Branch-and-Bound Search

Vipin Kumar

AI TR 85-11

This work was supported in part by Army Research Office grant #DAAG29-84-K-0060 to the Artificial Intelligence Laboratory at the University of Texas at Austin.

August 1985
Branch-and-Bound Search

INTRODUCTION

Branch-and-bound (B&B) is a problem solving technique which has been usefully employed in various problems encountered in Operations Research and Combinatorial Mathematics. Many heuristic search procedures used in Artificial Intelligence can be viewed as B&B procedures. The class of problems solved by branch-and-bound procedures can be abstractly stated as:

Given a (possibly infinite) discrete set \( X \) and a real-valued cost function \( f \) whose domain is \( X \), find an optimal element \( x^* \) of \( X \) such that \( f(x^*) = \min \{ f(x) \mid x \in X \} \).\(^1\)

Unless there is enough problem-specific knowledge available to obtain an optimum element of the set in some straightforward manner, the only available alternative may be to implicitly enumerate the set \( X \). For practical problems, the size of the set \( X \) is quite large, which makes exhaustive enumeration prohibitively time consuming.

Using the available knowledge about the problem, branch-and-bound procedures decompose the original set into sets of smaller and smaller sizes. The decomposition of each generated set \( S \) is continued until tests reveal either that \( S \) is singleton (in which case, we measure its value directly and compare with the currently best member's cost) or that there is an optimum element \( x^* \) not in \( S \) (in which case, the set is "pruned" or eliminated from further consideration). If the decomposition process is continued (and satisfies some properties), an optimum element will eventually be found. The utility of this approach derives from the fact that, in general, most of \( X \) will be pruned, whence only a small fraction of \( X \) need be enumerated.

\(^1\) Discussion in this article is also applicable (with appropriate modifications) to the case when a largest cost element is desired; i.e., an optimal element is defined as an element \( x^* \) of \( X \) such that \( f(x^*) = \max \{ f(x) \mid x \in X \} \).
Branch-and-Bound Search

B&B techniques appear to have been conceptualized in the early 60's to tackle integer programming and nonlinear assignment problems. Later similar techniques, with some modifications, were found to be applicable in many other problem domains. As more and more applications were discovered, the B&B methodology evolved. Various formal models of B&B were presented and later superseded by more general models [1], [17], [10], [8], [9], [11].

In the earlier formulations of B&B [17], [1], only the lower and upper bounds on the costs of the elements of (sub)sets (of X) were used for pruning. If two sets $X_1$ and $X_2$ are in the collection of sets under consideration, and the lower bound on the costs of elements in $X_1$ is no smaller than the upper bound on the costs of elements in $X_2$, then $X_1$ can be pruned. The use of bounds for pruning gave the procedure its name branch-and-bound.

The concept of pruning by bounds was later generalized to include pruning by "dominance" (see [10], [8], [9], [11]). Dominance may be used to perform pruning even when adequate lower-bound information is not available. Pruning by dominance provides a powerful mechanism for using problem-specific knowledge for efficient search of an optimum element of the set $X$.

**A GENERAL B&B FORMULATION**

Here we briefly describe the basic elements of a branch-and-bound formulation. The formulation presented here is very similar to the one given in [13]. As is discussed in [13], the dominance relation in our formulation is used for pruning in a manner somewhat different than in [10], [8], [9].
Basic Definitions

Let $Y$ be the set of all subsets of $X$, i.e., $Y = 2^X$. $X_i$ denotes a subset of $X$, and $A$ denotes a collection of subsets of $X$ (i.e., $A \subseteq Y$). For brevity, $A$ will sometimes be referred to simply as a ‘collection’. For notational convenience, the union of all subsets in any collection $A$ is denoted by $\bigcup(A)$; i.e., $\bigcup(A) = \bigcup\{X_i \mid X_i \in A\}$. We define $f^*(X_i)$ to be the minimum of the costs of the elements in $X_i$. Any element $x^* \in X_i$ such that $f(x^*) = f^*(X_i)$ is called an optimum element of $X_i$.

