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 $\mathcal{B}$ 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 $\mathcal{P}$) 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 $\mathcal{P}$) 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
Branch-and-Bound Algorithms

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

Let $f: \mathcal{P} \rightarrow E \cup \{\infty\}$ 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

\[ f(P_i) = \min \{ f(P_j) \mid (P_i, P_j) \in \mathcal{E} \} \tag{2.1} \]

and hence

\[ f(P_i) \leq f(P_j) \quad \text{for} \quad (P_i, P_j) \in \mathcal{E}. \tag{2.2} \]

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

\[ O(P_i) = \{ O(P_j) \mid f(P_j) = f(P_i), (P_i, P_j) \in \mathcal{E} \}. \]

($\mathcal{B}, O, f$ ($O$ 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 $P_i$. $g: \mathcal{P} \rightarrow E \cup \{\infty\}$ satisfies

(a) $g(P_i) \leq f(P_i)$ for $P_i \in \mathcal{P}$,

(b) $g(P_i) = f(P_i)$ for $P_i \in \mathcal{P}$,

(c) $g(P_i) \leq g(P_j)$ for $(P_i, P_j) \in \mathcal{E}$.

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

(A) $g(P_i) = f(P_i)$ for $P_i \in \mathcal{P}$

(B) $\mathcal{P} \supseteq \mathcal{P}$

(C) $P_i \in \mathcal{P}$ implies $P_j \in \mathcal{P}$ for $(P_i, P_j) \in \mathcal{E}$.

Note that condition (A) is assumed for simplicity even if $P_i \in \mathcal{P}$ is concluded.
due to $O(P_1) \cap O(P_0) = \emptyset$, since in this case the value $g(P_1)$ is not relevant to the computation process.

At this point, let us derive $\mathcal{F}$ 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 $P_j$ (including the case of $P_j = P_0$) is decomposed into two partial problems by adding constraints

$$(2.3) \quad x_k \leq K \text{ and } x_k \geq K + 1$$

respectively to the original constraint of $P_j$, 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 $P_j$ (i.e., obtained from $P_j$ 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). $f(P_j)$ is the optimal value of integer programming problem $P_j$, $O(P_j)$ is the set of optimal solutions of $P_j$, and $g(P_j)$ is the optimal value of the LP problem corresponding to $P_j$. We say that $P_j \in \mathcal{F}$ if all integer variables are fixed by additional constraints of type (2.3), and $P_j \in \mathcal{G}$ if the LP optimal solution happens to be an integer solution or the LP problem turns out to be infeasible. Obviously these $\mathcal{F}$, $f$, $O$, $\mathcal{G}$, $g$, $\mathcal{F}$ satisfy the above conditions.

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

- (I) $u(P_j) \geq f(P_j)$ for $P_j \in \mathcal{F}$
- (II) $u(P_j) = f(P_j)$ for $P_j \in \mathcal{G}$.

$u(P_j) = \infty$ denotes that no feasible solution is obtained for $P_j$ or that the computation of $u(P_j)$ is not attempted. $u = \infty$ stands for that $u(P_j)$ 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 $\mathcal{F}$ satisfying the following conditions.

When all optimal solutions of $P_0$ are sought:

(I) $a)$ $P_j \nsubseteq P_i \neq P_j$ implies $f(P_j) < f(P_j)$ (including $\infty < \infty$).

(ii) $a)$ $P_j \nsubseteq P_i \neq P_j$ implies that, for each descendant $P_j$ of $P_i$, there

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.
exists a descendant \( P^-_i \) of \( P_i \) satisfying \( P^-_i \text{DP}_j \).

When a single optimal solution of \( P_0 \) is sought:

(i) \( P_i \text{DP}_j \land P_i \neq P_j \) implies \( f(P_i) \leq f(P_j) \) and that \( P_j \) is not an ancestor of \( P_i \).

(ii) \( P_i \text{DP}_j \land P_i \neq P_j \) implies, for each descendant \( P^-_j \) of \( P_j \), there exists a descendant \( P^-_i \) of \( P_i \) satisfying \( P^-_i \text{DP}_j^- \).

(iii) There exists no sequence of nodes \( P_i^1, P_i^2, \ldots, P_i^k \) (\( k \geq 2 \) and \( P_i^1, P_i^2, \ldots, P_i^k \) are distinct) generated during computation, such that \( P_i^1 \) is a proper descendant of \( P_i \) or \( P_i \text{DP}_i^j \land f(P_i^j) = f(P_i^s) \) for \( s = 1, 2, \ldots, k \), and \( P_i^k = P_i \).

(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 additional condition:

\( P_i \text{DP}_j \land P_i \neq P_j \) 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 \( s: \mathcal{F} \to \mathcal{E} \) such that \( s(\mathcal{A}) \in \mathcal{A} \) for \( \mathcal{A} \in \mathcal{I} \), where \( \mathcal{I} \) denotes the family of independent subsets of \( \mathcal{P} \). The following four search functions are typical.

- \( s \) is the heuristic search function based on a heuristic function \( h: \mathcal{P} \to \mathcal{E} \), denoted \( s = s_h \), if

  \[
  h(s(\mathcal{A})) = \min \{ h(P_i) \mid P_i \in \mathcal{A} \} \quad \text{for } \mathcal{A} \in \mathcal{I}.
  \]

  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 \( s = s_h \), is defined by

  \[
  h(s_h(\mathcal{A})) = \min \{ h(P_i) \mid P_i \in \overline{N}(\mathcal{A}) \}\]

  \[
  s_h(\mathcal{A}) \in \overline{N}(\mathcal{A})
  \]

  for \( \mathcal{A} \in \mathcal{I} \), where

  \[
  \overline{N}(\mathcal{A}) = \{ P_i \in \mathcal{A} \mid d(P_i) = \max \{ d(P_j) \mid P_j \in \mathcal{A} \} \}\]
Finally, the breadth-first search function based on $h$, denoted $s_h$, is defined by

$$h(s_h(A)) = \min \{ h(P_j) \mid P_j \in \mathcal{R}(A) \}$$

$$s_h(A) \in \mathcal{R}(A)$$

$$\mathcal{R}(A) = \{ P_i \in \mathcal{A} \mid d(P_i) = \min \{ d(P_j) \mid P_j \in \mathcal{A} \} \}$$

