A General Paradigm for A.I. Search Procedures*

Dana S. Nau, Vipin Kumar, and Laveen Kanal

Laboratory for Pattern Analysis
Computer Science Department, University of Maryland, College Park, MD 20742

ABSTRACT

This paper summarizes work on a General Branch and Bound formulation which includes previous formulations as special cases and provides a unified approach to understanding most heuristic search procedures developed in A.I.

I. INTRODUCTION

A wide class of problems arising in Operations Research, decision making and Artificial Intelligence can be (abstractly) stated in the following form:

> Given a (possibly infinite) discrete set X and a real-valued objective function F whose domain is X, find an optimal element $x^* \in X$ such that $F(x^*) = \min\{F(x) | x \in X\}$.

Exhaustive enumeration of this set X for determining an optimum element is too inefficient for most practical problems. Hence, procedures (e.g. Branch and Bound [LAW66], A*, AO*, Alpha-Beta [NIL80], and B* [BER79]) have been developed to solve various versions of this problem efficiently by utilizing problem-specific knowledge. The underlying idea of such procedures is to decompose (or split) X into smaller and smaller sets. The utility of this approach derives from the fact that in general most of these sets will be pruned (or eliminated from further consideration), whence only a small fraction of X need be enumerated. In this paper we summarize a general abstract formulation of Branch and Bound(B&B), which extends previous work by many researchers (e.g. Mitten [MIT70], Lawler and Wood [LAW66], Balas [BAL68], Smith [SMI79], Reingold, Nievergelt, and Deo [REI77], Horowitz and Sahni [HOR78], Kohler and Steiglitz [KOH74], and Ibaraki [IBA78]). Until recently, only upper and lower bounds were used for pruning in B&B procedures (hence the name Branch and Bound). Many A.I. search procedures (A*, AO*, alpha-beta, etc.) use more sophisticated dominance relations to prune the sets, although they perform branching (i.e., set splitting) in a similar form. This caused at least some of the confusion ([POH72], [HAL71], [MAR78]) as to whether A*, AO* and other heuristic procedures are really Branch and Bound.

The addition in B&B of the new concept of dominance in pruning was introduced by Kohler & Steiglitz [KOH74] and further investigated by Ibaraki [IBA78]. Our formulation of B&B simplifies and generalizes the idea of dominance: we allow a set to be pruned when at least one of the remaining sets contains an optimal element. All other pruning techniques can be considered as special cases of this approach.

We have shown [NAU82] that A* and AO* are special cases of our general B&B formulation. Similar results can be given for SSS*, Alpha-Beta, B*, etc. ([KAN81],[KUM81],[KUM82]).

II. THE BASIC CONCEPT OF GENERAL BRANCH & BOUND

Our basic concept of General Branch and Bound is the procedure below. (comments are indicated by double slashes ("//")):

```
procedure PO:
1.  ACT := {X}  // ACT is the current active set //
2.  loop
3.      if ACT = {Z} for some Z
                and Z is a singleton {z} then
4.          return z
5.      endif
6.      SEL := select(ACT)
            // select some of the sets in ACT //
7.      SPL := split(SEL)
                // split the sets in SEL //
8.      ACT := prune((ACT-SEL) U SPL)
            // remove the selected sets from  //
            // ACT, replace them by the newly //
            // generated sets, and then prune //
            // unneeded sets from ACT //
9.  repeat
end PO
```

ACT, the active set, is a collection of subsets of X. select, the selection function, is any function which returns a collection SEL $\subseteq$ ACT. The domain of select is the set of all possible values which ACT might have at line 6 of PO.

* This work was supported by NSF Grant ENG-7822159 and NSF Grant MCS81-17391 to the Laboratory for Pattern Analysis at the University of Maryland.

split, the splitting function, has as its domain the set of all possible values which the collection SEL might have at line 7 of P0. split(SEL) returns a collection SPL of subsets of X such that--

1. every set in SPL is a subset of some set in SEL;

2. $\cup\{Y' \mid Y' \in SPL\} = \cup\{Y \mid Y \in SEL\}$; i.e., the sets in SPL contain precisely those elements which are members of the sets in SEL.

prune, the pruning function, has as its domain the set of all possible values which the collection of sets

$$R = (ACT-SEL) \cup SPL$$

might have at line 8 of P0. prune returns a collection of sets $R' \subseteq R$ such that

$\min\{F(y) \mid y \in Y \text{ for some } Y \in R'\} =$
$\min\{F(y) \mid y \in Y \text{ for some } Y \in R\}$;

i.e., at least one of the minimum elements of R is also present in R'.

In [NAU82], this concept is developed more fully by dealing with the ways in which the members of the active set ACT are represented, and the use of problem-dependent auxiliary data in the selection, splitting, and pruning functions. In [NAU82], it is also shown how formulations of B&B in the literature are special cases of GBB.

