Technical Report TR-226 January 1973

PROPERTIES OF HEURISTIC SEARCH STRATEGIES

by

G. J. VanderBrug

UNIVERSITY OF MARYLAND
COMPUTER SCIENCE CENTER
COLLEGE PARK, MARYLAND

Technical Report TR-226 January 1973

PROPERTIES OF HEURISTIC SEARCH STRATEGIES

by

G. J. VanderBrug

This work was supported by the National Aeronautics and Space Administration Grant NGR-21-002-270.

The author acknowledges the valuable assistance of his advisor Professor Jack Minker.

TABLE OF CONTENTS

ABSTRACT

A directed graph is used to model the search space of a state-space representation with single-input operators, an AND/OR is used for problem-reduction representations, and a theorem-proving graph is used for state-space representations with multiple-input operators. This paper surveys these three graph models and heuristic strategies for searching them. The completeness, admissibility, and optimality properties of search strategies which use the evaluation function $f = (1 - \omega)g + \omega h$ are presented and interpreted using a representation of the search process in the plane. The use of multiple-output operators to imply dependent successors, and thus obtain a formalism which includes all three types of representations, is discussed.

1. Introduction

Automatic problem-solving is usually divided into two parts. The first part, representation, is concerned with the process of formulating the problem to the problem-solver. The problem-solver uses this description of the problem to determine a solution. The manner in which the problem-solver looks for a solution is the second part of automatic problem-solving and is called search.

Early efforts in automatic problem-solving were directed towards constructing systems for solving problems from a restricted domain, and were for the most part empirical in nature. As work continued, in addition to the construction of more ad hoc systems, general methods and approaches were abstracted from these systems, and a collection of theoretical results has begun to form. One of these approaches is the use of graphs to model the heuristic search process. Trees and graphs were first used as a model, later AND/OR graphs, and more recently theorem-proving graphs have been used. General procedures, which allow for the introduction of problem-dependent heuristic information for searching these graph models have been developed, and properties such as completeness, admissibility, and optimality of these procedures under specified conditions have been proved.

Section 2 describes the use of directed graphs to model state-space representations and the types of heuristic search strategies used to look for a solution. Section 3 describes a method of representing the search process in the coordinate plane, and Section 4 uses it to interpret the completeness, admissibility, and optimality properties of heuristic search strategies. Direct proofs of these properties for search strategies which use the evaluation function $f = (1-\omega)g + \omega h$ are given in the Appendix. A search strategy for AND/OR graphs similar in structure to those for directed graphs, and the properties of this search strategy when the above evaluation function is used, are discussed in Section 5. Section 6 contains a similar discussion for theorem-proving graphs. AND/OR graphs are used to model the search in problem-reduction representations, and theorem-proving graphs are used to model the search in state-space representations with multiple-input operators.

2. Problem-Solving Using the Directed Graph Model

Many problems can be represented as state-space problems [Michie, 1970; Nilsson, 1971; Simon, 1971]. A *state-space representation* consists of a set of states (including start and goal states), and a set of operators which map states into states. A solution is a sequence of operators which transforms the start state into a goal state. By associating states with nodes and operators with edges, a directed graph can be used as a model for state-space representations which have operators with one input state and one output state (single-input/single-output operators). Usually only the form of a state is specified, and thus a representation implicitly defines a graph. The search process can be thought of as making explicit a part of an implicitly defined graph which, when the search is successful, includes a path from the start node to a goal node.

Algorithms for searching directed graphs fall into two categories [Sandewall, 1971]. The *labyrinthic methods* make a decision to search along a direction in the graph, and after searching for some time decide either to continue in this direction or to backup and try another direction. Labyrinthic methods were used in GPS [Ernst and Newell, 1969]. The *best-bud methods* assign a *merit ordering* to the set of nodes, typically by using an *evaluation function*. At each stage of the search these algorithms take a global look at all of the nodes which have been generated, but not yet expanded, and expand the node with the best merit. For example, if the merit ordering ranks the nodes according to the inverse of their levels, then the best-bud approach will result in a depth-first search of the graph. These algorithms attempt to generate the nodes of the graph according to the merit ordering. In general they do not generate the nodes according to the merit ordering, because the node n_1 may have an ancestor n_2 which has poorer merit, and hence n_1 cannot be generated until some time after the generation of n_2. In this paper the term *search strategy* will refer to a best-bud method for searching a graph.

