paper but is necessary to guarantee that a single optimal solution is obtained.)

A dominance relation D is called to be *consistent with* g if it satisfies the following additonal condition:

$P_i DP_j \wedge P_i \neq P_j$ implies $g(P_i) < g(P_j)$ in case all optimal solutions are sought, and $g(P_i) \leq g(P_j)$ in case a single optimal solution is sought.

$D = I$ (identity relation) indicates that the test based on D is not effective.

Examples of dominance relations in various combinatorial optimization problems may be found in [11] together with relevant references. The consistency assumption is satisfied in most of these examples. However, it would be still nice to prove properties without the consistency assumption, if possible, since D and g are usually designed independently without regard to the consistency between them.

The order to test the generated partial problems is specified by a *search function* s: $\mathscr{I} \to \mathscr{P}$ such that $s(\mathscr{A}) \in \mathscr{A}$ for $\mathscr{A} \in \mathscr{I}$, where $\mathscr{I}$ denotes the family of independent subsets of $\mathscr{P}$. The following four search functions are typical.

s is the *heuristic search function* based on a *heuristic function* h: $\mathscr{P} \to E$, denoted $s = s_h$, if

$$h(s(\mathscr{A})) = \min \{h(P_i) \mid P_i \in \mathscr{A}\} \text{ for } \mathscr{A} \in \mathscr{I}.$$

It is usually assumed that $h(P_i) \neq h(P_j)$ for $P_i \neq P_j$ by using an appropriate tie breaking rule if necessary. In particular, $s = s_g$ is called the *best-bound search function*. The *depth-first search function* based on h, denoted $s = \bar{s}_h$, is defined by

$$h(\bar{s}_h(\mathscr{A})) = \min \{h(P_i) \mid P_i \in \bar{N}(\mathscr{A})\}$$

$$\bar{s}_h(\mathscr{A}) \in \bar{N}(\mathscr{A})$$

for $\mathscr{A} \in \mathscr{I}$, where

$$\bar{N}(\mathscr{A}) = \{P_i \in \mathscr{A} \mid d(P_i) = \max \{d(P_j) \mid P_j \in \mathscr{A}\}\}$$

Finally, the *breadth-first search function* based on h, denoted $s = \tilde{s}_h$, is defined by

$$h(s_h(\mathscr{A})) = \min\{h(P_i) \mid P_i \in \tilde{N}(\mathscr{A})\}$$

$$\tilde{s}_h(\mathscr{A}) \in \tilde{N}(\mathscr{A})$$

$$\tilde{N}(\mathscr{A}) = \{P_i \in \mathscr{A} \mid d(P_i) = \min\{d(P_j) \mid P_j \in \mathscr{A}\}\}$$

It is known [9] that a heuristic search function s_h is most general among the above search functions in the sense that the other three can be viewed as s_h with special h. Thus properties proved for heuristic search are valid for all the above search functions.

A heuristic function h is called *nonmisleading* if $h(P_i)<h(P_j)$ implies $f(P_i)\leq f(P_j)$ for P_i, $P_j \in \mathscr{P}$. A nonmisleading h is considered as a theoretical goal when we design a heuristic function [4, 9]. Even if h is not nonmisleading, however, it is shown in [9] that the behavior of a branch-and-bound algorithm becomes close to that with a nonmisleading one if h is almost nonmisleading. Thus the analysis of the case of a nonmisleading h may help understand the behavior of branch-and-bound algorithms that are very nicely designed.

Based on these constituents, a formal description of a branch-and-bound algorithm is now given both for the case of all optimal solutions and for the case of a single optimal solution.

Branch-and-bound algorithm $A_a=((\mathscr{B}, O, f), (\mathscr{G}, g, u), D, s)$: all optimal solutions

In the following, $\mathscr{N} \subset \mathscr{P}$ denotes the set of nodes currently generated. A node in $\mathscr{N}$ is *active* if it is yet neither tested nor decomposed. $\mathscr{A}$ denotes the set of current active nodes. $\mathscr{O}$ stores the set of *best feasible solutions* currently available. z is called the *incumbent value* and stores the current best upper bound of $f(P_0)$. Generally $z\leq f(\mathscr{O})$ ($=f(x)$ for $x\in\mathscr{O}$) holds ($z<f(\mathscr{O})$ is possible since z is set in Step A2 even if optimal solutions are not known), but $z=f(\mathscr{O})$ is satisfied if $u=\infty$ or if the computation has terminated. It is assumed that $O(P_i)$ is obtained as a by-product of testing P_i if $P_i\in\mathscr{G}$.

A1(Initialize): $\mathscr{A}\leftarrow\{P_0\}$, $\mathscr{N}\leftarrow\{P_0\}$, $z\leftarrow\infty$ and $\mathscr{O}\leftarrow\phi$.

A2(Search): If $\mathscr{A}=\phi$, go to A9; else $P_i\leftarrow s(\mathscr{A})$, $z\leftarrow\min[z, u(P_i)]$ and go to A3.

A3(Test by $\mathscr{G}$): If $P_i\in\mathscr{G}$, go to A7; else go to A4.

A4(Lower bound test): If $g(P_i)>z$, go to A8; else go to A5.

A5(Dominance test): If there exists $P_k(\neq P_i)\in\mathscr{N}$ satisfying $P_k DP_i$, go to A8; else go to A6.

A6(Decompose): Generate sons $P_{i_1}, P_{i_2}, \ldots, P_{i_k}$ of P_i. Return to A2 After letting $\mathscr{A}\leftarrow\mathscr{A}\cup\{P_{i_1}, P_{i_2}, \ldots, P_{i_k}\}-\{P_i\}$ and $\mathscr{N}\leftarrow\mathscr{N}\cup\{P_{i_1}, P_{i_2}, \ldots, P_{i_k}\}$.

A7(Improve): Go to A8 after letting

$$\mathcal{O} \leftarrow \begin{cases} O(P_i) \text{ if } f(P_i)<f(\mathcal{O}) \\ \mathcal{O} \cup O(P_i) \text{ if } f(P_i)=f(\mathcal{O}) \\ \mathcal{O} \text{ otherwise.} \end{cases}$$

A8(Terminate P_i): $\mathcal{A} \leftarrow \mathcal{A}-\{P_i\}$ and return to A2.

A9(Halt): Halt. $\mathcal{O}=O(P_0)$ and $z=f(P_0)$ hold. □

Branch-and-bound algorithm $A_s=((\mathcal{B}, f), (\mathcal{G}, g, u), D, s)$: a single optimal solution.

In this case, $\mathcal{O}$ stores at most one solution x, and $z=f(x)$ always holds. It is assumed that a feasible solution x of P_i satisfying $f(x)=u(P_i)$ is obtained in the computation of $u(P_i)$ if $u(P_i)<\infty$ (thus an optimal solution of P_i is obtained if $P_i \in \mathcal{G}$).

A1, A5, A6, A8 are the same as those in A_a. A7 may be eliminated since it is never executed.

A2(Search): If $\mathcal{A}=\phi$, go to A9; else $P_i \leftarrow s(\mathcal{A})$, $z \leftarrow \min[z, u(P_i)]$, $\mathcal{O} \leftarrow \mathcal{O}$ if $u(P_i) \geq z$ else $\{x\}$, where x is a feasible solution of P_i with $f(x)=u(P_i)$. Go to A3.

A3(Test by $\mathcal{G}$): If $P_i \in \mathcal{G}$, go to A8; else go to A4.

A4(Lower bound test): If $g(P_i) \geq z$, go to A8; else go to A5.

A9(Halt): Halt. x stored in $\mathcal{O}$ and z satisfy $f(x)=z=f(P_0)$. □

The finiteness and correctness of the above two algorithms (or their special cases) may be found in references such as [1, 2, 6, 8, 13, 14, 16, 17, 18, 20].

Throughout this paper, the following parameters are used to measure the computational efficiency of a branch-and-bound algorithm A.

$T(A)$: The number of nodes decomposed in A6 before the termination in A9 is reached.

$B(A)$: The number of nodes decomposed in A6 prior to the last modification of $\mathcal{O}$ (which has occurred in A7 if all optimal solutions are sought, or in A2 if a single optimal solution is sought).

$T(A)$ is relevant to the total computation time of A, and $B(A)$ is relevant to the time when optimal solutions of P_0 are stored in $\mathcal{O}$. $B(A)$ is an important measure for the quality of solutions stored in $\mathcal{O}$ when the computation may be cut off before the normal termination in A9, due to the insufficiency of the available computer time. It is of course desirable to make $T(A)$ and $B(A)$ small.

In the subsequent discussion, subscripts a and s are sometimes added, e.g., A_a, A_s, $T_a(A)$, $B_s(A)$ and so on, to distinguish the cases of all optimal solutions and a single optimal solution respectively. No subscript is added, however, if it is not necessary to distinguish them.