Because the invariance:
$\min\{F(x) \mid x \in X\} = \min\{F(x) \mid x \in Y \text{ for some } Y \in ACT\}$
remains true through out the execution of the procedure P0, it is easy to see that, at termination, P0 finds an optimum element of X.

In various problem domains, it is possible to easily compute lower bounds on the F-values of the elements of subsets of X present in the active list ACT. These bounds can be used to perform pruning by the prune function. Also suppose that: (i) the lower bound on a singleton set {x} is always F(x), and (ii) the selection function is best-first, i.e., the set chosen from ACT by the function select is always the one having the least lower bound. Then it can be shown that the first singleton set selected by the select function will be an optimum element of X. This makes the procedure even more efficient. In this case, P0 can be rewritten as follows.

```
procedure P3B:  //best-first GBB//
1.  ACT3 := {X}
2.  loop //the main loop//
3.      if ACT3 =∅ then return "unknown" endif
4.      SEL3 := select3(ACT3)
5.      if SEL3 is a singleton {x} and goal(x) then
6.          return x
7.      else
8.          SPL3 := split3(SEL3)
9.          ACT3 := prune3(ACT3 - SEL3, SPL3)
10      endif
11. repeat
end P3B
```

III. A*

The well-known A* algorithm [NIL80] is a procedure for finding a least-cost path on a graph. To consider A* as a special case of GBB, we note that each node on the OPEN list of A* actually represents a path P from the source node to n. The set X consists of all paths from the source node to any goal nodes, and P represents the subset of X consisting of all extensions of P to goal nodes. This allows A* to be rewritten as an instantiation of P3B as follows.

```
procedure P7:  // A*, rewritten //
2.  ACT7 := list containing the null path
            from s to s
3.  GEN7 := NIL
4.  while ACT7 ≠ NIL do
5.      {P} := select7(ACT7)
            // select first member P of ACT7 //
6.      insert P into GEN7
7.      if goal7(P) then
8.          return P
9.      else
10          SPL7 := split7({P})
11          ACT7 := ACT7 - {P}
12          for every path Pn in SPL7 do
13              for every Q in ACT7 or GEN7 do
14                  if tip(Q)=n and L7(Q) ≤ L7(Pn) then
15                      goto PRUNE  // prune Pn //
16                  elseif tip(Q)=n and L7(Pn) < L7(Q)
17                      then call remove7(Q)
18                  endif
19              endfor
20              parent(Pn) := P
21              ACT7 := insert7(Pn,ACT7)
                // insert Pn into ACT7 after all  //
                //nodes n such that f'(n) < f'(Pn)//
22.  PRUNE: endfor
23.     endif
24. endwhile
25. return "unknown"
end P7


procedure remove7(P)
1.  if P ∈ ACT7 then remove P from ACT7 endif
2.  if P ∈ GEN7 then remove P from GEN7 endif
3.  for every Q such that parent(Q)=P do
4.      call remove7(Q)
5.  endfor
end remove7
```

(1) The active list (ACT7), which corresponds to the OPEN list in the usual formulation of A*, is a set of paths from the source node to various nodes in the graph.

(2) goal7(P) holds only if P is a path from the source node to a goal node.

(3) select7(ACT7) returns the first member of ACT7. This is the path P in ACT7 having the least lower bound L7(P), where L7(P) = cost(P) + h(tip(P)), and h is the usual A* heuristic function.

(4) split7(P) returns the paths created by expanding the tip node of P.

(5) insert7(P,ACT7) inserts P into ACT7 just after the last path Q in ACT7 such that L7(Q) < L7(P). It is shown in [NAU82] that the pruning done in P7 satisfies the properties of a pruning function.

IV. AO*

AO* [NIL80] is a procedure for finding a least-cost solution graph in a hypergraph (a formalization of an AND/OR graph). To consider AO* as a special case of GBB, we note that the search graph maintained in AO* actually represents a collection of partial solution graphs. The set X is the set of all complete solution graphs of the hypergraph searched, and each partial solution graph P represents the subset of X consisting of all extensions of P to complete solution graphs. The partial solution graph found in AO* by tracing the marked nodes is the one having the least lower bound. Thus AO* can be rewritten as an instantiation of P3B as follows.

```
procedure P9: //AO*, rewritten//
1.  ACT9 := the partial solution graph
            containing only the source nodes
2.  loop
    //the test below will never succeed, and is
      included merely to illustrate that P9 is
      an instantiation of P3B//
3.  if ACT9 = ∅ then return "unknown" endif
4.  SEL9 := select9(ACT9)
5.  if SEL9 is a singleton {r} and goal (r) then
6.        return r
7.  else
8.        SPL9 := split9(SEL9)
9.        ACT9 := prune9(ACT - SEL9, SPL9)
10. endif
11. repeat
end P9
```

The functions used above are defined as follows.

(1) The active set (ACT9), which corresponds to the search graph G in Nilsson's formulation of AO*, is the set of all partial solution graphs in G.