The basic form of a search strategy is as follows.

1. Place the start node s in the set S (the set of nodes currently being considered for expansion).
2. Expand the node n in S with the best merit. If n is a goal node, then we have found a solution. Otherwise, place the successors of n in S, place n in the set $\tilde{S}$ (the set of nodes which have been expanded), and repeat this step.

This is a simplified form since, for example, it does not consider the possibility of discovering that a node which has already been expanded has better merit than was originally assigned to it.

One of the earliest search strategies is the uniform-cost strategy which uses the evaluation function $f = g$, where g is the cost from the start node to the current node [Nilsson, 1971]. The Graph Traverser [Doran and Michie, 1966; Doran, 1967] assigned a merit ordering with the function $f = h$, where h is an estimated cost from the current node to the nearest goal node. Hart, Nilsson, and Raphael [1968] developed an algorithm, called the A* algorithm, which used the evaluation function $f = g + h$ to assign a merit ordering to the nodes. Pohl [1970a] considered the evaluation function $f = (1-\omega)g + \omega h$, where $0 \leq \omega \leq 1$, which includes the uniform cost algorithm $(\omega=0)$, the A* algorithm $(\omega=\frac{1}{2})$, and the pure heuristic algorithm $(\omega=1)$.

A search strategy is said to be *complete* if whenever there exists a solution to the problem the strategy will find one. An *admissible* strategy is one which terminates with a minimal solution whenever one exists. The concept of *optimality* applies to strategies which are admissible and is defined as follows. Let h_1 and h_2 be two heuristic functions such that $h_2 < h_1 \leq h_p$, where h_p is a perfect heuristic function. An admissible strategy is said to be optimal if searching with h_2 expands all of the nodes that searching with h_1 expands. Admissibility can be viewed as the optimality of the solution, whereas optimality is really the optimality of the search process.

The uniform-cost strategy searches along contours of equal cost from the start node, and it is easy to understand why the first solution found is a minimal cost solution. Hart et al. [1968] showed that a heuristic component can be used to direct the search and still retain the admissibility property. They proved that if A* used a heuristic function h which satisfied the *lower bound condition* ($h \leq h_p$) then A* is admissible for *δ-graphs* (graphs where the arc costs are greater than or equal to some $\delta > 0$). They also proved that if A* used a *consistent* heuristic function ($h(n) + h(n') \leq k(n,n')$, where $k(n,n')$ is the cost to go from n to n') then A* is optimal for δ-graphs. Pohl[1969, 1970a] showed that search with $f = (1-\omega)g + \omega h_p$ for $\omega \in [\frac{1}{2},1]$ expands only the nodes on a minimal solution path. In the same paper he considered the case where the error in the heuristic function h is bounded by ϵ ($h_p - \epsilon \leq h \leq h_p + \epsilon$), and showed with a worst case analysis, that search with $\omega = 1$ will expand at least as many nodes as search with $\omega = \frac{1}{2}$. Pohl [1970b] has also shown that a search strategy with $\omega \in [0,1)$ is complete for the special case when the graph has unit arc costs. A bidirectional procedure which is admissible for heuristics which satisfy the lower bound condition has been developed [Pohl, 1969, 1971] for searching directed graphs.

3. Representing the Search Process in the Plane

Kowalski [1970] has introduced a method of using one quadrant of the coordinate plane to represent a search space. In this section we describe this method and show how it can be used to represent the direction of search for the evaluation functions $f = (1-\omega)g + \omega h$ and $f_\alpha = g + \alpha h$.