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 $P_i, P_j \in \mathcal{P}$. 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 $A_\alpha = (\mathcal{A}, O, f), (\mathcal{S}, g, u), D, s)$: all optimal solutions

In the following, $\mathcal{N} \subset \mathcal{P}$ denotes the set of nodes currently generated. A node in $\mathcal{N}$ is active if it is yet neither tested nor decomposed. $\mathcal{A}$ denotes the set of current active nodes. $\mathcal{O}$ stores the set of best feasible solutions currently available. $z$ is called the incumbent value and stores the current best upper bound of $f(P_i)$. Generally $z \leq f(\mathcal{O}) (= f(x)$ for $x \in \mathcal{O}$) holds ($z < f(\mathcal{O})$ is possible since $z$ is set in Step A2 even if optimal solutions are not known), but $z = f(\mathcal{O})$ 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 $P_i \in \mathcal{O}$.

A1 (Initialize): $\mathcal{A} = \{ P_0 \}, \mathcal{N} = \{ P_0 \}, z = \infty$ and $\mathcal{O} = \emptyset$.

A2 (Search): If $\mathcal{A} = \emptyset$, go to A9; else $P_i = s(\mathcal{A}), z = \min \{ z, u(P_i) \}$ and go to A3.

A3 (Test by $\mathcal{S}$): If $\mathcal{A} = \emptyset$, 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 $P_k(\#P_i) \in \mathcal{N}$ satisfying $P_k(DP_i)$, go to A8; else go to A6.

A6 (Decompose): Generate sons $P_1, P_2, \ldots, P_k$ of $P_i$. Return to A2

After letting $\mathcal{A} = \mathcal{A} \cup \{ P_1, P_2, \ldots, P_k \} - \{ P_i \}$ and $\mathcal{N} = \mathcal{N} \cup \{ P_1, P_2, \ldots, P_k \}$.
A7(Improve): Go to A8 after letting
\[
\mathcal{O}(P_i) \quad \text{if } f(P_i) < f(\mathcal{O})
\]
\[
\mathcal{O} \cup \mathcal{O}(P_i) \quad \text{if } f(P_i) = f(\mathcal{O})
\]
\[
\mathcal{O} \quad \text{otherwise.}
\]

A8(Terminate \(P_i\)): \(\mathcal{S} + \mathcal{S} - \{P_i\}\) and return to A2.

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

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

In this case, \(\mathcal{S}\) 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 \(P_i \in \mathcal{S}\)).

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

A2/Search): If \(\mathcal{S} = \emptyset\), go to A9; else \(P_i \leftarrow \mathcal{S}(\mathcal{S})\), \(z \leftarrow \min[z, u(P_i)]\), \(\mathcal{O} \leftarrow \mathcal{O}\) if \(u(P_i) \leq z\) else \(\{x\}\), where \(x\) is a feasible solution of \(P_i\) with \(f(x) = u(P_i)\).

Go to A3.

A3/Test by \(\mathcal{S}\): If \(P_i \in \mathcal{S}\), 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 \(f(x) = z = f(P_0)\). \(\square\)

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.

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.
3. Power of Lower Bounding Functions under Heuristic Search

Consider two branch-and-bound algorithms $A(g'_1) = ((\mathcal{E}, O, f), (\mathcal{S}_1, g'_1, u), D, s'_h)$ and $A(g'_2) = ((\mathcal{E}, O, f), (\mathcal{S}_2, g'_2, u), D, s'_h)$. $A(g'_1)$ and $A(g'_2)$ differ only in $\mathcal{S}$ and $g$, and both algorithms use the same heuristic search function $s'_h$ which is not dependent on $(\mathcal{S}, g)$. We say that $(\mathcal{S}_1, g'_1)$ is tighter than $(\mathcal{S}_2, g'_2)$ and denote by $g'_1 \preceq g'_2$, if $\mathcal{S}_1 \supset \mathcal{S}_2$ and $g'_1(p_i) \geq g'_2(p_i)$ for $p_i \in \mathcal{S}$. 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 $A(g'_1) = ((\mathcal{E}, O, f), (\mathcal{S}_1, g'_1, u), D, s'_h)$ and $A(g'_2) = ((\mathcal{E}, O, f), (\mathcal{S}_2, g'_2, u), D, s'_h)$ 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 $(\mathcal{S}, f), (\mathcal{S}_1, g'_1), (\mathcal{S}_2, g'_2), h$ and $D$. Nodes in $\mathcal{S}_1 = \mathcal{S}_2$ (in this case) are indicated by double circles, and dominance relation $P_i \Downarrow P_j$ is denoted by $P_i \rightarrow P_j$. $u = \infty$ is assumed in this example (note however that $u(p_i) = f(p_i)$ are assumed for $P_i \in \mathcal{S}$ 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 $s'_h$.  

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.
Counterexample to the conjecture $g_1 \geq g_2 \Rightarrow T(g_1) \leq T(g_2) \land B(g_1) \leq B(g_2)$ under heuristic search used in the proof of Theorem 3.1. (Nodes in $\mathcal{C}$ are denoted by double circles. Dominance relation $P_i \supset P_j$ is indicated by $P_i \supset P_j$. Some relations derivable from others by conditions (i)$\land$(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 \( g_1 \leq g_2 \Rightarrow T(g_1) \leq T(g_2) \wedge B(g_1) \leq B(g_2) \) 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 \( B_a(g_1) = B_a(g_2) \) holds, i.e., the \( B_a \)-count is independent of \((G, g)\), and \( g_1 \geq g_2 \) implies \( B_a(g_1) \leq B_a(g_2) \) (proper inequality is possible only if \( g_1 \neq g_2 \)).