3. Power of Lower Bounding Functions under Heuristic Search

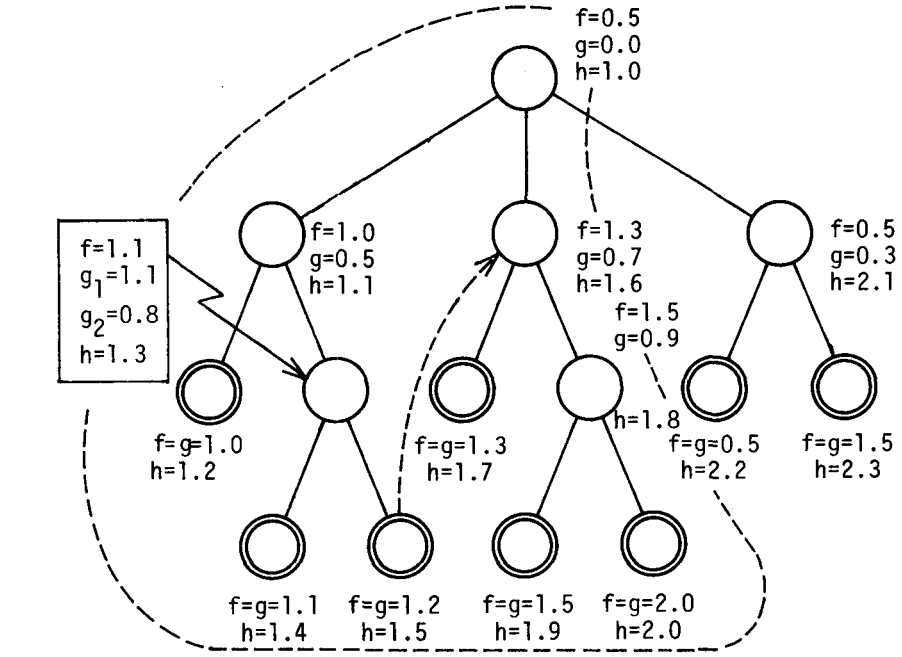
Consider two branch-and-bound algorithms $A(g_1) = ((\mathscr{B}, O, f), (\mathscr{G}_1, g_1, u), D, s_h)$ and $A(g_2) = ((\mathscr{B}, O, f), (\mathscr{G}_2, g_2, u), D, s_h)$. $A(g_1)$ and $A(g_2)$ differ only in $\mathscr{G}$ and g, and both algorithms use the same heuristic search function s_h which is not dependent on $(\mathscr{G}, g)$. We say that $(\mathscr{G}_1, g_1)$ is *tighter* than $(\mathscr{G}_2, g_2)$ and denote by $g_1 \geq g_2$, if $\mathscr{G}_1 \supset \mathscr{G}_2$ and $g_1(P_i) \geq g_2(P_i)$ for $P_i \in \mathscr{P}$. For example, let $g_2(P_i)$ denote the LP optimal value for an integer programming problem P_i, as mentioned in Section 2. It is known that g_2 can be improved to g_1 satisfying $g_1 \geq g_2$ by employing the concept of penalty (e.g., [21]) or by resorting to the group theoretic approach [7] in case P_i is an all-integer problem. It has been conjectured that $g_1 \geq g_2$ implies $T(g_1) \leq T(g_2)$ and $B(g_1) \leq B(g_2)$, where $T(g_k)$ and $B(g_k)$ are abbreviations of $T(A(g_k))$ and $B(A(g_k))$ respectively. But this is not generally true as we shall see below.

Theorem 3.1. Let $A(g_1) = ((\mathscr{B}, O, f), (\mathscr{G}_1, g_1, u), D, s_h)$ and $A(g_2) = ((\mathscr{B}, O, f), (\mathscr{G}_2, g_2, u), D, s_h)$ be branch-and-bound algorithms using heuristic search. Then $g_1 \geq g_2$ does not necessarily imply $T(g_1) \leq T(g_2)$ or $B(g_1) \leq B(g_2)$. This is true even if s_h is further restricted to be (i) a depth-first search function, or (ii) a breadth-first search function. Furthermore, (iii) $g_1 \geq g_2$ does not necessarily imply $T(g_1) \leq T(g_2)$ even if h of s_h is nonmisleading.

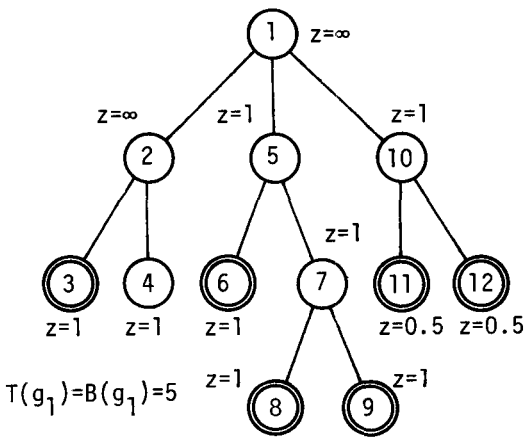
Proof. See the example given in Fig. 1. Fig. 1(a) gives $(\mathscr{B}, f)$, $(\mathscr{G}_1, g_1)$, $(\mathscr{G}_2, g_2)$, h and D. Nodes in $\mathscr{G}_1 = \mathscr{G}_2$ (in this case) are indicated by double circles, and dominance relation $P_i DP_j$ is denoted by $P_i \cdots\!\!\rightarrow P_j$. $u = \infty$ is assumed in this example (note however that $u(P_i) = f(P_i)$ are assumed for $P_i \in \mathscr{G}$ by definition). The computation processes of $A(g_1)$ and $A(g_2)$ are illustrated in Fig. 1(b) and Fig. 1(c) respectively. The node numbers denote the order in which nodes are tested. The z-value attached to each node is the incumbent value after the update in A2. It is easy to see that $T(g_1) = B(g_1) = 5 > T(g_2) = B(g_2) = 4$ in spite of $g_1 \geq g_2$.

(i) is true since s_h used in Fig. 1 is a depth-first search function. (ii) is proved by the example in Fig. 2; this has $T(g_1) = B(g_1) = 4 > T(g_2) = B(g_2) = 3$ in spite of $g_1 \geq g_2$. (iii) is proved by considering the subtree of Fig. 1(a) surrounded by broken curve; h of this portion is nonmisleading and gives $T(g_1) = 4 > T(g_2) = 3$. □

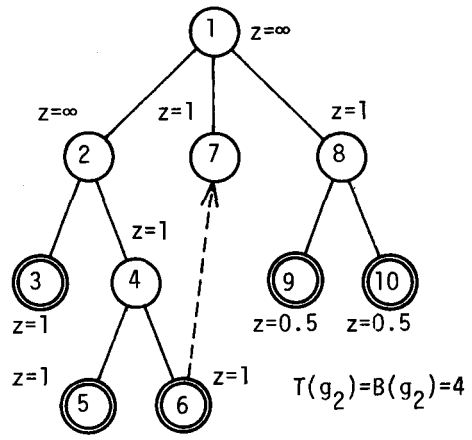
The next theorem treats the final case, i.e., the B-count under a nonmisleading s_h.



(a) ($\mathcal{B}$, f), g and h (u=∞ is assumed)



(b) Computation process of $A(g_1)$



(c) Computation process of $A(g_2)$

Fig. 1 Counterexample to the conjecture $g_1 \geq g_2 \Rightarrow T(g_1) \leq T(g_2) \wedge B(g_1) \leq B(g_2)$ under heuristic search used in the proof of Theorem 3.1. (Nodes in $\mathcal{G}$ are denoted by double circles. Dominance relation $P_i DP_j$ is indicated by $P_i \dashrightarrow P_j$. Some relations derivable from others by conditions (i)∼(iii) of D are not indicated. Node numbers in (b) and (c) denote the order of nodes tested in $A(g_1)$ and $A(g_2)$ respectively.)

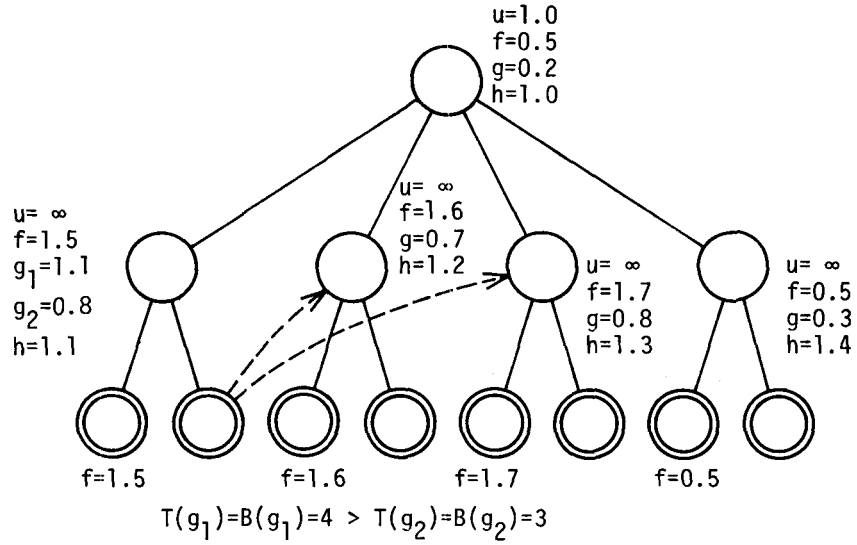


Fig. 2 Counterexample to the conjecture $g_1 \geq g_2 \Rightarrow T(g_1) \leq T(g_2) \wedge B(g_1) \leq B(g_2)$ under breadth-first search used in the proof of Theorem 3.1. (Only relevant values of u, f, g and h are indicated.)

Theorem 3.2. Let $A(g_1)$ and $A(g_2)$ be defined as in Theorem 3.1, and assume in addition that h is nonmisleading. Then $B_s(g_1) = B_s(g_2)$ holds, i.e., the B_s-count is independent of $(\mathscr{G}, g)$, and $g_1 \geq g_2$ implies $B_a(g_1) \leq B_a(g_2)$ (proper inequality is possible only if $\mathscr{G}_1 \supsetneqq \mathscr{G}_2$).

Proof. Let $\mathscr{S} = P_{j_1} P_{j_2} \ldots P_{j_t}$ be the sequence of nodes selected in A2 of $A(g_1)$ (and $A(g_2)$) with A3 (test by $\mathscr{G}$) replaced by: If $P_i \in \mathscr{T}$ go to A7; else go to A4, and with A4 (lower bound test) and A5 (dominance test) suppressed. (Thus $\mathscr{S}$ is the sequence of all nodes in $\mathscr{P}$ arranged in the order selected by search function s_h. It is independent of $(\mathscr{G}, g)$ and D.) The sequences of nodes actually selected in $A(g_1)$ and $A(g_2)$ are subsequences of $\mathscr{S}$ (Proposition 4.3 of [9]), and when h is nonmisleading,

$$f(P_{j_1}) \leq f(P_{j_2}) \leq \ldots \leq f(P_{j_t})$$

holds (Lemma 5.1 of [9]).

We give a proof only for the case of all optimal solutions. Let P_{j_a} be the last node which is selected in A2 of $A(g_1)$ and satisfies $P_{j_a} \in \mathscr{G}_1$, $f(P_{j_a}) = f(P_0)$. Then P_{j_b} $(1 \leq b \leq a)$ is terminated neither by lower bound test since $g(P_{j_b}) \leq f(P_{j_b}) \leq f(P_{j_a}) = f(P_0) \leq z_1$ (P_{j_b}), nor by dominance test since P_{j_b}

is not dominated by any other nodes because of $f(P_{j_b}) = f(P_0)$ and condition $(i)_a$ of D. Here $z_k(P)$ denotes the incumbent value right after P is selected and z is updated in A2 of $A(g_k)$. This property can be extended to $A(g_2)$. Although some of P_{j_b} $(1 \le b \le a)$ may be terminated by $\mathscr{G}_2$ (in $A(g_2)$), such P_{j_b} is also terminated by $\mathscr{G}_1$ (in $A(g_1)$) since $\mathscr{G}_1 \supset \mathscr{G}_2$. Consequently P_{j_b} $(1 \le b \le a)$ is terminated in $A(g_2)$ only if it is terminated in $A(g_1)$. This proves $B_a(g_1) \le B_a(g_2)$. The result concerning proper inequality is also obvious. □

As we have seen in the above theorems, T and B are not monotonically related to $(\mathscr{G}, g)$ in most cases. In order to guarantee the monotomic relation, the consistency assumption on D with respect to g (defined in Section 2) seems to be crucial.