Each node n in the search space is represented at the point in the coordinate plane (i,j) , where $h(n) = i$ and $g(n) = j$. The start node (or nodes if there is more than one) is located on the horizontal h-axis, and all goal nodes are located on the vertical g-axis. Each heuristic function h determines the locations of the nodes in the quadrant. Strictly speaking a search strategy may place a node in more than one position during the search process. The h-component of a node never changes, but the g-component may assume as many values as there are paths from the start node with distinct costs. The δ-finiteness of a graph guarantees that only finitely many nodes are contained in any finite region of the plane.

Each choice of ω in $f_\omega = (1-\omega)g + \omega h$ determines the direction in which the space will be searched. Kowalski calls search with $\omega = \frac{1}{2}$ *diagonal search* because it defines all nodes which lie on the same diagonal to have equal merit and attempts to generate nodes in the direction indicated in Fig. 1.a. *Upwards diagonal search* differs from diagonal search in that nodes with the same f-value do not necessarily have equal merit. It defines the node n to have better merit than the node n' iff $f(n) < f(n')$, and $h(n) < h(n')$ when $f(n) = f(n')$; and attempts to generate nodes in the same direction as diagonal search except that the search proceeds up successive diagonals as indicated in Fig. 1.b. Figures 1.c - 1.f illustrate the direction of search for various values of ω . If the distinction between the h-components is made for nodes with the same f-value as is done in upward diagonal search, then along each line in these figures the search proceeds in the direction of the h-axis.

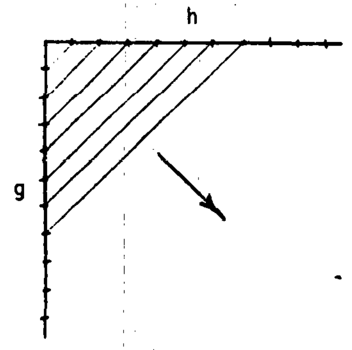


Fig. 1.a $\omega=\frac{1}{2}$, Diagonal Search.

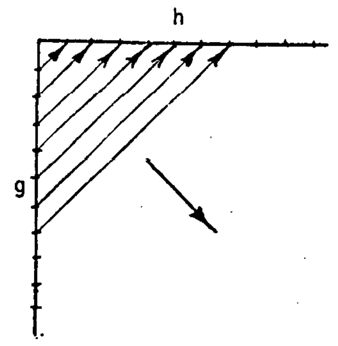


Fig. 1.b $\omega=\frac{1}{2}$, Upper Diagonal Search.

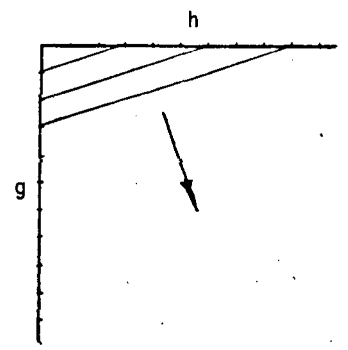


Fig. 1.c $\omega=\frac{1}{4}$.

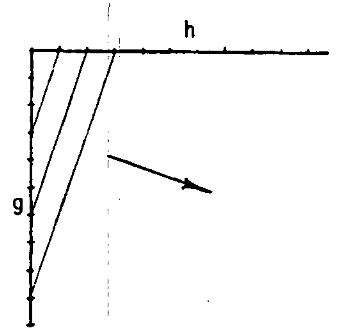


Fig. 1.d $\omega=\frac{3}{4}$.

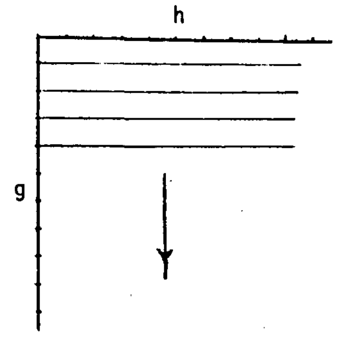


Fig. 1.e $\omega=0$.