Proof. Let \( \mathcal{S} = P_{j_1} P_{j_2} \ldots P_{j_t} \) be the sequence of nodes selected in \( A_2 \) of \( A(g_1) \) (and \( A(g_2) \)) with \( A_3 \) (test by \( \mathcal{S} \)) replaced by: If \( P_{j_1} \in \mathcal{S} \) go to \( A_7 \); else go to \( A_4 \), and with \( A_4 \) (lower bound test) and \( A_5 \) (dominance test) suppressed. (Thus \( \mathcal{S} \) is the sequence of all nodes in \( \mathcal{S} \) arranged in the order selected by search function \( s_h \). It is independent of \((G, g)\) and \( D \).) The sequences of nodes actually selected in \( A(g_1) \) and \( A(g_2) \) are subsequences of \( \mathcal{S} \) (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 \( A_2 \) of \( A(g_1) \) and satisfies \( P_{j_a} \in \mathcal{S}_1 \), \( f(P_{j_a}) = f(P_0) \). 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) of \( T \). 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 \( p_{j_b} \) \((1 \leq b \leq a)\) may be terminated by \( \mathcal{F}_2 \) (in \( A(g_2) \)), such \( p_{j_b} \) is also terminated by \( \mathcal{F}_1 \) (in \( A(g_1) \)) since \( \mathcal{F}_1 \supset \mathcal{F}_2 \). Consequently \( p_{j_b} \) \((1 \leq b \leq a)\) is terminated in \( A(g_2) \) only if it is terminated in \( A(g_1) \). This proves \( B_a(g_1) \leq B_a(g_2) \). The result concerning proper inequality is also obvious. \( \square \)

As we have seen in the above theorems, \( T \) and \( B \) are not monotonically related to \((\mathcal{F}, g)\) in most cases. In order to guarantee the monotonic 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 \( g_1 \geq g_2 \) implies \( T(g_1) \leq T(g_2) \) and \( B(g_1) \leq B(g_2) \).

**Proof.** We consider the case of all optimal solutions only. Let \( \mathcal{F}^1 = p_{i_1} p_{i_2} \ldots p_{i_a} \) and \( \mathcal{F}^2 = p_{j_1} p_{j_2} \ldots p_{j_b} \) be the sequences of nodes selected in \( A(g_1) \) and \( A(g_2) \) respectively. These are subsequences of \( \mathcal{F} \) introduced in the proof of Theorem 3.2. We first show by induction that \( \mathcal{F}^1 \) is a subsequence of \( \mathcal{F}^2 \). (This immediately implies \( T(g_1) \leq T(g_2) \), and is a key step to prove \( B(g_1) \leq B(g_2) \).)

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

\[
\mathcal{F}^1_a = p_{i_1} p_{i_2} \ldots p_{i_a}, 1 \leq a \leq s,
\]

there is the unique initial portion of \( \mathcal{F}^2 \),

\[
\mathcal{F}^2 = p_{j_1} p_{j_2} \ldots p_{j_b}
\]

such that \( p_{i_a} = p_{j_b}, \mathcal{A}_1(p_{i_a}) \subset \mathcal{A}_2(p_{j_b}), z_1(p_{i_a}) = z_2(p_{j_b}) \) and \( \mathcal{F}^1_a \) is a subsequence of \( \mathcal{F}^2 \), where \( \mathcal{A}_k(p) \) 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 \( \mathcal{A}_1(p_{i_a}) \subset \mathcal{A}_2(p_{j_b}) \) since \( \mathcal{A}_1(p_{i_a}) = \{p_{i_1}, p_{i_2}, \ldots, p_{i_{a-1}}\} \cup \mathcal{A}_2(p_{j_b}) \) and \( \mathcal{A}_2(p_{j_b}) = \{p_{j_1}, p_{j_2}, \ldots, p_{j_{b-1}}\} \cup \mathcal{A}_2(p_{j_b}) \).

This induction hypothesis is trivially true for \( a = 1 \), since \( \mathcal{F}^1_1 \) satisfies the above conditions. In order to prove the general case, we first show that
$P_{ia}$ is terminated in $A(g_1)$ whenever $P_{jb}$ is terminated in $A(g_2)$. The following two cases are considered corresponding to how $P_{ja}$ is terminated.

(a) $P_{ja}$ is terminated by $G_2$ or by lower bound test: If $P_{jb} \in G_2$, then $P_{ia}(=P_{ia}) \in G_1$ by $G_1 \supset G_2$. If $g_2(P_{ja}) > z_2(P_{ja})$, then $g_1(P_{ia}) \geq g_2(P_{ja}) > z_2(P_{ia}) = z_1(P_{ia})$ (by induction hypothesis). In either case $P_{ia}$ is terminated in $A(g_1)$.

(b) $P_{ia}$ is terminated by dominance test: Then some $P_{ja} \in N_2(P_{jb})$ satisfies $P_{ja} \preceq P_{ja}$, where $N_k(P)$ denotes $N$ when $P$ is selected in $A(g_k)$. Assume $P_{ja} \notin N_1(P_{ia})$ since otherwise $P_{ia}$ is also terminated. This means that a proper ancestor $P_{iq}$ of $P_{ja}$ 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.](image)

If $P_{ia}$ has been terminated by lower bound test, or by $G_1$, it follows that

$$z_1(P_{ia}) \leq z_1(P_{iq}) \quad \text{(since $P_{ia}$ is selected after $P_{iq}$ in $A(g_1)$)}$$

$$\leq g_1(P_{ia}) \leq g_1(P_{ja}) \quad \text{(by condition (c) of $g$)}$$

$$< g_1(P_{ia}) \quad \text{(since $D$ is consistent with $g$)}.$$ 

Hence $P_{ia}$ is terminated in $A(g_1)$ by lower bound test. Thus assume that $P_{iq}$ has been terminated in $A(g_1)$ by dominance test, i.e., $P_{ia} \preceq P_{ja}$ for some $P_{ja} \in N_1(P_{ia})$.