Theorem 3.3. Let $A(g_1)$ and $A(g_2)$ be as defined in Theorem 3.1, where $A(g_1)$ and $A(g_2)$ use a heuristic search function s_h. In addition, assume that D is consistent with g. Then $g_1 \ge g_2$ implies $T(g_1) \le T(g_2)$ and $B(g_1) \le B(g_2)$.

Proof. We consider the case of all optimal solutions only. Let $\bar{\mathscr{P}}^1 = P_{i_1} P_{i_2} \cdots P_{i_s}$ and $\bar{\mathscr{P}}^2 = P_{j_1} P_{j_2} \cdots P_{j_t}$ be the sequences of nodes selected in $A(g_1)$ and $A(g_2)$ respectively. These are subsequences of $\mathscr{S}$ introduced in the proof of Theorem 3.2. We first show by induction that $\bar{\mathscr{P}}^1$ is a subsequence of $\bar{\mathscr{P}}^2$. (This immediately implies $T(g_1) \le T(g_2)$, and is a key step to prove $B(g_1) \le B(g_2)$.)

For that, consider a slightly stronger induction hypothesis that for any

$$\bar{\mathscr{P}}^1_a = P_{i_1} P_{i_2} \cdots P_{i_a}, \quad 1 \le a \le s,$$

there is the unique initial portion of $\bar{\mathscr{P}}^2$,

$$\bar{\mathscr{P}}^2_b = P_{j_1} P_{j_2} \cdots P_{j_b}$$

such that $P_{i_a} = P_{j_b}$, $\mathscr{A}_1(P_{i_a}) \subset \mathscr{A}_2(P_{j_b})$, $z_1(P_{i_a}) = z_2(P_{j_b})$ and $\bar{\mathscr{P}}^1_a$ is a subsequence of $\bar{\mathscr{P}}^2_b$, where $\mathscr{A}_k(P)$ is the set of active nodes when P is selected in A2 of $A(g_k)$ $(k = 1, 2)$, and $z_k(P)$ was defined in the proof of Theorem 3.2. This also implies $\mathscr{N}_1(P_{i_a}) \subset \mathscr{N}_2(P_{j_b})$ since $\mathscr{N}_1(P_{i_a}) = \{P_{i_1}, P_{i_2}, \ldots, P_{i_{a-1}}\} \cup \mathscr{A}_1(P_{i_a})$ and $\mathscr{N}_2(P_{j_b}) = \{P_{j_1}, P_{j_2}, \ldots, P_{j_{b-1}}\} \cup \mathscr{A}_2(P_{j_b})$.

This induction hypothesis is trivially true for $a = 1$, since $\bar{\mathscr{P}}^2_1$ satisfies the above conditions. In order to prove the general case, we first show that

P_{i_a} is terminated in $A(g_1)$ whenever P_{j_b} is terminated in $A(g_2)$. The following two cases are considered corresponding to how P_{j_b} is terminated.

(a) P_{j_b} is terminated by $\mathscr{G}_2$ or by lower bound test: If $P_{j_b} \in \mathscr{G}_2$, then $P_{i_a}(=P_{i_b}) \in \mathscr{G}_1$ by $\mathscr{G}_1 \supset \mathscr{G}_2$. If $g_2(P_{j_b}) > z_2(P_{j_b})$, then $g_1(P_{i_a}) \geq g_2(P_{j_b}) > z_2(P_{j_b}) = z_1(P_{i_a})$ (by induction hypothesis). In either case P_{i_a} is terminated in $A(g_1)$.

(b) P_{j_b} is terminated by dominance test: Then some $P_{j_p} \in \mathscr{N}_2(P_{j_b})$ satisfies $P_{j_p}\ DP_{j_b}$, where $\mathscr{N}_k(P)$ denotes $\mathscr{N}$ when P is selected in $A(g_k)$. Assume $P_{j_p} \notin \mathscr{N}_1(P_{i_a})$ since otherwise P_{i_a} is also terminated. This means that a proper ancestor P_{i_q} of P_{j_p} has been terminated in $A(g_1)$ (see Fig. 3).

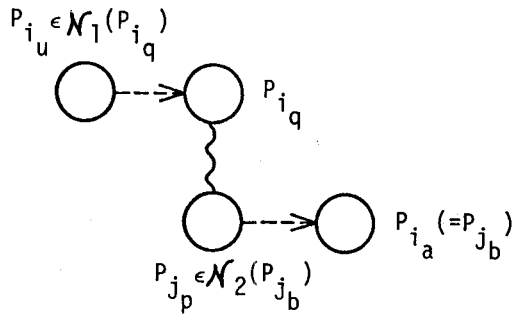


Fig. 3 Illustration of the set of nodes used in the proof of Theorem 3.3.

If P_{i_q} has been terminated by lower bound test, or by $\mathscr{G}_1$, it follows that

$$\begin{aligned} z_1(P_{i_a}) &\leq z_1(P_{i_q}) \text{ (since } P_{i_a} \text{ is selected after } P_{i_q} \text{ in } A(g_1)) \\ &\leq g_1(P_{i_q}) \leq g_1(P_{j_p}) \text{ (by condition (c) of } g) \\ &< g_1(P_{i_a}) \text{ (since } D \text{ is consistent with } g). \end{aligned}$$

Hence P_{i_a} is terminated in $A(g_1)$ by lower bound test. Thus assume that P_{i_q} has been terminated in $A(g_1)$ by dominance test, i.e., $P_{i_u}\ DP_{i_q}$ for some $P_{i_u} \in \mathscr{N}_1(P_{i_q})$.

Then $P_{i_u} \in \mathscr{N}_2(P_{i_q})$ follows since $\mathscr{N}_2(P_{i_q}) \supset \mathscr{N}_1(P_{i_q})$ by induction hypothesis. Thus P_{i_q} must have been terminated in $A(g_2)$ by dominance test, a contradiction. This proves that P_{i_a} is terminated in $A(g_1)$.

Now the induction can proceed one step unless $\bar{\mathscr{P}}^1_a = \bar{\mathscr{P}}^1$ (in this case the proof is done):

$$\bar{\mathscr{P}}^1_{a+1} = P_{i_1} P_{i_2} \cdots P_{i_a} P_{i_{a+1}}$$

$$\bar{\mathscr{P}}^2_{b+w} = P_{j_1} P_{j_2} \cdots P_{j_b} \cdots P_{j_{b+w}},$$

where $P_{i_{a+1}} = P_{j_{b+w}}$. This $\bar{\mathscr{P}}^2_{b+w}$ exists since $\mathscr{A}_2(P_{j_b})$ contains $P_{j_{b+w}}$ $(= P_{i_{a+1}})$ by $\mathscr{A}_2(P_{j_b}) \supset \mathscr{A}_1(P_{i_a})$ (induction hypothesis) and $P_{j_{b+w}}$ is eventually tested in $A(g_2)$. Obviously $\bar{\mathscr{P}}^1_{a+1}$ is a subsequence of $\bar{\mathscr{P}}^2_{b+w}$. We then prove $\mathscr{A}_1(P_{i_{a+1}}) \subset \mathscr{A}_2(P_{j_{b+w}})$. Note that $\mathscr{A}_1(P_{i_{a+1}}) \subset \mathscr{A}_2(P_{j_{b+1}})$ follows from $\mathscr{A}_1(P_{i_a}) \subset \mathscr{A}_2(P_{j_b})$ (induction hypothesis) and the fact that P_{i_a} is decomposed in $A(g_1)$ only if P_{j_b} is decomposed in $A(g_2)$. By definition of heuristic search,

$$h(P_{i_{a+1}}) = \min \{h(P) \mid P \in \mathscr{A}_1(P_{i_{a+1}})\}$$

and all nodes $P_{j_{b+v}}$, $1 \leq v < w$, satisfy

$$h(P_{j_{b+v}}) < h(P_{j_{b+w}}) \ (=h(P_{i_{a+1}})),$$

i.e., $P_{j_{b+v}} \notin \mathscr{A}_1(P_{i_{a+1}})$ for $v = 1, 2, \ldots, w-1$. This proves $\mathscr{A}_1(P_{i_{a+1}}) \subset \mathscr{A}_2(P_{j_{b+w}})$. To prove $z_1(P_{i_{a+1}}) = z_2(P_{j_{b+w}})$, note that a proper ancestor P_{i_e} $(1 \leq e \leq a)$ of each $P_{j_{b+v}}$ $(1 \leq v < w)$ (we assume $w > 1$ since otherwise the proof is trivial) has been terminated in $A(g_1)$ by $\mathscr{G}$ or by lower bound test. (If P_{i_e} is terminated in $A(g_1)$ by dominance test, it is also terminated in $A(g_2)$ by dominance test since $\mathscr{N}_1(P_{i_e}) \subset \mathscr{N}_2(P_{i_e})$ follows from the induction hypothesis.) Thus

$$u(P_{j_{b+v}}) \geq g_1(P_{j_{b+v}}) \geq g_1(P_{i_e})$$

and $g_1(P_{i_e}) \geq z_1(P_{i_e}) \geq z_1(P_{i_{e+1}})$ if P_{i_e} is terminated in $A(g_1)$ by $\mathscr{G}$ or by lower bound test. This shows $u(P_{j_{b+v}}) \geq z_1(P_{i_{a+1}})$, i.e., z_1 would not have been improved even if $P_{j_{b+v}}$ is tested in $A(g_1)$, implying $z_1(P_{i_{a+1}}) \leq z_2(P_{j_{b+w}})$. On the other hand $z_1(P_{i_{a+1}}) \geq z_2(P_{j_{b+w}})$ since $\bar{\mathscr{P}}^1_{a+1}$ is a subsequence of $\bar{\mathscr{P}}^2_{b+w}$, and hence $z_1(P_{i_{a+1}}) = z_2(P_{j_{b+w}})$.