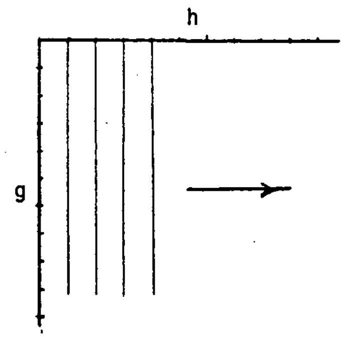


Fig. 1.f $\omega=1$.

Fig. 1. The direction of search in the coordinate representation of the search space for various values of ω.

As the process of searching the graph proceeds the area of the quadrant covered grows. All the nodes which lie in this area are not immediately expanded, because paths from the start node to them may not have yet been found. A node n which lies in the region which has been searched will be expanded provided there is a path from the start node to n such that all of the nodes on this path also lie in the region.

A merit ordering can be defined by an evaluation function written either as $f_\omega = (1-\omega)g + \omega h$ for $\omega \varepsilon [0,1]$, or as $f_\alpha = g + \alpha h$ for $\alpha \varepsilon [0,\infty)$. Since scaling the evaluation function does not affect the order in which the nodes are generated, the relationship between these two forms is given by $\alpha = \omega/(1-\omega)$. When the first form is used changes in the value of ω correspond most naturally to changing the direction of search as illustrated in Fig. 1. However, the parameter α in the second form can be thought of as being part of the heuristic component of the evaluation function. When this is done the direction of search remains diagonal, but all the nodes are moved either towards the g-axis if $\alpha < 1$, or away from the g-axis if $\alpha > 1$. Thus the above two forms point out the two ways in which a change in an evaluation function can be viewed, either as a change in the direction of search or as a change in the position of the nodes.

4. Completeness, Admissibility, and Optimality

In this section we discuss the completeness, admissibility, and optimality of search strategies, which use $f = (1-\omega)g + \omega h$ for $\omega \varepsilon [0,1]$. For each of these properties we state a theorem which gives the subinterval of $[0,1]$ in which the given property holds, and justify the theorem in terms of the coordinate representation of the search process. We also provide simple proofs of the admissibility and optimality properties when the evaluation function is written as $f = g + \alpha h$. More complex proofs for $f = (1-\omega)g + \omega h$ which are similar to those given by Hart et al. and Pohl are included in the Appendix.

THEOREM 1. [Completeness] If $\omega \in [0,1)$, then a search strategy is complete for all δ-graphs.

The direction of search for $\omega = 1$, the only value of ω for which the search is incomplete, is parallel to the g-axis as indicated in Fig. 2.a. If the direction of search is not parallel to the g-axis and there is a path from the start node s to a goal node t , then the search will eventually cover a region which includes all of the nodes on this path. But if the direction of search is parallel to the g-axis, there may be an infinite number of nodes in one of the infinite regions which is encountered by the search prior to the region which includes all of the nodes on the solution path. This may keep the search from finding the goal node. In terms of the graph itself, search with $\omega = 1$ is incomplete because the heuristic may indefinitely lead it down a path (or a set of paths) which does not contain a goal node. Of course, if there are only finitely many nodes in the search space, then a search strategy is complete for any ω .

If the heuristic is perfect then $g(n_i) + h(n_i) = g(t)$ for all nodes on a minimal solution path $s=n_0,\ldots,n_k=t$, and all n_i lie on the diagonal of the minimal goal node t (see Fig. 2.b). If the heuristic is not perfect, but does satisfy the lower bound condition, then the nodes $n_0,\ldots,n_k$ are all pulled to the left and lie in the triangular region bounded by the two axes and the diagonal (including the boundary). Thus, diagonal, upper diagonal, and any search with $\omega \le \frac{1}{2}$ (see Fig. 2.c) will have expanded $n_0,\ldots,n_{k-1}$ by the time search reaches the location of the minimal goal t , and therefore t will be found before any other goal node. However, if $\omega > \frac{1}{2}$ (see Fig. 2.d) then the search strategy may not have expanded $s=n_0,\ldots,n_{k-1}$ by the time search reaches t , and may expand the nodes $s=n'_0,\ldots,n'_\ell=t'$, where t' is