Then $P_{ia} \in N_2(P_{ia})$ follows since $N_2(P_{ja}) \supset N_1(P_{ja})$ by induction hypothesis. Thus $P_{ia}$ must have been terminated in $A(g_2)$ by dominance test, a contradiction. This proves that $P_{ia}$ is terminated in $A(g_1)$.

Now the induction can proceed one step unless $G_a = G^1$ (in this case the proof is done).
\[
\mathcal{G}^1_{a+1} = P_{i_1} P_{i_2} \ldots P_{i_a} P_{i_{a+1}} \\
\mathcal{G}^2_{b+w} = P_{j_1} P_{j_2} \ldots P_{j_b} \ldots P_{j_{b+w}},
\]

where \(P_{i_{a+1}} = P_{j_{b+w}}\). This \(\mathcal{G}^2_{b+w}\) exists since \(\mathcal{G}^2_{j_b} \supset \mathcal{G}^1_{j_b}\) (induction hypothesis) and \(P_{j_{b+w}}\) is eventually tested in \(A(g_2)\). Obviously \(\mathcal{G}^1_{a+1}\) is a subsequence of \(\mathcal{G}^2_{b+w}\). We then prove \(\mathcal{G}^2_{j_{b+1}} \subset \mathcal{G}^1_{j_{b+1}}\). Note that \(\mathcal{G}^1_{j_{b+1}} \subset \mathcal{G}^2_{j_{b+1}}\) (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,

\[
h(P_{i_{a+1}}) = \min \{h(p) \mid p \in \mathcal{G}^2_{j_{b+1}}\}
\]

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

\[
h(P_{j_{b+1}}) < h(P_{i_{a+1}}) = h(P_{i_{a+1}}),
\]

i.e., \(P_{j_{b+1}} \notin \mathcal{G}^2_{j_{b+1}}\) for \(v = 1, 2, \ldots, w-1\). This proves \(\mathcal{G}^2_{j_{b+1}} \subset \mathcal{G}^1_{j_{b+1}}\).

To prove \(z_1(P_{j_{b+1}}) = z_2(P_{j_{b+1}})\), note that a proper ancestor \(P_{i_e}\) (\(1 \leq e \leq a\)) of each \(P_{j_{b+1}}\) \((1 \leq v < w)\) (we assume \(w > 1\) since otherwise the proof is trivial) has been terminated in \(A(g_1)\) by \(\mathcal{G}\) 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 \(\mathcal{G}^1_{j_{b+1}} \subset \mathcal{G}^2_{j_{b+1}}\) follows from the induction hypothesis.) Thus

\[
u(P_{j_{b+1}}) \geq g_1(P_{j_{b+1}}) \geq g_1(P_{i_e})
\]

and \(g_1(P_{i_e}) \geq z_1(P_{i_e}) \geq z_1(P_{i_{a+1}})\) if \(P_{i_e}\) is terminated in \(A(g_1)\) by \(\mathcal{G}\) or by lower bound test. This shows \(u(P_{j_{b+1}}) \geq z_1(P_{i_{a+1}})\), i.e., \(z_1\) would not have been improved even if \(P_{j_{b+1}}\) is tested in \(A(g_1)\), implying \(z_1(P_{j_{b+1}}) \leq z_2(P_{j_{b+1}})\).

On the other hand \(z_1(P_{i_{a+1}}) \geq z_2(P_{j_{b+1}})\) since \(\mathcal{G}^2_{a+1}\) is a subsequence of \(\mathcal{G}^2_{b+w}\), and hence \(z_1(P_{i_{a+1}}) = z_2(P_{j_{b+1}})\).

Consequently \(\mathcal{G}^1\) is a subsequence of \(\mathcal{G}^2\), and \(T_a(g_1) \leq T_a(g_2)\) is an immediate consequence of it. To treat the \(B\)-count, let
\( \mathcal{F}_1^{\text{opt}} = \{ p_i \in \mathcal{F} | p_i \in \mathcal{F}_1', f(p_i) = f(p_0), \text{ and no proper ancestor of } p_i \text{ belongs to } \mathcal{F}_1' \} \).

\( \mathcal{F}_2^{\text{opt}} \) is similarly defined. Obviously \( B_a(g_k) \) is equal to the number of nodes decomposed in \( A(g_k) \) before all nodes in \( \mathcal{F}_1^k \) are tested. By \( \mathcal{F}_1 \supset \mathcal{F}_2 \), it follows that, for any \( p_j \in \mathcal{F}_2^{\text{opt}} \), there exists \( p_i \in \mathcal{F}_1^{\text{opt}} \) such that \( p_i \) is an ancestor (including the case of \( p_i = p_j \)) of \( p_j \). This property and that \( \mathcal{F}_1^{\text{opt}} \) is a subsequence of \( \mathcal{F}_2^{\text{opt}} \) 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 differences 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 \( A(g_1) = (\mathcal{F}, \mathcal{G}, f, (\mathcal{G}_1, g_1', u), D, s_1) \) and \( A(g_2) = (\mathcal{F}, \mathcal{G}, f, (\mathcal{G}_2, g_2', u), D, s_2) \) using best-bound search functions \( g_1' \) and \( g_2' \) that are usually different. Classes of branch-and-bound algorithms 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 monotonic 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 \( \varepsilon \).

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 \( P_i \) \( \langle P_k DP_j \rangle \)) 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

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.
smaller depth results in an $g$ compatible with $D$.) Usually, however, we will not assume any particular tie breaking rule unless otherwise stated.