A branching function $\text{BRANCH}$ is any function which divides the members of the collection $A$ into subsets which collectively include precisely the same elements of $X$ as the original collection $A$. Mathematically, it is any function mapping collections into collections such that:

(i) $X_i \in \text{BRANCH}(A) \Rightarrow X_i \subseteq X_j$ for some $X_j \in A$.

(ii) $\bigcup(\text{BRANCH}(A)) = \bigcup(A)$.

Often the function $\text{BRANCH}$ is defined as a composition of selection and splitting functions. A selection function is any function $\text{SELECT}$ mapping collections into collections such that $\text{SELECT}(A) \subseteq A$. A splitting function $\text{SPLIT}$ is any function satisfying the properties of a branching function. $\text{BRANCH}$ is then defined as

$$\text{BRANCH}(A) = (A - \{\text{SELECT}(A)\}) \cup \text{SPLIT}(\text{SELECT}(A)).$$

Although this definition of $\text{BRANCH}$ is mathematically equivalent to the one given above, it emphasizes the characteristic that only the elements from a certain selected subset of the collection $A$ are divided, and the rest are returned unchanged. In fact, in many implementations of $\text{BRANCH}$, only one selected element from the collection $A$ is divided, and the rest are returned unchanged.
The dominance relation $D$ is the binary relation between subsets $X_i, X_j$ of $X$ such that $X_i D X_j$ if and only if $f^*(X_i) \leq f^*(X_j)$. Clearly, if $X_i$ and $X_j$ are present in a collection $A$ and $X_i$ dominates $X_j$, then $X_j$ can be pruned.

The pruning function $\text{PRUNE}$ prunes a dominated subset of $A$. It is defined as $\text{PRUNE}(A) = A - A^D$, where $A^D$ is a subset of $A$ such that for all $X_i \in A^D$ there exists some $X_j \in A - A^D$ such that $X_j D X_i$. From the definition of dominance, it follows that $A - A^D$ will contain at least one optimal element of $\cup(A)$.

An Abstract B&B Procedure

The procedure $\text{BB}$ given below represents the essence of many B&B procedures. Here, $A$ denotes the collection of subsets of $X$ upon which the branching and pruning operations are performed in each iteration of $\text{BB}$, and $|S|$ denotes the cardinality of a set $S$.

procedure $\text{BB}$ (* B&B procedure to search for an optimum element of a set $X$ *)

begin
    $A := \{X\};$ (* initialize the collection $A$ *)

    while $|\cup(A)| \neq 1$ do (* loop until $A$ contains only one element of $X$ *)
        $A := BRANCH(A);$ (* branch on the collection $A$ *)
        $A := PRUNE(A)$ (* eliminate the dominated subsets from $A$ *)
    end
end

No element of $X$ is lost from $A$ in the branching operation, and at least one optimal element of $\cup(A)$ is there in $\text{PRUNE}(A)$. Hence, if the procedure $\text{BB}$ terminates, $A$ contains only an optimal element of $X$. Note that the termination of $\text{BB}$ is not guaranteed. In order to guarantee the termination of $\text{BB}$, $\text{BRANCH}$ and $\text{PRUNE}$ must satisfy certain
additional properties.

Best-first Branch-and-Bound

In many problem domains it is possible to define a function $lb$ on the subsets $X_i$ of $X$ such that for all $x \in X_i$, $lb(X_i) \leq f(x)$; i.e., $lb(X_i)$ is a lower bound on the costs of the elements of $X_i$. Furthermore, we require that for all $x \in X$, $lb(\{x\}) = f(x)$; i.e., the lower bounds for singleton sets are not unnecessarily loose. This lower bound information can be fruitfully used in selecting an element for branching. If in every cycle of BB's loop an element of $A$ is chosen for branching which has the least lower bound of all the elements of $A$, then the selection rule is called best-first, and the branch-and-bound procedure using this strategy is called best-first branch-and-bound. An interesting feature of best-first B&B is that whenever a singleton set $\{x\}$ is selected for branching, the procedure can terminate. This is because $f^*(\{x\}) = f(x) = lb(\{x\}) \leq lb(X_i) \leq f^*(X_i)$ for all $X_i \subseteq A$, and thus $\{x\}$ dominates all the other elements in $A$.²