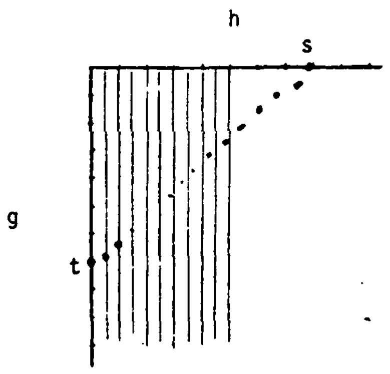


Fig. 2.a With ω=1, search may never cover the region which includes all of the nodes on a solution path.

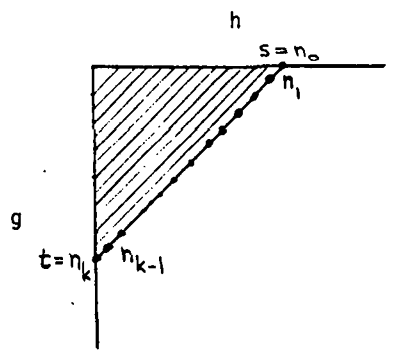


Fig. 2.b Diagonal search with a perfect heuristic. Only nodes on a minimal solution path are expanded.

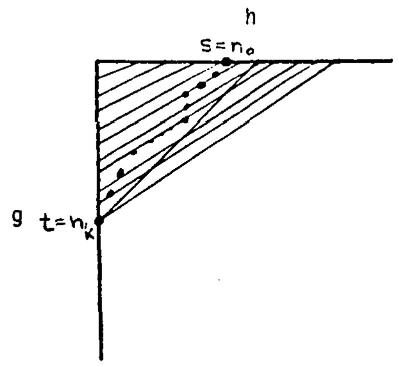


Fig. 2.c Search with ω<1/2 and a heuristic which satisfies the lower bound condition.

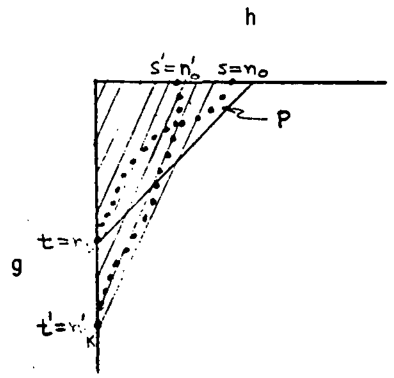


Fig. 2.d Search with ω>1/2 and a heuristic which satisfies the lower bound condition. The nodes along the subpath labeled P will prevent the minimal goal node t from being found.

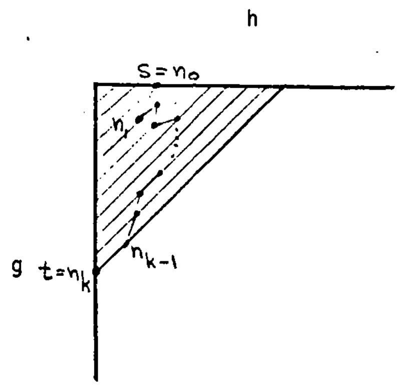


Fig. 2.e Diagonal search with a consistent heuristic. The search never has to backtrack to an earlier diagonal.

Fig. 2. Illustrations of different search strategies with various conditions on the heuristic. Although not explicitly shown in the figures, all nodes which lie in a region of search and have ancestors forming a path back to the start node which also lie in the region of search are expanded.

a nonminimal goal, before it expands the remaining nodes on the path $s=n_0,\ldots,n_k=t$. Thus search with $\omega > \frac{1}{2}$ may find a nonminimal solution, whereas search with $\omega \le \frac{1}{2}$ is admissible.

THEOREM 2. [Admissibility] If the heuristic satisfies the lower bound condition then a search strategy with $\omega \le \frac{1}{2}$ is admissible for all δ-graphs.