**Theorem 4.1.** Let $A(g_1) = ((\mathcal{B}, O, f), (\mathcal{C}_1, g_1, u), D, s_{g_1})$ and $A(g_2) = ((\mathcal{B}, O, f), (\mathcal{C}_2, g_2, u), D, s_{g_2})$ 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 $B(g_1) \leq B(g_2)$.

![Diagram](image)

**Fig. 4** Counterexample to the conjecture $g_1 \geq g_2$ $\Rightarrow$ $T(g_1) \leq T(g_2) \land B(g_1) \leq B(g_2)$ 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 $A(g) = ((\mathcal{B}, O, f), (\mathcal{C}, g, u), D, s_{g})$ be a branch-and-bound algorithm using best-bound search, where $D$ is consistent with $g$.

Define

\[ \mathcal{F} = \{ P_1 \in \mathcal{P} \mid g(P_1) \leq f(P_0) \} \]  
\[ \mathcal{F}_D = \{ P_1 \in \mathcal{P} \mid \text{no } P_j \neq P_1 \in \mathcal{P} \text{ satisfies } P_jDP_1 \}. \]

Then $T_a(g) = |\mathcal{F} \cap \mathcal{F}_D - \mathcal{F}|$.
Proof. It is known (e.g., lemma 6.1 of [11] and Lemma 3 of [8]) that
\[ g(P_j) \leq g(P_{j2}) \] holds if \( P_{j1} \) is selected after \( P_{j2} \) in \( A(g) \), and that \( z \) is set
to \( f(P_0) \) before any \( P_i \) with \( g(P_j) > f(P_0) \) 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
there exists \( P_j \in \mathcal{N}(P_i) \); i.e., \( P_i \notin \mathcal{F} \). 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., \( P_{j2} \in \mathcal{N}(P_k) \) (note
that \( P_k \) is not terminated by \( \mathcal{G} \) or by lower bound test since \( g(P_k) \leq g(P_j) <
g(P_{i1}) \) implies \( P_k \in \mathcal{F} - \mathcal{G} \)). By condition (ii) of \( D \), there exists a
descendant \( P_m \) of \( P_k \) such that \( P_{j2} \in \mathcal{N}(P_m) \) (see Fig. 5). If \( P_m \) is eventually generated
in \( A(g) \), it is generated before \( P_i \) since \( g(P_m) < g(P_j) < g(P_{i1}) \). In
addition, \( P_{j2} \) 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 \( \mathcal{F} \) 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 \( P_i \notin \mathcal{F} \). Therefore we have \( T_a(g) = |\mathcal{F} \cap \mathcal{F} - \mathcal{G}| \).

\[ \text{Fig. 5 Relative positions of } P_i, P_j, P_k, P_{j2}, P_m \text{ used in the}
\text{proof of Theorem 4.2.} \]

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 con-
sistent with \( g_1 \). Then \( g_1 \geq g_2 \) implies \( T_a(g_1) \leq T_a(g_2) \) and \( B_a(g_1) \leq B_a(g_2) +
|\mathcal{H}^* \cap \mathcal{F} - \mathcal{G}_1| \), where

\[ \mathcal{H}^* = \{ P_i \in \mathcal{F} | g_1(P_i) = g_2(P_i) = f(P_0) \} \]

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.
Proof. \( T_a(g_1) \leq T_a(g_2) \): By Lemma 4.2, we have

\[
(4.4) \quad T_a(g_1) = |\mathcal{F}_1 \cap \mathcal{P}_D - \mathcal{G}_1|, \quad T_a(g_2) \geq |\mathcal{F}_2 \cap \mathcal{P}_D - \mathcal{G}_2|
\]

where \( \mathcal{F}_k \) 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 \( g_1 \geq g_2 \) implies \( \mathcal{F}_1 \subset \mathcal{F}_2 \) and \( \mathcal{G}_1 \supset \mathcal{G}_2 \), \( T_a(g_1) \leq T_a(g_2) \) immediately follows.

\[
B_a(g_1) \leq B_a(g_2) + |\mathcal{X}^* \cap \mathcal{P}_D - \mathcal{G}_1|;
\]

Let

\[
(4.5) \quad \mathcal{X}_1 = \{ p \in \mathcal{P} | g_1(p) < f(P_0), g_2(p) < f(P_0) \}
\]

\[
(4.6) \quad \mathcal{X}_1' = \{ p \in \mathcal{P} | g_1(p) = f(P_0), g_2(p) < f(P_0) \}.
\]

Under best-bound search, nodes in \( \mathcal{X}_1 \cup \mathcal{X}_1' \) are tested in \( A(g_2) \) before nodes in \( \mathcal{X}^* \). Furthermore, all nodes \( p_j \) satisfying \( p_j \in \mathcal{G}_2 \land f(p_j) = f(P_0) \) belong to \( \mathcal{X}^* \). Thus, denoting the set of nodes decomposed in \( A(g_2) \) by \( \mathcal{I}_2 \), we have

\[
|\mathcal{X}_1 \cup \mathcal{X}_1'| \cap \mathcal{I}_2 | \leq B_a(g_2).
\]

Next note that nodes in \( \mathcal{X}_1 \) are tested in \( A(g_1) \) before nodes in \( \mathcal{X}_1' \cup \mathcal{X}^* \), and nodes \( p_j \) satisfying \( p_j \in \mathcal{G}_1 \land f(p_j) = f(P_0) \) belong to \( \mathcal{X}_1' \cup \mathcal{X}^* \). Thus, denoting the set of nodes decomposed in \( A(g_1) \) by \( \mathcal{I}_1 \), we have

\[
B_a(g_1) \leq |\mathcal{X}_1 \cap \mathcal{I}_1| + |(\mathcal{X}_1' \cup \mathcal{X}^*) \cap \mathcal{I}_1|
\]

\[
= |\mathcal{X}_1 \cap \mathcal{I}_1| + |\mathcal{X}_1' \cap \mathcal{I}_1| + |\mathcal{X}^* \cap \mathcal{I}_1|
\]

\[
\leq |\mathcal{X}_1 \cup \mathcal{X}_1'| \cap \mathcal{I}_1 | + |\mathcal{X}^* \cap \mathcal{P}_D - \mathcal{G}_1|
\]