If the bounds $lb(X_i)$ are good approximations of $f^*(X_i)$, then best-first B&B can be very efficient. In the extreme case, if $lb(X_i) = f^*(X_i)$ for all $X_i \subseteq X$, then the B&B procedure finds an optimal element of $X$ by splitting only those sets which contain optimal elements. On the other hand, if the bounds $lb(X_i)$ are not good approximations of $f^*(X_i)$, then best-first B&B can be very inefficient and may require a lot of storage.

---

² If we select more than one element of $A$ for branching, then the selection rule is still called best-first, as long as at least one of the selected elements (let's call it $X_j$) has the least lower bound of all the elements of $A$. In such cases, B&B can successfully terminate if $X_j$ is singleton.
Depth-first Branch-and-Bound

Another way of selecting a set for branching from the active collection $A$, is to select a set from those sets which have been generated most recently as a result of branching [16]. This selection rule is called depth-first, and the branch-and-bound procedure using such a rule is called depth-first branch-and-bound. The major advantage of the depth-first selection rule over the best-first selection rule is that, in general, it requires less storage. But the depth-first search, being an uninformed search, can be much slower than the best-first search. See [7] for a theoretical comparison of various search strategies used in B&B procedures. Smith [22] presents a model of B&B search trees for a class of B&B procedures, and studies expected time and space complexities of B&B for different search strategies.

More on Branching and Pruning

In this abstract formulation, a number of details have been left out. For example, we have only defined the basic properties of a branching function. In a practical implementation of a B&B procedure, a branching function is chosen which is natural for the problem domain in question and satisfies the properties given here. The splitting function is usually suggested by the problem. But sometimes, a problem may have many natural splitting functions (e.g., the traveling salesman problem has many natural splitting functions [21]). The selection function can be depth-first, best-first or some other function suitable for the problem at hand.

For pruning, in each cycle of BB, a dominated subset $A^D$ of the collection $A$ needs to be constructed. Note that for any two subsets $X_1$, $X_2$ of $X$, at least one of them dominates the other (either $f^*(X_1) \geq f^*(X_2)$, or $f^*(X_2) \geq f^*(X_1)$). Hence, in theory $A^D$ could
be constructed to have all but one set of the collection \(A\). This would make the procedure BB terminate in a very few cycles, since in every cycle of BB, all but one of the generated sets will be eliminated. In practice, we may not know which sets in \(A\) dominate which other sets in \(A\) without exhaustively enumerating the elements in the sets which are members of \(A\). However, partial knowledge from the problem domain is often available to reveal that certain sets in \(A\) dominate certain other sets in \(A\). This partial knowledge of the dominance relation can be used to construct a dominated subset \(A^D\), of \(A\).

Next we present a practical B&B procedure to find a shortest path in a directed graph, and show how domain knowledge about directed graphs is used to devise practical branching and pruning functions. See [16], [6], [21] for other applications of B&B.

Example: Finding a Shortest Path in a Graph

We are given a directed graph \(G\). Each arc \((n,m)\) in \(G\) has a cost \(c(n,m) \geq 0\), and for every path \(P\) in \(G\), \(\text{cost}(P)\) is defined as the sum of the arc costs of \(P\). The problem is to find a least-cost directed path from a source node \(s\) in \(G\) to a terminal node \(t\) in \(G\). For this problem, \(X\) is the set containing each path from \(s\) to \(t\). For \(x \in X\), \(f(x) = \text{cost}(x)\).

If \((m,n)\) is an arc in \(G\), then \(n\) is called a successor of \(m\). Suppose \(P = (n_1, n_2, \ldots, n_j)\) is a path in \(G\), then \(Pn\) is the path \((n_1, n_2, \ldots, n_j, n)\). We can use a path \(P\) from \(s\) to a node \(n\) to represent the set of paths in \(X\) which are extensions of \(P\).\(^8\) Let \(n_1, \ldots, n_k\) be successors of \(n\) in \(G\), then a natural splitting function on \(P\) is \(\text{SPLIT}(P) = \{Pn_i \mid 1 \leq i \leq k\}\).