As pointed out by Pohl [1969] and Kowalski [1970] the admissibility property can easily be shown when the evaluation function is written as $f = g + \alpha h$. We know that $f = g + h$ is admissible if h satisfies the lower bound condition. The heuristic αh for $\alpha \le 1$ satisfies the lower bound condition whenever h satisfies the condition, and thus search with $f = g + \alpha h$ for $\alpha \le 1$ is admissible if h satisfies the lower bound condition.

The optimality theorem tells us that for all of the values of ω which give admissible searches, the use of better heuristics will result in improved searches.

THEOREM 3. [Optimality] Let $\omega \in [0,\frac{1}{2}]$, h_1 be consistent, and $h_2 < h_1 \le h_p$, where h_p is the perfect heuristic. Then search with h_2 expands every node expanded by search with h_1 for all δ-graphs that contain a minimal solution.

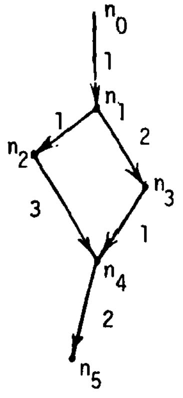
Consider first the justification for the case of $\omega = \frac{1}{2}$ (diagonal search). If the heuristic is consistent then all the nodes which precede n in the graph lie on an earlier diagonal than the diagonal on which n lies. Diagonal search then moves from diagonal to diagonal, and never has to backtrack to an earlier diagonal to expand a node (see Fig. 2.e). In particular, all the nodes in the triangular region will be expanded when t is found. Under these conditions search with an evaluation function which uses a better heuristic h_1 expands no more nodes than search with an evaluation function which uses a poorer heuristic h_2 . This can be understood most easily if we think in terms of the nodes located as they would be for a perfect heuristic, and of the movement which results from using an estimate to the perfect heuristic. The poorer heuristic h_2 pulls more nodes inside the triangular region than h_1, and hence search with $g + h_1$ expands fewer nodes than search with $g + h_2$.

For $\omega < \frac{1}{2}$ the reason for optimality is the same. The poorer heuristic pulls more nodes inside the triangular region than the better heuristic, only now the triangular region in question is as shown in Fig. 2.c.

Although not noted in the literature, the extension of the optimality theorem of Hart et al. can also easily be shown when the evaluation function is written $f = g + \alpha h$. Let h be a consistent heuristic and $\bar{h} = \alpha h$. Then since $\alpha \leq 1$ and h is consistent, $\bar{h}(n) - \bar{h}(n') = \alpha[h(n) - h(n')] \leq h(n) - h(n') \leq k(n,n')$, where $k(n,n')$ is the cost from n to n'. Hence $\bar{h}$ is consistent and $f = g + \alpha h$ for $\alpha \in [0,1]$ (or equivalently, $f = (1-\omega)g + \omega h$ for $\omega \in [0,\frac{1}{2}]$) is optimal.

If the hypothesis of the optimality theorem is changed from $h_2 < h_1 \leq h$ to $h_2 \leq h_1 \leq h$, then it is possible that search with $g + h_1$ may not expand a node n which lies on the minimal cost diagonal and is expanded by search with $g + h_2$. This is because a search strategy resolves ties arbitrarily, and search with the poorer heuristic h_2 may choose to expand a node n which is tied for merit with a minimal goal node, while search with the better heuristic h_1 chooses not to expand n. Thus, if $h_2 \leq h_1 \leq h$ then search with h_2 expands all the nodes expanded by search with h_1 except possibly for a set of nodes which have the same merit as a minimal goal node.