(by \( \mathcal{I}_2 \supset \mathcal{I}_1 \) and by \( \mathcal{X}^* \cap \mathcal{I}_1 = \mathcal{X}^* \cap \mathcal{P}_D - \mathcal{G}_1 \) derived from Lemma 4.2)

\[
\leq B_a(g_2) + |\mathcal{X}^* \cap \mathcal{P}_D - \mathcal{G}_1|.
\]

The term \( |\mathcal{X}^* \cap \mathcal{P}_D - \mathcal{G}_1| \) is necessary in the above theorem because a best-bound search function \( s \) (without a tie breaking rule) does not decide which nodes in \( \mathcal{X}^* \) 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.

![Diagram of branch-and-bound algorithm](image)

(a) \((j, f), g_1 \) and \( g_2 \) \((u=\infty \) is assumed\)

(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.)

(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 $s_1$ is compatible with $D$, then
\[ g_1 \geq g_2 \Rightarrow T_s(g_1) \leq T_s(g_2) + |\mathcal{F}_D^* \cap \mathcal{F}_1|. \]

(ii) If $s_1$ is compatible with $D$ and $g_1(P_1) > g_2(P_1)$ for $P_1 \in \mathcal{F} - \mathcal{F}_1$, then $g_1 \geq g_2$ implies $T_s(g_1) \leq T_s(g_2)$.

(iii) If $g_1(P_j) > g_1(P_j)$ for every $(P_j, P_j) \in \mathcal{E}$ with $P_j \in \mathcal{F} - \mathcal{F}_1$, and if $P_k \in \mathcal{F}_1$ has higher priority than $P_k \notin \mathcal{F}_1$ satisfying $g_1(P_k) = g_1(P_j)$ in selecting an active node in $A_2$ 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) 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 $\mathcal{F}$ 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 $s_1$ is compatible with $D$, then
\[ g_1 \geq g_2 \Rightarrow B_s(g_1) \leq B_s(g_2) + |\mathcal{F}_D^* \cap \mathcal{F}_1|. \]

(ii) If $u=\infty$, $s_1$ is compatible with $D$ and $g_1(P_1) > g_2(P_1)$ for $P_1 \in \mathcal{F} - \mathcal{F}_1$, then $g_1 \geq g_2$ implies $B_s(g_1) \leq B_s(g_2)$.

(iii) If $u=\infty$, $g_1(P_j) > g_1(P_j)$ for every $(P_j, P_j) \in \mathcal{E}$ with $P_j \in \mathcal{F} - \mathcal{F}_1$, and if $P_k \in \mathcal{F}_1$ has higher priority than $P_k \notin \mathcal{F}_1$ satisfying $g_1(P_k) = g_1(P_j)$ in selecting an active node in $A_2$ 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.

\[ \text{Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.} \]
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.)

<table>
<thead>
<tr>
<th>Search Strategies</th>
<th>D</th>
<th>General Properties</th>
<th>Theorems</th>
<th>Consistent with $g$ Properties</th>
<th>Theorems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heuristic</td>
<td></td>
<td>3.1</td>
<td></td>
<td>$T(g_1) \leq T(g_2)$</td>
<td>3.3</td>
</tr>
<tr>
<td>(General)</td>
<td></td>
<td>3.1</td>
<td></td>
<td>$B(g_1) \leq B(g_2)$</td>
<td>3.3</td>
</tr>
<tr>
<td>Heuristic</td>
<td></td>
<td>3.1</td>
<td></td>
<td>$T(g_1) \leq T(g_2)$</td>
<td>3.3</td>
</tr>
<tr>
<td>(Nonmisleading)</td>
<td></td>
<td>3.2</td>
<td></td>
<td>$B(g_1) \leq B(g_2)$</td>
<td>3.3</td>
</tr>
<tr>
<td>(Nonmisleading)</td>
<td></td>
<td>3.2</td>
<td></td>
<td>$T(g_1) \leq T(g_2)$</td>
<td>3.3</td>
</tr>
<tr>
<td>Depth-First</td>
<td></td>
<td>3.1</td>
<td></td>
<td>$T(g_1) \leq T(g_2)$</td>
<td>3.3</td>
</tr>
<tr>
<td>Breadth-First</td>
<td></td>
<td>3.1</td>
<td></td>
<td>$B(g_1) \leq B(g_2)$</td>
<td>3.3</td>
</tr>
<tr>
<td>Best-Bound(a)</td>
<td></td>
<td>4.1</td>
<td></td>
<td>$T_a(g_1) \leq T_a(g_2)$</td>
<td>4.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.1</td>
<td></td>
<td>$B_a(g_1) \leq B_a(g_2) +</td>
<td>4.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$</td>
<td>E^* \cap D - F_1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$\geq \mu$</td>
<td></td>
</tr>
</tbody>
</table>

(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 $A(u_1) = (D, o, f, (g, u_1), D, s)$ and $A(u_2) = (D, o, f, (g, u_2), D, s)$ 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 $u_1(p) \leq u_2(p)$ holds for any $p \in \mathcal{G}$. 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 $v = u(P_0)$, and did not assume the general class of heuristic search).

Theorem 5.1. Let $A(u_1) = ((\mathcal{R}, \emptyset, f), (\mathcal{G}, g, u_1), D, s_h)$ and $A(u_2) = ((\mathcal{R}, \emptyset, f), (\mathcal{G}, O, u_2), D, s_h)$ 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.