Consequently $\bar{\mathscr{P}}^1$ is a subsequence of $\bar{\mathscr{P}}^2$, and $T_a(g_1) \leq T_a(g_2)$ is an immediate consequence of it. To treat the B-count, let

$$\mathscr{P}^1_{\text{opt}} = \{P_i \in \mathscr{P} \mid P_i \in \mathscr{G}_1,\ f(P_i) = f(P_0),\ \text{and no proper ancestor of } P_i \text{ belongs to } \mathscr{G}_1\}.$$

$\mathscr{P}^2_{\text{opt}}$ is similarly defined. Obviously $B_a(g_k)$ is equal to the number of nodes decomposed in $A(g_k)$ before all nodes in $\mathscr{P}^k_{\text{opt}}$ are tested. By $\mathscr{G}_1 \supset \mathscr{G}_2$, it follows that, for any $P_j \in \mathscr{P}^2_{\text{opt}}$, there exists $P_i \in \mathscr{P}^1_{\text{opt}}$ such that P_i is an ancestor (including the case of $P_i = P_j$) of P_j. This property and that $\bar{\mathscr{S}}^1$ is a subsequence of $\bar{\mathscr{S}}^2$ lead to $B_a(g_1) \leq B_a(g_2)$. □

A property similar to Theorem 3.3 has been known as Theorem 2 of [13] (see [14] for the complete proof), though there are some nontrivial deffer-ences such as [13] treats only depth-first and breadth-first search functions, definitions of u and D are slightly different, and [13] does not discuss the B-count.

4. Power of Lower Bounding Functions under Best-Bound Search

Along the line discussed in Section 3, we treat in this section two branch-and-bound algorithms $A(g_1) = ((\mathscr{B}, o, f), (\mathscr{G}_1, g_1, u), D, s_{g_1})$ and $A(g_2) = ((\mathscr{B}, o, f), (\mathscr{G}_2, g_2, u), D, s_{g_2})$ using best-bound search functions s_{g_1} and s_{g_2} that are usually different. Classes of branch-and-bound algo-rithms in which $g_1 \geq g_2$ implies $T(g_1) \leq T(g_2)$ and $B(g_1) \leq B(g_2)$ are clarified. In addition, for some classes not satisfying this monotomic relation, weak statements such as $T(g_1) \leq T(g_2)+\varepsilon$ and $B(g_1) \leq B(g_2)+\varepsilon$ are proved for some small positive number ε.

The effect of g under best-bound search was first investigated in [8, 18] (see also [5]); special cases of Theorem 4.4 (i) - (iii) were therein proved. The phenomenon that $g_1 \geq g_2$ does not necessarily imply $T_s(g_1) \leq T_s(g_2)$ (the first half of Theorem 4.4) was also observed in [13].

When best-bound search is concerned, it may not be reasonable to assume $g(P_i) \neq g(P_j)$ for all pairs $P_i \neq P_j$. In case $g(P_i) = g(P_j)$ holds for some $P_i \neq P_j$, some tie breaking rule is used to determine the node selected first. A tie breaking rule which selects P_i before P_j if $g(P_i) = g(P_j) \wedge (\exists$ a proper descendant P_k of $P_i)$ $(P_k DP_j)$ is sometimes used in the following discussion. If s_g uses such a tie breaking rule, it is said to be *compatible with* D. (As an example consider the case in which $P_i DP_j$ occurs only if P_i and P_j are in the same depth. Then a tie breaking rule putting a priority on nodes with

smaller depth results in an s_g compatible with D.) Usually, however, we will not assume any particular tie breaking rule unless otherwise stated.

Theorem 4.1. Let $A(g_1) = ((\mathscr{B}, O, f), (\mathscr{G}_1, g_1, u), D, s_{g_1})$ and $A(g_2) = ((\mathscr{B}, O, f), (\mathscr{G}_2, g_2, u), D, s_{g_2})$ be branch-and-bound algorithms using best-bound search. Then $g_1 \geq g_2$ does not necessarily imply $T(g_1) \leq T(g_2)$ or $B(g_1) \leq B(g_2)$.

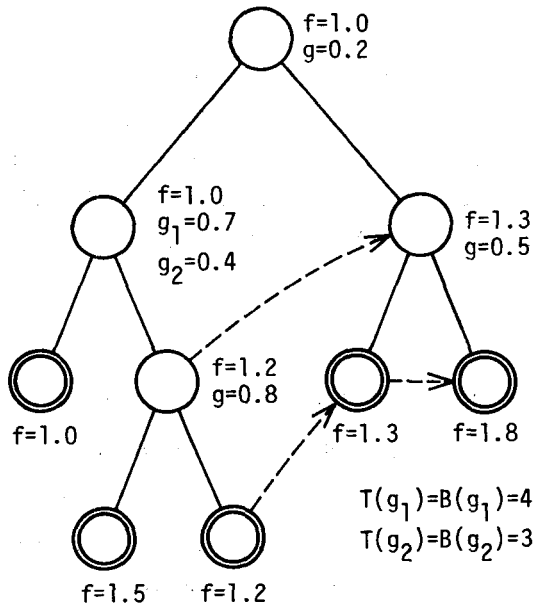


Fig. 4 Counterexample to the conjecture $g_1 \geq g_2 \Rightarrow T(g_1) \leq T(g_2) \wedge B(g_1) \leq B(g_2)$ under best-bound search in the proof of Theorem 4.1. ($u=\infty$ is assumed.)

Proof. The example of Fig. 4 has $T(g_1) = B(g_1) = 4 > T(g_2) = B(g_2) = 3$, in spite of $g_1 \geq g_2$. □

Note that D used in Fig. 4 is not consistent with g_1. Under the consistency assumption, stronger statements can be made as shown in the rest of this section. The following lemma is useful to prove them.

Lemma 4.2. Let $A(g) = ((\mathscr{B}, O, f), (\mathscr{G}, g, u), D, s_g)$ be a branch-and-bound algorithm using best-bound search, where D is consistent with g. Define

$$\mathscr{F} = \{P_i \in \mathscr{P} \mid g(P_i) \leq f(P_0)\} \tag{4.1}$$

$$\mathscr{P}_D = \{P_i \in \mathscr{P} \mid \text{no } P_j (\neq P_i) \in \mathscr{P} \text{ satisfies } P_j D P_i\}. \tag{4.2}$$

Then $T_a(g) = |\mathscr{F} \cap \mathscr{P}_D - \mathscr{G}|$.

Proof. It is known (e.g., lemma 6.1 of [11] and Lemma 3 of [8]) that $g(P_{i_1}) \leq g(P_{i_2})$ holds if P_{i_2} is selected after P_{i_1} in $A(g)$, and that z is set to $f(P_0)$ before any P_i with $g(P_i) > f(P_0)$ is selected. When all optimal solutions are sought, therefore, P_i is terminated by $\mathcal{G}$ or by lower bound test if and only if $P_i \notin \mathcal{F} - \mathcal{G}$. Next consider a node $P_i \in \mathcal{F} - \mathcal{G}$ and assume that there exists $P_j (\neq P_i)$ satisfying $P_j D P_i$, i.e., $P_i \notin \mathcal{P}_D$. Then $g(P_j) < g(P_i)$ since D is consistent with g. Thus $P_j \in \mathcal{N}(P_i)$ if P_j is eventually generated in $A(g)$, implying that P_i is terminated by dominance test. On the other hand, if P_j is not generated in $A(g)$, a proper ancestor P_k of P_j must have been terminated by dominance test, i.e., $P_\ell D P_k$ for some other $P_\ell \in \mathcal{N}(P_k)$ (note that P_k is not terminated by $\mathcal{G}$ or by lower bound test since $g(P_k) \leq g(P_j) < g(P_i) \leq f(P_0)$ implies $P_k \in \mathcal{F} - \mathcal{G}$). By condition $(ii)_a$ of D, there exists a descendant P_m of P_ℓ such that $P_m D P_j$ (see Fig. 5). If P_m is eventually generated in $A(g)$, it is generated before P_i since $g(P_m) < g(P_j) < g(P_i)$. In addition, $P_m D P_i$ holds by the transitivity of D (note that D is a partial ordering). Thus P_i is terminated by dominance test. On the other hand, if P_m is not generated in $A(g)$, we can repeat the above argument. However, this process can not be repeated indefinitely since $\mathcal{P}$ is finite; showing that P_i is terminated by dominance test. Consequently a node $P_i \in \mathcal{F} - \mathcal{G}$ is terminated by dominance test if and only if $P_i \notin \mathcal{P}_D$. Therefore we have $T_a(g) = |\mathcal{F} \cap \mathcal{P}_D - \mathcal{G}|$. □

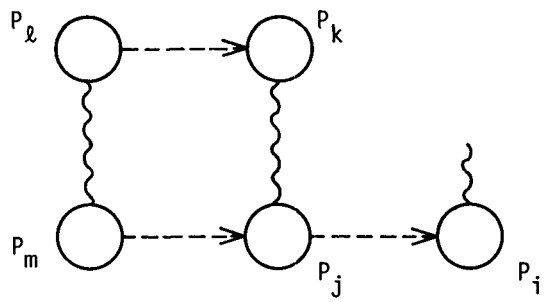


Fig. 5 Relative positions of P_i, P_j, P_k, P_ℓ, P_m used in the proof of Theorem 4.2.

Theorem 4.3. Let $A(g_1)$ and $A(g_2)$ be defined as in Theorem 4.1, where $A(g_1)$ and $A(g_2)$ use best-bound search. In addition, assume that D is consistent with g_1. Then $g_1 \geq g_2$ implies $T_a(g_1) \leq T_a(g_2)$ and $B_a(g_1) \leq B_a(g_2) + |\mathcal{X}^* \cap \mathcal{P}_D - \mathcal{G}_1|$, where

$$\mathcal{X}^* = \{P_i \in \mathcal{P} \mid g_1(P_i) = g_2(P_i) = f(P_0)\}. \tag{4.3}$$

Proof. $T_a(g_1) \le T_a(g_2)$: By Lemma 4.2, we have

$$T_a(g_1) = |\mathscr{F}_1 \cap \mathscr{P}_D - \mathscr{G}_1|, \quad T_a(g_2) \ge |\mathscr{F}_2 \cap \mathscr{P}_D - \mathscr{G}_2|, \tag{4.4}$$

where $\mathscr{F}_k$ is defined by (4.1) for $A(g) = A(g_k)$, k=1, 2. Note that the second relation is inequality because D may not be consistent with g_2. Since $g_1 \ge g_2$ implies $\mathscr{F}_1 \subset \mathscr{F}_2$ and $\mathscr{G}_1 \supset \mathscr{G}_2$, $T_a(g_1) \le T_a(g_2)$ immediately follows.