(2) goal9(P) holds only if P is a complete solution graph.

(3) select9 returns the member P of ACT9 having the least lower bound L9(P). This happens to be the partial solution graph found in Nilsson's version of AO* by tracing the marked connectors.

(4) L9(P) (the lower bound mentioned in item 3) is the sum of all of the arc costs of P, plus the sum of the h-values of the tip nodes of P. This is the same as the value q(P) maintained by Nilsson.

(5) split9(SEL9) returns the set of all partial solution graphs in SEL9 which contain n, where n is the node select in Nilsson's version of AO*.

(6) prune9(ACT9 - SEL9, SPL9) returns the set of all partial solution graphs in the search graph formed by expanding the node n selected in Nilsson's version of AO*.

V. CONCLUSIONS

We have summarized our work showing that the A.I. search procedures A* and AO* are special instances of our general branch and bound formulation. It can be shown that a number of other A.I. procedures are also special cases of GBB. It is possible to visualize many variations of existing search algorithms being generated from this Branch and Bound paradigm, which provides a theoretical basis for a better understanding of the performance of such algorithms and the relationships among them (e.g., [KAN81],[KUM81],[KUM82]). In particular, we conjecture that all procedures for top-down search of problem reduction representations can be examined and understood as instantiations of this General Branch and Bound procedure.

REFERENCES

BAL68 Balas, E. A Note on the Branch-and-Bound Principle. Operations Research 16 (1968), 442-444. Errata p.886.

BER79 Berliner, H. The B* Tree Search Algorithm: A Best-First Proof Procedure. Artificial Intelligence 12 (1979), 23-40.

HAL71 Hall, P. A. V. Branch-and-Bound and Beyond. Proc. Second Internat. Joint Conf. Artif. Intell. (1971), 641-658.

HOR78 Horowitz, E. and Sahni, S. Fundamentals of Computer Algorithms. Computer Science Press, Potomac, MD, 1978.

IBA77 Ibaraki, T. The Power of Dominance Relations in Branch and Bound Algorithms. J. ACM 24 (1977), 264-279.

IBA78 Ibaraki, T. Branch-and-Bound Procedure and State-Space Representation of Combinatorial Optimization Problems. Information and Control 36 (1978), 1-27.

KAN79 Kanal, L. Problem-Solving Models and Search Strategies for Pattern Recognition. IEEE Trans. Pattern Analysis and Machine Intell. 1 (1979), 193-201.

KAN81 Kanal, L. and Kumar, V. A Branch and Bound Formulation for Sequential and Parallel Game Tree Searching. Proc. Seventh International Joint Conference on Artificial Intelligence, Vancouver (August 1981), 569-571.

KAN81b Kanal, L. and Kumar, V. Parallel Implementations of a Structural Analysis Algorithm. Proc. IEEE Computer Society Conf. Pattern Recognition and Image Processing, Dallas (Aug. 1981), 452-458.

KOH74 Kohler, W. H. and Steiglitz, K. Characterization and Theoretical comparison of Branch-and-Bound Algorithms for Permutation Problems. J.ACM 21 (1974) 140-156.

KUM81 Kumar, V., and Kanal, L. Branch and Bound Formulations for Sequential and Parallel And/Or Tree Search and Their Applications to Pattern Analysis and Game Playing. submitted for publication 1981.

KUM82 Kumar, V., Nau, D. and Kanal, L. A General Model for Problem Reduction and Game Tree Search. working paper 1982.

LAW66 Lawler, E. L., and Wood, D. E. Branch-and-Bound Methods: A Survey. Operations Research 14 (1966), 699-719.

MAR73 Martelli, A. and Montanari, U. Additive AND/OR Graphs. Proc. Third Internat. Joint Conf. Artif. Intell. (1973), 1-11.

MAR78 Martelli, A. and Montanari, U. Optimizing Decision Trees through Heuristically Guided Search. Comm. ACM 21 (1978), 1025-1039.

MIT70 Mitten, L. G. Branch and Bound Methods: General Formulations and Properties. Operations Research 18 (1970), 24-34. Errata in Operations Research 19 (1971), 550.

NAU82 Nau, D., Kumar, V. and Kanal, L. General Branch & Bound, and Its Relation to A* and AO*. working paper,1982.

NIL80 Nilsson, N. Principles of Artificial Intelligence. Tioga Publ. Co., Palo Alto, CA, 1980.

POH72 Pohl, I. Is Heuristic Search Really Branch and Bound? Proc Sixth Annual Princeton Conf. Inform. Sci. and Systems (1972), 370-373.

RIE77 Reingold, Nievergelt and Deo N. Combinatorial Optimization. Prentice Hall, 1977.

SMI79 Smith, D. R. On the Computational Complexity of Branch and Bound Search Strategies, Ph.D. Dissertation, Duke Univ., Durham, NC, 1979. Tech. Rep. NPS 52-79-114, Naval Postgraduate School, Monterey, CA, 1979.