![Diagram](image)

Fig. 7 Counterexample to the conjecture $u_1 \leq u_2 \Rightarrow T(u_1) \leq T(u_2) \land B(u_1) \leq B(u_2)$ 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 \( B(u_1) \leq B(u_2) \).

\[
\begin{align*}
T(u_1) &= B(u_1) = 7 > T(u_2) = B(u_2) = 5
\end{align*}
\]

**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 \( A(u_1) = ((B, O, f), (S, g, u_1), D, s_g) \) and \( A(u_2) = ((B, O, f), (S, g, u_2), D, s_g) \) be branch-and-bound algorithms using best-bound search. Then \( u_1 \leq u_2 \) implies \( T_a(u_1) = T_a(u_2), B_a(u_1) = B_a(u_2), T_s(u_1) \leq T_s(u_2) \) 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 \( r_i \) is decomposed if and only if \( r_i \in S \cap D - S \) (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 \( \mathcal{F}_1 = r'_1 \cdots r'_s \) and \( \mathcal{F}_2 = r''_1 \cdots r''_t \) be the sequences of nodes selected in \( A(u_1) \) and \( A(u_2) \), respectively. It is shown by induction that \( \mathcal{F}_1 \) is a subsequence of \( \mathcal{F}_2 \).
To prove this, assume that for $T_a = p_{i_1} \ldots p_{i_s}$, $1 \leq a \leq s$, there exists $T_b = p_{j_1} \ldots p_{j_s}$ satisfying that $p_{i_a} = p_{j_b}$, $\mathcal{S}_1(p_{i_a}) \subseteq \mathcal{S}_2(p_{j_b})$, $z_1(p_{i_a}) \leq z_2(p_{j_b})$ and $T_a$ is a subsequence of $T_b$. For $a = 1$, this induction hypothesis is trivially true since $T_a$ 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 $\mathcal{S}$: Then $p_{i_a} (= p_{j_b})$ is also terminated in $A(u_1)$ by $\mathcal{S}$.

(b) $p_{j_b}$ is terminated in $A(u_2)$ by lower bound test, i.e., $g(p_{j_b}) \geq z_2(p_{j_b})$: Then $z_2(p_{j_b}) = f(p_0)$ holds as a characteristic of best-bound search, and there exists $p_{j_p}$ (1 \leq p \leq b) satisfying $u_2(p_{j_p}) = f(p_0)$ in $\mathcal{S}_2$. Assume that a proper ancestor $p_{i_q}$ of $p_{j_p}$ has been terminated in $A(u_1)$, since otherwise $z_1(p_{i_a}) \leq u_1(p_{j_b}) \leq u_2(p_{j_b}) = f(p_0) \leq g(p_{j_p}) = g(p_{i_b})$ and hence $p_{i_a}$ is terminated in $A(u_1)$ by lower bound test. If $p_{j_p}$ has been terminated in $A(u_1)$ by lower bound test, it follows

\[ g(p_{i_a}) \geq g(p_{i_q}) \quad \text{(since } p_{i_a} \text{ is selected after } p_{i_q}) \]
\[ \geq z_1(p_{i_a}) = f(p_0) \]
\[ \geq z_1(p_{i_q}) \quad \text{(since } p_{i_q} \text{ is selected after } p_{i_q}) \]

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

(c) $p_{j_b}$ is terminated in $A(u_2)$ by dominance test $d_{i_x} p_{i_y}$ for $p_{i_x} \in \mathcal{N}(p_{i_y})$: 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.
\[ g(P_i^a) \geq g(P_i^a_{i_d}) \text{ (since } P_i^a \text{ is selected after } P_i^a_{i_d} \) \\
(5.2) \quad \geq z_1(P_i^a_{i_d}) \geq z_1(P_i^a) \]

and \( P_i^a \) is terminated in \( A(u_1) \) by lower bound test.

Now assume that \( \mathcal{G}_a^1 \neq \mathcal{G}_a^1 \) (since otherwise the proof is done) and define the following two sequences

\[
\mathcal{G}_a^{i+1} = P_i^a P_1^a \ldots P_i^a P_i^a_{i+1} \\
\mathcal{G}_b^{i+2} = P_j^a P_j^a \ldots P_j^b_{j+b} \ldots P_j^b_{j+b+2},
\]

such that \( P_i^a = P_j^a \). \( \mathcal{G}_a^{i+1} \) is a subsequence of \( \mathcal{G}_b^{i+2} \). \( \mathcal{G}_a^{i+1} \) in \( \mathcal{G}_a^{i+2} \) can be proved in a manner similar to the proof of Theorem 3.3. To prove \( z_1(P_i^a_{i+1}) \leq z_2(P_j^a_{j+b+2} \) ), assume contrary, i.e., \( z_1(P_i^a_{i+1}) > z_2(P_j^a_{j+b}) \).

Then there exists \( P_{j+b+1} \) \( (1 \leq k \leq \omega) \) such that \( z_2(P_i^a_{i+1}) = u_2(P_j^a_{j+b+1}) < z_1(P_i^a_{i+1}) \).

Since \( P_{j+b+1} \) is not generated in \( A(u_1) \), there is \( P_i^a = P_i^a \) in \( \mathcal{G}_a \) and \( \mathcal{G}_b \) respectively such that (1) \( P_i^a \) is a proper ancestor of \( P_j^a \) and (2) \( P_i^a \) is terminated in \( A(u_1) \) but \( P_j^a \) is not terminated in \( A(u_2) \). (2) is possible only if \( P_i^a \) is terminated by lower bound test, as obvious from the above proof.