$B_a(g_1) \le B_a(g_2) + |\mathscr{K}^* \cap \mathscr{P}_D - \mathscr{G}_1|$: Let

$$\mathscr{H}_1 = \{P_i \in \mathscr{P} \mid g_1(P_i) < f(P_0),\ g_2(P_i) < f(P_0)\} \tag{4.5}$$

$$\mathscr{K}_1' = \{P_i \in \mathscr{P} \mid g_1(P_i) = f(P_0),\ g_2(P_i) < f(P_0)\}. \tag{4.6}$$

Under best-bound search, nodes in $\mathscr{H}_1 \cup \mathscr{K}_1'$ are tested in $A(g_2)$ before nodes in $\mathscr{K}^*$. Furthermore, all nodes P_j satisfiying $P_j \in \mathscr{G}_2 \wedge f(P_j) = f(P_0)$ belong to $\mathscr{K}^*$. Thus, denoting the set of nodes decomposed in $A(g_2)$ by $\mathscr{P}_2$, we have

$$|(\mathscr{H}_1 \cup \mathscr{K}_1') \cap \mathscr{P}_2| \le B_a(g_2).$$

Next note that nodes in $\mathscr{H}_1$ are tested in $A(g_1)$ before nodes in $\mathscr{K}_1' \cup \mathscr{K}^*$, and nodes P_j satisfying $P_j \in \mathscr{G}_1 \wedge f(P_j) = f(P_0)$ belong to $\mathscr{K}_1' \cup \mathscr{K}^*$. Thus, denoting the set of nodes decomposed in $A(g_1)$ by $\mathscr{P}_1$, we have

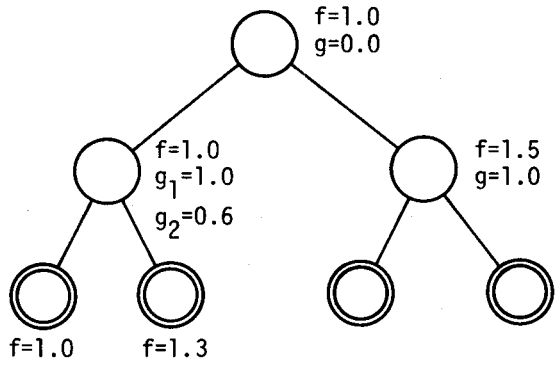
$$\begin{aligned} B_a(g_1) &\le |\mathscr{H}_1 \cap \mathscr{P}_1| + |(\mathscr{K}_1' \cup \mathscr{K}^*) \cap \mathscr{P}_1| \\ &= |\mathscr{H}_1 \cap \mathscr{P}_1| + |\mathscr{K}_1' \cap \mathscr{P}_1| + |\mathscr{K}^* \cap \mathscr{P}_1| \\ &\le |(\mathscr{H}_1 \cup \mathscr{K}_1') \cap \mathscr{P}_2| + |\mathscr{K}^* \cap \mathscr{P}_D - \mathscr{G}_1| \\ &\qquad \text{(by } \mathscr{P}_2 \supset \mathscr{P}_1 \text{ and by } \mathscr{K}^* \cap \mathscr{P}_1 = \mathscr{K}^* \cap \mathscr{P}_D - \mathscr{G}_1 \\ &\qquad \text{derived from Lemma 4.2)} \\ &\le B_a(g_2) + |\mathscr{K}^* \cap \mathscr{P}_D - \mathscr{G}_1|. \quad \square \end{aligned}$$

The term $|\mathscr{K}^* \cap \mathscr{P}_D - \mathscr{G}_1|$ is necessary in the above theorem because a best-bound search function s_g (without a tie breaking rule) does not decide which nodes in $\mathscr{K}^*$ should be tested first. This term, however, seems to be very small in most cases encountered in practice.

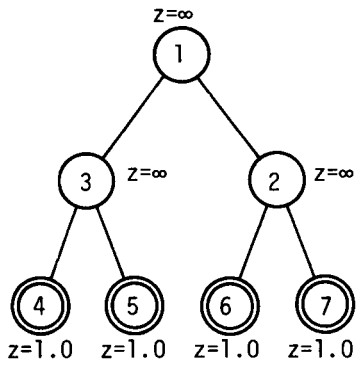
When a single optimal solution is sought, the situation is somewhat different. Theorems 4.4 and 4.5 below summarize the results for T and B respectively. For proofs of these results, see [12].

Theorem 4.4. Let $A(g_1)$ and $A(g_2)$ be defined as in Theorem 4.1, where $A(g_1)$ and $A(g_2)$ use best-bound search. Furthermore assume that D is consistent

with g_1. In general, $g_1 \geq g_2$ does not necessarily imply $T_s(g_1) \leq T_s(g_2)$ even if $D=I$ is assumed (as shown in Fig. 6). However, the following properties are true.

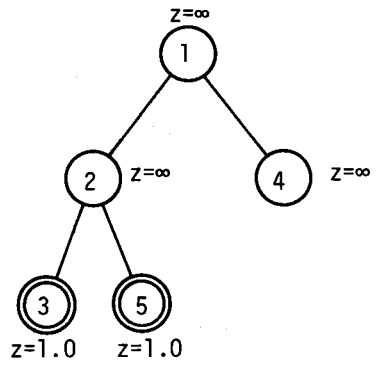


(a) ($\mathcal{B}$, f), g_1 and g_2 ($u=\infty$ is assumed)



$T_s(g_1)=B_s(g_1)=3$

(b) Computational process of $A(g_1)$ (The tie breaking rule selects node 2 prior to node 3 though both have the same g_1-value.)



$T_s(g_2)=B_s(g_2)=2$

(c) Computational process of $A(g_2)$ (The tie breaking rule selects node 3 prior to node 4 though both have the same g_2-value.)

Fig. 6 Counterexample to the conjecture $g_1 \geq g_2 \Rightarrow T_s(g_1) \leq T_s(g_2) \wedge B_s(g_1) \leq B_s(g_2)$ under best-bound search and $D=I$, used in the proof of Theorem 4.4.

(i) If s_{g_1} is compatible with D, then

$$g_1 \geq g_2 \Rightarrow T_s(g_1) \leq T_s(g_2) + |\mathscr{K}^* \cap \mathscr{P}_D - \mathscr{G}_1|.$$

(ii) If s_{g_1} is compatible with D and $g_1(P_i) > g_2(P_i)$ for $P_i \in \mathscr{P} - \mathscr{G}_1$, then $g_1 \geq g_2$ implies $T_s(g_1) \leq T_s(g_2)$.

(iii)† If $g_1(P_j) > g_1(P_i)$ for every $(P_i, P_j) \in \mathscr{E}$ with $P_i \in \mathscr{P} - \mathscr{G}_1$, and if $P_k \in \mathscr{G}_1$ has higher priority than $P_\ell \notin \mathscr{G}_1$ satisfying $g_1(P_\ell) = g_1(P_k)$ in selecting an active node in A2 of $A(g_1)$, then $g_1 \geq g_2$ implies $T_s(g_1) \leq T_s(g_2)$. □

Special cases of Theorem 4.4 (i) ∿ (iii) assuming a certain dominance relation specific to the shortest path problem were proved in [5, 8, 15, 18]. (Their results are more general in the sense that $\mathscr{B}$ could be infinite.)

Theorem 4.5. Let $A(g_1)$ and $A(g_2)$ be defined as in Theorem 4.1, where $A(g_1)$ and $A(g_2)$ use best-bound search. Furthermore assume that D is consistent with g. Although $g_1 \geq g_2$ does not necessarily imply $B_s(g_1) \leq B_s(g_2)$ even if $D=I$ is assumed, the following properties are true.

(i) If $u=\infty$ and s_{g_1} is compatible with D, then

$$g_1 \geq g_2 \Rightarrow B_s(g_1) \leq B_s(g_2) + |\mathscr{K}^* \cap \mathscr{P}_D - \mathscr{G}_1|.$$

(ii) If $u=\infty$, s_{g_1} is compatible with D and $g_1(P_i) > g_2(P_i)$ for $P_i \in \mathscr{P} - \mathscr{G}_1$, then $g_1 \geq g_2$ implies $B_s(g_1) \leq B_s(g_2)$.

(iii)† If $u=\infty$, $g_1(P_j) > g_1(P_i)$ for every $(P_i, P_j) \in \mathscr{E}$ with $P_i \in \mathscr{P} - \mathscr{G}_1$, and if $P_k \in \mathscr{G}_1$ has higher priority than $P_\ell \notin \mathscr{G}_1$ satisfying $g_1(P_\ell) = g_1(P_k)$ in selecting an active node in A2 of $A(g_1)$, then $g_1 \geq g_2$ implies $B_s(g_1) \leq B_s(g_2)$. □

The results in Sections 3 and 4 are summarized in Table 1, where — indicates that a monotone dependence of T (or B) on g is not guaranteed.

---

† The first condition on g may by changed as follows: "If s_{g_1} is compatible with D and $g_1(P_i) < g_1(P_j)$ holds for every $(P_i, P_j) \in \mathscr{E}$ with $P_i \notin \mathscr{G}_1$, $P_j \in \mathscr{G}_1$".

Table 1. Computational efficiency of $A(g_1)$ and $A(g_2)$ with $g_1 \geq g_2$.
(Entries — denote that $T(g_1) \leq T(g_2)$ or $B(g_1) \leq B(g_2)$ does not necessarily hold.)