\(^8\) In actual implementations of B&B, the set \(X\) and its subsets are not represented explicitly. Instead, some problem-specific data structure is used which implicitly represents \(X\) and its subsets [19].
larger cost than the extension of $P''$. Thus the structure of the graph reveals dominance between two paths even when adequate bound information is not available. Note that we may use lower bound information to conclude $P' \geq D \geq P''$ (or $P'' \geq D \geq P'$) only if $n$ is the terminal node. Fig. 3 shows the operation of a best-first B&B using dominance for pruning on the graph of Fig. 1.

First two steps as shown in Fig. 3.a and 3.b are identical to the ones shown in Fig. 2.a and 2.b. Fig. 3.c shows $A$ after splitting $(s,n_1)$. The path $(s,n_2)$ dominates $(s,n_1,n_2)$ because they both end at $n_2$ and $\text{cost}((s,n_2)) \leq \text{cost}((s,n_1,n_2))$; hence $(s,n_1,n_2)$ is pruned. In the next step $(s,n_2)$ is chosen for splitting. As shown in Fig. 3.d, the resulting path $(s,n_2,t)$ dominates $(s,n_1,n_2)$; hence $(s,n_1,n_2)$ is pruned. Now $A$ contains only $(s,n_2,t)$ which is a singleton set. Therefore BB terminates with $(s,n_2,t)$ as a shortest path between $s$ and $t$.

**RELATIONSHIP WITH AI SEARCH ALGORITHMS**

The central idea of branching and pruning to discover an optimal element of a set is at the heart of many AI search algorithms. For example, the $A^*$ algorithm [20] for state-space search can be viewed as a best-first B&B procedure [19]. $A^*$ differs only slightly from the best-first B&B procedure for finding a shortest path presented in this article, as $A^*$ uses a more informed lower bound. The nodes on the OPEN list in $A^*$ represent the active set of paths upon which branching and pruning is performed. The process of node selection and expansion corresponds to branching. Node elimination corresponds to pruning dominated paths. Many other heuristic procedures for searching state-space graphs can be viewed as B&B procedures using different selection and pruning functions. The AND/OR graph search algorithm AO* [20] and the game tree search
Branch-and-Bound Search

procedures alpha-beta [20], SSS* [23] and B* [2] can also be viewed as B&B procedures [13]. In [15] it is shown that all these procedures can be viewed as instantiations of a general B&B procedure for searching AND/OR graphs.

RELATIONSHIP WITH DYNAMIC PROGRAMMING

B&B procedures are closely related to dynamic programming. Historically, the earlier dynamic programming procedures used "structural" dominance for pruning, but did not use bounds. Selection was essentially breadth-first. Whereas, as noted before, the earlier formulations of B&B used bounds for selection and pruning but did not use dominance (in its general form) for pruning. In the recent formulations of dynamic programming [18], [12] and branch-and-bound, both dominance and bounds are used. Nevertheless, B&B and dynamic programming are different techniques for solving optimization problems. As discussed in [11], [14], dynamic programming can be viewed as a bottom-up search, whereas B&B can be viewed as a top-down search. In the context of state-space graphs, the difference between them vanishes, as for every state-space graph, it is possible to construct a dual state-space graph such that the top-down search of one is equivalent to the bottom-up search of the other [14]. That is why Dijkstra's algorithm [3] for finding shortest path in a graph could be classified both as B&B [5] and as dynamic programming [4].

REFERENCES


[15] V. Kumar, D. S. Nau, and L. N. Kanal, A General Paradigm for AND/OR Graph and Game Tree Search. Submitted for publication., Tech. Report, Computer Science Department, University of Texas at Austin, Austin, TX 78712., 1985.


Fig. 1. A directed graph $G$. Arc costs are given next to the arcs.

Fig. 2. Steps of a best-first B&B search procedure using bounds for pruning.
Fig. 3. Steps of a best-first B&B search procedure using dominance for pruning.