Therefore

\[ z_1(P_i^a_{i+1}) \leq g(P_i^a) = g(P_j^a) < z_2(P_j^a), \]

and \( P_j^a \) (a descendant of \( P_i^a \)) satisfies

\[ (z_1(P_i^a_{i+1}) \leq z_1(P_j^a) \leq g(P_j^a) \leq g(P_j^a_{j+b}) \leq u_2(P_j^a_{j+b+2}), \]

which is a contradiction. This proves \( z_1(P_i^a_{i+1}) \leq z_2(P_j^a_{j+b+2}). \)

Consequently \( \mathcal{G}_a \) is a subsequence of \( \mathcal{G}_b \), 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 \( B_s(u_1) \leq B_s(u_2) \) in Theorem 3.3 (use also the property \( z_1(P_i^a_{i+1}) \leq z_2(P_j^a_{j+b}) \) for \( P_i^a = P_j^a \)). \( \square \)

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 \( P_i^a \), \( P_j^a \in \mathcal{A}_1 \), and \( P_i^a \) is selected before \( P_j^a \) in \( A(u_1) \), then \( P_i^a \) is selected before
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.

![Diagram](image)

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 $\tilde{I} = P_{i_1}P_{i_2} \ldots P_{i_p}$ 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 $P_{i_p} \in \mathcal{S} \land f(P_{i_p}) = f(P_0)$. We show below that
A(u_2) also selects exactly the same sequence of nodes and that P_{i_1} (1 \leq i \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 \mathcal{C}^+ and f(P_{i_p}) = f(P_0), since \mathcal{C} 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 \mathcal{T} 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 \mathcal{T}_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 \mathcal{C}_1(P_{i_1}) = \mathcal{C}_2(P_{i_1}) 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}) = 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 \mathcal{C}_1(P_{i_{a-1}}) = \{P_{i_1}, P_{i_2}, \ldots, P_{i_{a-1}}\} \cup \mathcal{C}_2(P_{i_{a-1}}) = \{P_{i_1}, P_{i_2}, \ldots, P_{i_{a-1}}\}. 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 \mathcal{T}_a and \mathcal{T}_b 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_{i_a} is terminated in A(u_2) by \mathcal{C}: Then P_{i_a} (= P_{j_b}) is also terminated by \mathcal{C}.

(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:

\[ g(P^a_j) > g(P^c_j) \quad (\text{since } D \text{ is consistent with } g) \]

(5.4) \[ b_i \leq g(P^d_i) > z^s_j(P^e_i) \geq z^s_i(P^f_i). \]

(c) \( P^g_j \) is terminated in \( A(u^h_2) \) by dominance test. The proof of 5.2 ((c)-part) can be used after changing (5.2) to:

\[ g(P^a_j) > g(P^a_j) \]

(5.3) \[ g(P^a_j) > g(P^a_j) \]

The construction of \( \mathcal{F}^1_{a+1} \) and \( \mathcal{F}^2_{b+h} \) to complete the induction step can also be done in the same manner as Theorem 5.2. Thus we have \( T^a(u^1_1) \leq T^a(u^2_1) \) and \( B^a(u^1_1) \leq B^a(u^2_1) \).

The results of this section are summarized in Table 2.

Table 2. Computational efficiency of \( A(u^1_1) \) and \( A(u^2_2) \) with \( u^1_1 \leq u^2_2 \).

(Entries \( \sim \) denote that \( T(u^1_1) \leq T(u^2_2) \) or \( B(u^1_1) \leq B(u^2_2) \) does not necessarily hold.)

<table>
<thead>
<tr>
<th>Search Strategies</th>
<th>( D )</th>
<th>General</th>
<th>Consistent with ( g )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Theorems</td>
<td>Theorems</td>
</tr>
<tr>
<td>Heuristic</td>
<td>—</td>
<td>5.1</td>
<td>( T(u^1_1) \leq T(u^2_1) )</td>
</tr>
<tr>
<td>(General)</td>
<td>—</td>
<td>5.1</td>
<td>( B(u^1_1) \leq B(u^2_1) )</td>
</tr>
<tr>
<td>Heuristic</td>
<td>( T^a(u^1_1) = T^a(u^2_1) )</td>
<td>5.3</td>
<td>( T^a(u^1_1) = T^a(u^2_1) )</td>
</tr>
<tr>
<td>(Nonmisleading)</td>
<td>—</td>
<td>5.3</td>
<td>( T^a(u^1_1) \leq T^a(u^2_1) )</td>
</tr>
<tr>
<td></td>
<td>( B^a(u^1_1) = B^a(u^2_1) )</td>
<td>5.3</td>
<td>( B^a(u^1_1) = B^a(u^2_1) )</td>
</tr>
<tr>
<td></td>
<td>( B^s(u^1_1) \leq B^s(u^2_1) )</td>
<td>5.3</td>
<td>( B^s(u^1_1) \leq B^s(u^2_1) )</td>
</tr>
<tr>
<td>Depth-First</td>
<td>—</td>
<td>5.1</td>
<td>( T(u^1_1) \leq T(u^2_1) )</td>
</tr>
<tr>
<td>Breadth-First</td>
<td>—</td>
<td>5.1</td>
<td>( B(u^1_1) \leq B(u^2_1) )</td>
</tr>
<tr>
<td>Best-Bound</td>
<td>( T^a(u^1_1) = T^a(u^2_1) )</td>
<td>5.2</td>
<td>( T^a(u^1_1) = T^a(u^2_1) )</td>
</tr>
<tr>
<td></td>
<td>( T^a(u^1_1) \leq T^a(u^2_1) )</td>
<td>5.2</td>
<td>( T^a(u^1_1) \leq T^a(u^2_1) )</td>
</tr>
<tr>
<td></td>
<td>( B^a(u^1_1) = B^a(u^2_1) )</td>
<td>5.2</td>
<td>( B^a(u^1_1) = B^a(u^2_1) )</td>
</tr>
<tr>
<td></td>
<td>( B^s(u^1_1) \leq B^s(u^2_1) )</td>
<td>5.2</td>
<td>( B^s(u^1_1) \leq B^s(u^2_1) )</td>
</tr>
</tbody>
</table>
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 $\mathcal{A}$ 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


Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.