Search Strategies \ D	General		Consistent with g	
	Properties	Theorems	Properties	Theorems
Heuristic	—	3.1	$T(g_1) \leq T(g_2)$	3.3
(General)	—	3.1	$B(g_1) \leq B(g_2)$	3.3
Heuristic	—	3.1	$T(g_1) \leq T(g_2)$	3.3
(Nonmisleading)	$B_a(g_1) \leq B_a(g_2)$	3.2	$B(g_1) \leq B(g_2)$	3.3
	$B_s(g_1) = B_s(g_2)$	3.2		
Depth-First	—	3.1	$T(g_1) \leq T(g_2)$	3.3
Breadth-First	—	3.1	$B(g_1) \leq B(g_2)$	3.3
Best-Bound[(a)]			$T_a(g_1) \leq T_a(g_2)$	4.3
	—	4.1	— (b)	4.4
	—	4.1	$B_a(g_1) \leq B_a(g_2) + \lvert \mathscr{K}^* \cap \mathscr{S}_D - \mathscr{G}_1 \rvert$	4.3
			— (c)	4.5

(a) Note that search functions also change when g is improved, in case of best-bound search.

(b) See Theorem 4.4 for detailed analysis.

(d) See Theorem 4.5 for detailed analysis.

5. Power of Upper Bounding Functions

Consider two branch-and-bound algorithms $A(u_1) = ((\mathscr{B}, O, f), (\mathscr{G}, g, u_1), D, s)$ and $A(u_2) = ((\mathscr{B}, O, f), (\mathscr{G}, g, u_2), D, s)$ with different upper bounding functions u_1 and u_2. u_1 is said to be *tighter* than u_2, denoted by $u_1 \leq u_2$, if $u_1(P_i) \leq u_2(P_i)$ holds for any $P_i \in \mathscr{P}$. Again $u_1 \leq u_2$ does not generally imply $T(u_1) \leq T(u_2)$ and $B(u_1) \leq B(u_2)$ as we shall see below. Classes of branch-and-bound algorithms for which $u_1 \leq u_2$ implies $T(u_1) \leq T(u_2)$ and $B(u_1) \leq B(u_2)$ are also clarified in this section. Properties of u along this line were first examined in [13], and a result similar to Theorem 5.4 was

therein obtained (though [13] did not consider the B-count, treated only upper bounding functions of type $u = u(P_0)$, and did not assume the general class of heuristic search).

Theorem 5.1. Let $A(u_1) = ((\mathscr{B}, O, f), (\mathscr{G}, g, u_1), D, s_h)$ and $A(u_2) = ((\mathscr{B}, O, f), (\mathscr{G}, O, u_2), D, s_h)$ be branch-and-bound algorithms using heuristic search. Then $u_1 \leq u_2$ does not necessarily imply $T(u_1) \leq T(u_2)$ or $B(u_1) \leq B(u_2)$. This is true even if s_h is restricted to be a depth-first search function or a breadth-first search function.

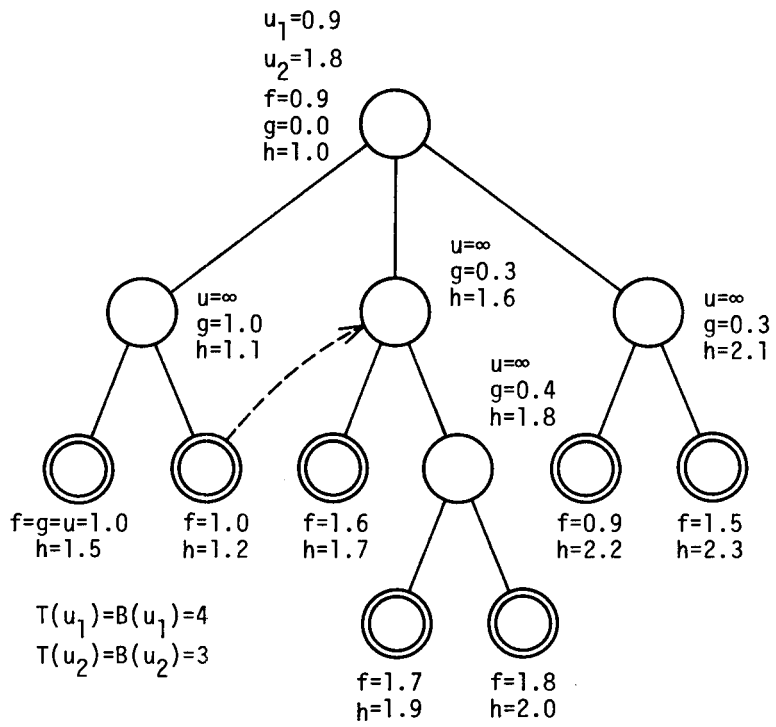


Fig. 7 Counterexample to the conjecture $u_1 \leq u_2 \Rightarrow T(u_1) \leq T(u_2) \wedge B(u_1) \leq B(u_2)$ under heuristic search, used in the proof of Theorem 5.1. ($u_k = u_k(P_0)$ is assumed for k=1, 2. Only relevant parameters are indicated.)

Proof. The example in Fig. 7 has $T(u_1) = B(u_1) = 4 > T(u_2) = B(u_2) = 3$ in spite of $u_1 \leq u_2$. The search function s_h used here is also a depth-first search function. The case of a breadth-first search function is treated in Fig. 8, in which $T(u_1) = B(u_1) = 7 > T(u_2) = B(u_2) = 5$ in spite of $u_1 \leq u_2$. □

Note that the examples used in the above proof have upper bounding functions of type $u_k = u_k(P_0)$. Thus Theorem 5.1 holds even if u_k is restricted

to be of type $u_k = u_k(P_0)$. For best-bound search, however, $u_1 \leq u_2$ always results in $T(u_1) \leq T(u_2)$ and $B(u_1) \leq B(u_2)$.

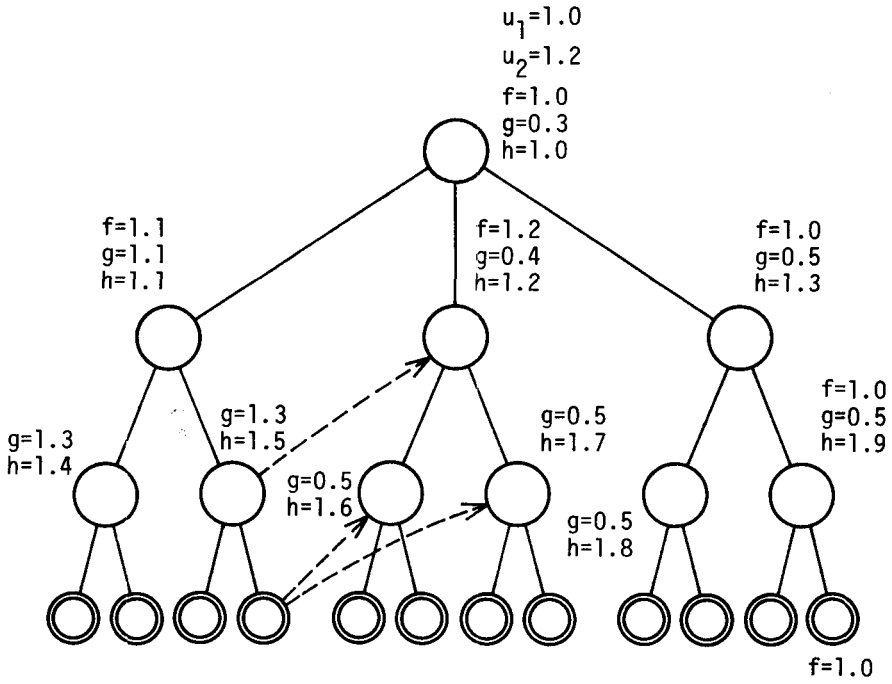


Fig. 8 Counterexample to the conjecture $u_1 \leq u_2 \Rightarrow T(u_1) \leq T(u_2) \wedge B(u_1) \leq B(u_2)$ under breadth-first search, used in the proof of Theorem 5.1. ($u_k = u_k(P_0)$ is assumed for $k=1, 2$. Only relevant parameters are indicated.)

Theorem 5.2. Let $A(u_1) = ((\mathscr{B}, O, f), (\mathscr{G}, g, u_1), D, s_g)$ and $A(u_2) = ((\mathscr{B}, O, f), (\mathscr{G}, g, u_2), D, s_g)$ be branch-and-bound algorithms using best-bound search. Then $u_1 \leq u_2$ implies $T_a(u_1) = T_a(u_2)$, $B_a(u_1) = B_a(u_2)$, $T_s(u_1) \leq T_s(g_2)$ and $B_s(u_1) \leq B_s(u_2)$.

Proof. When all optimal solutions are sought, computational processes of $A(u_1)$ and $A(u_2)$ proceed independently of u_1 and u_2, since P_i is decomposed if and only if $P_i \in \mathscr{F} \cap \mathscr{P}_D - \mathscr{G}$ (this set does not depend on u_1 or u_2) by Lemma 4.2. This proves $T_a(u_1) = T_a(u_2)$ and $B_a(u_1) = B_a(u_2)$.

When a single optimal solution is sought, let $\bar{\mathscr{P}}^1 = P_{i_1} P_{i_2} \cdots P_{i_s}$ and $\bar{\mathscr{P}}^2 = P_{j_1} P_{j_2} \cdots P_{j_t}$ be the sequences of nodes selected in $A(u_1)$ and $A(u_2)$ respectively. It is shown by induction that $\bar{\mathscr{P}}^1$ is a subsequence of $\bar{\mathscr{P}}^2$.

To prove this, assume that for $\bar{\mathcal{P}}^1_a = P_{i_1} P_{i_2} \ldots P_{i_a}$, $1 \le a \le s$, there exists $\bar{\mathcal{P}}^2_b = P_{j_1} P_{j_2} \ldots P_{j_b}$ satisfying that $P_{i_a} = P_{j_b}$, $\mathcal{N}_1(P_{i_a}) \subset \mathcal{N}_2(P_{j_b})$, $z_1(P_{i_a}) \le z_2(P_{j_b})$ and $\bar{\mathcal{P}}^1_a$ is a subsequence of $\bar{\mathcal{P}}^2_b$. For $a = 1$, this induction hypothesis is trivially true since $\bar{\mathcal{P}}^2_1$ satisfies the above conditions. In a general case, we first show that P_{i_a} is terminated in $A(u_1)$ if P_{j_b} is terminated in $A(u_2)$. Three cases are considered.

(a) P_{j_b} is terminated in $A(u_2)$ by $\mathcal{G}$: Then P_{i_a} $(=P_{j_b})$ is also terminated in $A(u_1)$ by $\mathcal{G}$.

(b) P_{j_b} is terminated in $A(u_2)$ by lower bound test, i.e., $g(P_{j_b}) \ge z_2(P_{j_b})$: Then $z_2(P_{j_b}) = f(P_0)$ holds as a characteristic of best-bound search, and there exists P_{j_p} $(1 \le p \le b)$ satisfying $u_2(P_{j_p}) = f(P_0)$ in $\bar{\mathcal{P}}^2_b$. Assume that a proper ancestor P_{i_q} of P_{j_p} has been terminated in $A(u_1)$, since otherwise $z_1(P_{i_a}) \le u_1(P_{j_p}) \le u_2(P_{j_p}) = f(P_0) \le g(P_{j_b}) = g(P_{j_a})$ and hence P_{i_a} is terminated in $A(u_1)$ by lower bound test. If P_{i_q} has been terminated in $A(u_1)$ by lower bound test, it follows

$$
\begin{aligned}
g(P_{i_a}) &\ge g(P_{i_q}) \quad (\text{since } P_{i_a} \text{ is selected after } P_{i_q}) \\
&\ge z_1(P_{i_q}) = f(P_0) \qquad (5.1) \\
&\ge z_1(P_{i_a}) \quad (\text{since } P_{i_a} \text{ is selected after } P_{i_q})
\end{aligned}
$$

and P_{i_a} is terminated in $A(u_1)$ by lower bound test. On the other hand, if P_{i_q} has been terminated by dominance test $P_{i_r} DP_{i_q}$ for $P_{i_r} \in \mathcal{N}_1(P_{i_q})$, we have $\mathcal{N}_2(P_{i_q}) \supset \mathcal{N}_1(P_{i_q})$ (this follows from the induction hypothesis) and P_{i_q} must have been terminated in $A(u_2)$ by dominance test. This contradicts that P_{j_p} (a proper descendant of P_{i_q}) was generated in $A(u_2)$.

(c) P_{j_b} is terminated in $A(u_2)$ by dominance test $P_{j_c} DP_{j_b}$ for $P_{j_c} \in \mathcal{N}_2(P_{j_b})$: Assume that a proper ancestor P_{i_d} of P_{j_c} has been terminated in $A(u_1)$ since otherwise P_{i_a} is also terminated in $A(u_1)$ by dominance test. For simplicity, assume that P_{i_d} has been terminated by lower bound test. (If P_{i_d} has been terminated by dominance test, the argument used for Fig. 3 in the proof of Theorem 3.3 can be used. The following argument is also valid for this case with minor modification.) Then it follows.

$$
\begin{aligned}
g(P_{i_a}) &\geq g(P_{i_d}) \text{ (since } P_{i_a} \text{ is selected after } P_{i_d}) \\
&\geq z_1(P_{i_d}) \geq z_1(P_{i_a})
\end{aligned} \tag{5.2}
$$

and P_{i_a} is terminated in $A(u_1)$ by lower bound test.

Now assume that $\bar{\mathscr{S}}^1_a \neq \bar{\mathscr{S}}^1$ (since otherwise the proof is done) and define the following two sequences

$$\bar{\mathscr{S}}^1_{a+1} = P_{i_1} P_{i_2} \cdots P_{i_a} P_{i_{a+1}}$$

$$\bar{\mathscr{S}}^2_{b+w} = P_{j_1} P_{j_2} \cdots P_{j_b} \cdots P_{j_{b+w}},$$

such that $P_{i_{a+1}} = P_{j_{b+w}}$. $\bar{\mathscr{S}}^1_{a+1}$ is a subsequence of $\bar{\mathscr{S}}^2_{b+w}$. $\mathscr{A}_1(P_{i_{a+1}}) \subset \mathscr{A}_2(P_{j_{b+w}})$ can be proved in a manner similar to the proof of Theorem 3.3. To prove $z_1(P_{i_{a+1}}) \leq z_2(P_{j_{b+w}})$, assume contrary, i.e., $z_1(P_{i_{a+1}}) > z_2(P_{j_{b+w}})$. Then there exists $P_{j_{b+k}}$ $(1 \leq k < w)$ such that $z_2(P_{j_{b+w}}) = u_2(P_{j_{b+k}}) < z_1(P_{i_{a+1}})$. Since $P_{j_{b+k}}$ is not generated in $A(u_1)$, there is $P_{i_u} = P_{j_v}$ in $\bar{\mathscr{S}}^1_a$ and $\bar{\mathscr{S}}^2_b$ respectively such that (1) P_{j_v} is a proper ancestor of $P_{j_{b+k}}$ and (2) P_{i_u} is terminated in $A(u_1)$ but P_{j_v} is not terminated in $A(u_2)$. (2) is possible only if P_{i_u} is terminated by lower bound test, as obvious from the above proof. Therefore

$$z_1(P_{i_u}) \leq g(P_{i_u}) = g(P_{j_v}) < z_2(P_{j_v}),$$

and $P_{j_{b+k}}$ (a descendant of P_{j_v}) satisfies

$$(z_1(P_{i_{a+1}}) \leq)\ z_1(P_{i_u}) \leq g(P_{j_v}) \leq g(P_{j_{b+k}}) \leq u_2(P_{j_{b+k}}),$$

which is a contradiction. This proves $z_1(P_{i_{a+1}}) \leq z_2(P_{j_{b+w}})$.

Consequently $\bar{\mathscr{S}}^1$ is a subsequence of $\bar{\mathscr{S}}^2$, and $T_s(u_1) \leq T_s(u_2)$ immediately follows. $B_s(u_1) \leq B_s(u_2)$ can also be proved in a manner similar to the proof of $B_a(u_1) \leq B_a(u_2)$ in Theorem 3.3 (use also the property $z_1(P_{i_a}) \leq z_2(P_{j_b})$ for $P_{i_a} = P_{j_b}$). □

Remark. In the above proof, it is assumed that $A(u_1)$ and $A(u_2)$ use the same search function including the tie breaking rule. In other words, if P_i, $P_j \in \mathscr{A}_1$, and P_i is selected before P_j in $A(u_1)$, then P_i is selected before

P_j in $A(u_2)$ when P_i, $P_j \in \mathscr{A}_2$.

The next theorem treats a special case of heuristic search, i.e., when h of s_h is nonmisleading.

Theorem 5.3. Let $A(u_1)$ and $A(u_2)$ be defined as in Theorem 5.1, where $A(u_1)$ and $A(u_2)$ use the same heuristic search function s_h. Furthermore assume that h is nonmisleading. Then $u_1 \leq u_2$ implies $T_a(u_1) = T_a(u_2)$, $B_a(u_1) = B_a(u_2)$ and $B_s(u_1) \leq B_s(u_2)$, but does not necessarily imply $T_s(u_1) \leq T_s(u_2)$.

Proof. The last result is shown by the example given in Fig. 9, in which $T_s(u_1) = 3 > T_s(u_2) = 2$ holds though $u_1 \leq u_2$ holds and h is nonmisleading.

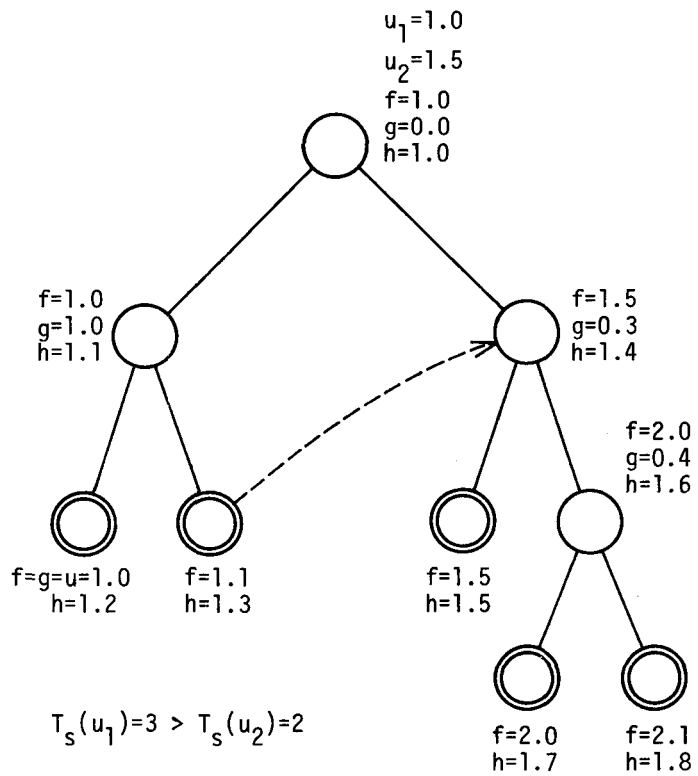


Fig. 9 Counterexample to the conjecture $u_1 \leq u_2 \Rightarrow T_s(u_1) \leq T_s(u_2)$ under nonmisleading heuristic search, used in the proof of Theorem 5.3. ($u_k = u_k(P_0)$ is assumed for k=1, 2.)

$T_a(u_1) = T_a(u_2)$ and $B_a(u_1) = B_a(u_2)$: Let $\bar{\mathscr{S}}_I = P_{i_1} P_{i_2} \ldots P_{i_p}$ be the initial portion of the sequence of the nodes selected in $A(u_1)$ such that P_{i_p} is the last node satisfying $P_{i_p} \in \mathscr{G} \wedge f(P_{i_p}) = f(P_0)$. We show below that

$A(u_2)$ also selects exactly the same sequence of nodes and that P_{i_r} $(1 \leq r \leq p)$ is decomposed in $A(u_1)$ if and only if it is decomposed in $A(u_2)$. Then it implies that P_{i_p} is also the last node selected in $A(u_2)$ such that $P_{i_p} \in \mathscr{G} \wedge f(P_{i_p}) = f(P_0)$, since $\mathscr{G}$ and f are independent of u_1 and u_2. This proves $B_a(u_1) = B_a(u_2)$. To prove $T_a(u_1) = T_a(u_2)$, note that z_1 and z_2 are set to $f(P_0)$ when $A(u_1)$ and $A(u_2)$ complete the $\bar{\mathscr{S}}_I$ portion. The rest of computation then proceeds independently of u_1 and u_2. Thus $T_a(u_1) = T_a(u_2)$ follows.

Now to prove the above assertion by induction, assume that $\bar{\mathscr{S}}_a = P_{i_1} P_{i_2} \ldots P_{i_a}$ $(1 \leq a \leq p)$ is the sequence of the first a nodes selected in both $A(u_1)$ and $A(u_2)$, and that $\mathscr{A}_1(P_{i_a}) = \mathscr{A}_2(P_{i_a})$ holds. For $a = 1$, this is trivially true. Let $a < p$ since otherwise the proof is done. We show that P_{i_a} is terminated in $A(u_1)$ if and only if it is terminated in $A(u_2)$. (Then the induction can proceed one step, and it completes the proof.) First note that $f(P_{i_1}) = f(P_{i_2}) = \ldots = f(P_{i_a}) = f(P_0)$ holds as a characteristic of a nonmisleading heuristic search function (Lemma 5.1 of [9]), and that $z_2(P_{i_a}) \geq z_1(P_{i_a}) \geq f(P_0)$. Thus $g(P_{i_a}) \leq f(P_0) \leq z_1(P_{i_a}) \leq z_2(P_{i_a})$ and P_{i_a} is not terminated in $A(u_1)$ or $A(u_2)$ by lower bound test. Furthermore the induction hypothesis implies $\mathscr{N}_1(P_{i_a}) = \{P_{i_1}, P_{i_2}, \ldots, P_{i_{a-1}}\} \cup \mathscr{A}_1(P_{i_a}) = \{P_{i_1}, P_{i_2}, \ldots, P_{i_{a-1}}\} \cup \mathscr{A}_2(P_{i_a}) = \mathscr{N}_2(P_{i_a})$. Thus P_{i_a} is terminated in $A(u_1)$ by dominance test if and only if it is terminated in $A(u_2)$ by dominance test. This completes the proof.

A proof for $B_s(u_1) \leq B_s(u_2)$ is similar [12]. □

Theorem 5.4. Let $A(u_1)$ and $A(u_2)$ be defined as in Theorem 5.1, where $A(u_1)$ and $A(u_2)$ use heuristic search. Furthermore assume that D is consistent with g. Then $u_1 \leq u_2$ implies $T(u_1) \leq T(u_2)$ and $B(u_1) \leq B(u_2)$.

Proof. We consider the case of all optimal solutions. The case of a single optimal solution is similar. The proof is done by slightly modifying the proof of Theorem 5.2 (the case of a single optimal solution). Assume that $\bar{\mathscr{S}}_a^1$ and $\bar{\mathscr{S}}_b^2$ satisfy the same induction hypothesis as in Theorem 5.2. To prove that P_{i_a} is terminated in $A(u_1)$ if P_{j_b} is terminated in $A(u_2)$, three cases are considered.

(a) P_{j_b} is terminated in $A(u_2)$ by $\mathscr{G}$: Then P_{i_a} $(=P_{j_b})$ is also terminated by $\mathscr{G}$.

(b) P_{j_b} is terminated in $A(u_2)$ by lower bound test: The proof of

Theorem 5.2 ((b)-part) can be used after changing (5.1) to:

$$(5.3)\qquad g(P_{i_a})=g(P_{j_b})>z_2(P_{j_b})=u_2(P_{j_p})\geq g(P_{j_p})\geq g(P_{i_q})>z_1(P_{i_q})\geq z_1(P_{i_a}).$$

(c) P_{j_b} is terminated in $A(u_2)$ by dominance test: The proof of 5.2 ((c)-part) can be used after changing (5.2) to:

$$(5.4)\qquad \begin{aligned} g(P_{i_a}) &> g(P_{j_c}) \text{ (since } D \text{ is consistent with } g)\\ &\geq g(P_{i_d}) > z_1(P_{i_d}) \geq z_1(P_{i_a}). \end{aligned}$$

The construction of $\bar{\mathscr{T}}^1_{a+1}$ and $\bar{\mathscr{T}}^2_{b+w}$ to complete the induction step can also be done in the same manner as Theorem 5.2. Thus we have $T_a(u_1) \leq T_a(u_2)$ and $B_a(u_1) \leq B_a(u_2)$. □

The results of this section are summarized in Table 2.

Table 2. Computational efficiency of $A(u_1)$ and $A(u_2)$ with $u_1 \leq u_2$. (Entries — denote that $T(u_1) \leq T(u_2)$ or $B(u_1) \leq B(u_2)$ does not necessarily hold.)

Search Strategies \ D	General		Consistent with g	
	Properties	Theorems	Properties	Theorems
Heuristic	—	5.1	$T(u_1) \leq T(u_2)$	5.4
(General)	—	5.1	$B(u_1) \leq B(u_2)$	5.4
Heuristic	$T_a(u_1) = T_a(u_2)$	5.3	$T_a(u_1) = T_a(u_2)$	5.3
(Nonmisleading)	—	5.3	$T_s(u_1) \leq T_s(u_2)$	5.4
	$B_a(u_1) = B_a(u_2)$	5.3	$B_a(u_1) = B_a(u_2)$	5.3
	$B_s(u_1) \leq B_s(u_2)$	5.3	$B_s(u_1) \leq B_s(u_2)$	5.3
Depth-First	—	5.1	$T(u_1) \leq T(u_2)$	5.4
Breadth-First	—	5.1	$B(u_1) \leq B(u_2)$	5.4
Best-Bound	$T_a(u_1) = T_a(u_2)$	5.2	$T_a(u_1) = T_a(u_2)$	5.2
	$T_s(u_1) \leq T_s(u_2)$	5.2	$T_s(u_1) \leq T_s(u_2)$	5.2
	$B_a(u_1) = B_a(u_2)$	5.2	$B_a(u_1) = B_a(u_2)$	5.2
	$B_s(u_1) \leq B_s(u_2)$	5.2	$B_s(u_1) \leq B_s(u_2)$	5.2

6. Further Comments

We have extensively studied how u and g affect the T-count and B-count. Another count often used to measure the performance of a branch-and-bound algorithm A is the required memory size. This is usually evaluated by

$M(A)$: The maximum size of $\mathscr{A}$ attained during the execution of A.

It may be possible to develop a similar theory treating how $M(A)$ depends on u and g. Some results are included in [12].

Finally, in concluding this paper, we emphasize that the results in this paper is primarily of theoretical interest. Even if an improvement of u and g possibly makes the resulting algorithm less efficient, our empirical knowledge tells that such phenomenon occurs extremely rarely. Thus an effort should always be directed to obtain tighter upper and lower bounding functions when we want to design efficient branch-and-bound algorithms.

Acknowledgement

The author wishes to thank Professors H. Mine and T. Hasegawa of Kyoto University for their comments. This work is partially supported by Scientific Research Grant-In-Aid from the Ministry of Education, Science and Culture, Japan.

References

[1] Agin, N.: Optimum Seeking with Branch and Bound. *Management Science*, Vol.13 (1966), B176-B185.

[2] Balas, E.: A Note on Branch-and-Bound Principle. *Operations Research*, Vol.16 (1968), 442-445.

[3] Dakin, R. J.: A Tree Search Algorithm for Mixed Integer Programming Problems. *The Computer Journal*, Vol.8 (1965), 250-255.

[4] Fox, B. L., and Schrage, L. E.: The Values of Various Strategies in Branch-and-Bound. Technical Report, Graduate School of Business, University of Chicago, 1972.

[5] Gelperin, D.: On the Optimality of A*. *Artificial Intelligence*, Vol.8 (1977), 69-76.

[6] Geoffrion, A. M., and Marsten, R. E.: Integer Programming Algorithms: A Framework and State-of-the-Art Survey. *Management Science*, Vol.18 (1972), 465-491.

[7] Gomory, R. E.: On the Relation between Integer and Noninteger Solutions to Linear Programs. *Proc. National Academy of Science*, Vol.53 (1965), 260-265.

[8] Hart, P. E., Nilsson, N. J., and Raphael, B.: A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Trans. on System Science and Cybernetics*, Vol.SSC-4 (1968), 100-107.

[9] Ibaraki, T.: Theoretical Comparisons of Search Strategies in Branch-and-Bound Algorithms. *International J. of Computer and Information Sciences*, Vol.5 (1976), 315-344.

[10] Ibaraki, T.: Computational Efficiency of Approximate Branch-and-Bound Algorithms. *Mathematics of Operations Research*, Vol.1 (1976), 287-298.

[11] Ibaraki, T.: The Power of Dominance Relations in Branch-and-Bound Algorithms. *J. of ACM*, Vol.24 (1977), 264-279.

[12] Ibaraki, T.: The Power of Upper and Lower Bounding Functions in Branch-and-Bound Algorithms. Working Paper, Department of Applied Mathematics and Physics, Kyoto University, Japan, 1976.

[13] Kohler, W.H., and Steiglitz, K.: Characterization and Theoretical Comparison of Branch-and-Bound Algorithms for Permutation Problems. *J. of ACM*, Vol.21 (1974), 140-156

[14] Kohler, W. H.: Exact and Approximate Algorithms for Permutation Problems. Ph. D. Dissertation, Princeton University, Princeton, N.J., 1972.

[15] Kowalski, R.: Search Strategies for Theorem-Proving. In *Machine Intelligence* 5, B. Meltzer and D. Michie, eds., Edinburgh University Press, 181-201, 1970.

[16] Lawler, E. L., and Wood, D. E.: Branch-and-Bound Methods: A survey. *Operations Research*, Vol.14 (1966), 699-719.

[17] Mitten, L. G.: Branch-and-Bound Methods: General Formulation and Properties. *Operations Research*, Vol.18 (1970), 24-34.

[18] Nilsson, N. J.: *Problem-Solving Methods in Artificial Intelligence*, McGraw-Hill, New York, 1971.

[19] Pohl, I.: First Results on the Effect of Error in Heuristic Search. In *Machine Intelligence* 5, B. Meltzer and D. Michie, eds., Edinburgh University Press, 219-236, 1970.

[20] Rinnooy Kan, A. H. G.: On Mitten's Axioms for Branch-and-Bound. *Operations Research*, Vol.24 (1976), 1176-1178.

[21] Tomlin, J. A.: An Improved Branch and Bound Method for Integer Programming. *Operations Research*, Vol.19 (1971), 1070-1075.

Toshihide IBARAKI: Department of
Applied Mathematics and Physics,
Faculty of Engineering,
Kyoto University Kyoto, 606, Japan