Fig. 3 shows that neither the admissibility nor the optimality property hold for $\omega \in (\frac{1}{2},1]$. It also shows that for $\omega \in [0,\frac{1}{2})$, search with a perfect heuristic may expand a node off of a minimal solution path [Pohl, 1969, 1970]. Thus in the interval $[0,\frac{1}{2}]$, as better heuristics are used better searches result; although except for $\omega = \frac{1}{2}$, the best heuristic (a perfect heuristic) does not result in the best solution (expanding only nodes on a minimal solution



$h_2(n_0) = 5$	$h_1(n_0) = 5$	$h_p(n_0) = 6$
$h_2(n_1) = 4$	$h_1(n_1) = 4$	$h_p(n_1) = 5$
$h_2(n_2) = 2$	$h_1(n_2) = 4$	$h_p(n_2) = 5$
$h_2(n_3) = 3$	$h_1(n_3) = 3$	$h_p(n_3) = 3$
$h_2(n_4) = 2$	$h_1(n_4) = 2$	$h_p(n_4) = 2$
$h_2(n_5) = 0$	$h_1(n_5) = 0$	$h_p(n_5) = 0$

n_0 = start

n_5 = goal

Fig. 3.a
The values of the perfect heuristic h_p, and the two approximations h_1 and h_2

Node	Merit	Order of Expansion
n_0:	$\frac{1}{4}0 + \frac{3}{4}5 = 3\frac{3}{4}$	(1)
n_1:	$\frac{1}{4}1 + \frac{3}{4}4 = 3\frac{1}{4}$	(2)
n_2:	$\frac{1}{4}2 + \frac{3}{4}2 = 2$	(3)
n_3:	$\frac{1}{4}3 + \frac{3}{4}3 = 3$	
n_4:	$\frac{1}{4}5 + \frac{3}{4}2 = 2\frac{3}{4}$	(4)
n_5:	$\frac{1}{4}7 + \frac{3}{4}0 = 1\frac{3}{4}$	(5)

Fig. 3.b
Search with $f = \frac{1}{4}g + \frac{3}{4}h_1$

Node	Merit	Order of Expansion
n_0:	$\frac{1}{4}0 + \frac{3}{4}5 = 3\frac{3}{4}$	(1)
n_1:	$\frac{1}{4}1 + \frac{3}{4}4 = 3\frac{1}{4}$	(2)
n_2:	$\frac{1}{4}2 + \frac{3}{4}4 = 3\frac{1}{2}$	
n_3:	$\frac{1}{4}3 + \frac{3}{4}3 = 3$	(3)
n_4:	$\frac{1}{4}4 + \frac{3}{4}2 = 2\frac{1}{2}$	(4)
n_5:	$\frac{1}{4}6 + \frac{3}{4}0 = 1\frac{1}{2}$	(5)

Fig. 3.c
Search with $f = \frac{1}{4}g + \frac{3}{4}h_2$

Node	Merit	Order of Expansion
n_0:	$\frac{3}{4}0 + \frac{1}{4}6 = 1\frac{1}{2}$	(1)
n_1:	$\frac{3}{4}1 + \frac{1}{4}5 = 2$	(2)
n_2:	$\frac{3}{4}2 + \frac{1}{4}5 = 2\frac{3}{4}$	(4)
n_3:	$\frac{3}{4}3 + \frac{1}{4}3 = 3\frac{1}{4}$	(3)
n_4:	$\frac{3}{4}4 + \frac{1}{4}2 = 3\frac{1}{2}$	(5)
n_5:	$\frac{3}{4}6 + \frac{1}{4}0 = 4$	(6)

Fig. 3.d
Search with $f = \frac{3}{4}g + \frac{1}{4}h_p$

Fig. 3 Examples using the graph shown in Fig. 1.a.

1) Fig. 3.b shows that a search strategy is not admissible for $\omega \in (1/2,1]$. The solution n_0,n_1,n_3,n_4,n_5 has cost 6 and is minimal, but search with $f = \frac{1}{4}g + \frac{3}{4}h_1$ finds the solution n_0,n_1,n_2,n_4,n_5 which is nonminimal.

2) Fig. 3.b and 3.c show that a search strategy is not optimal for $\omega \in (1/2,1]$. $h_1 \leq h_2 \leq h_p$, but for $\omega = 3/4$, search with h_2 expands the node n_3 which is not expanded by search with h_1.

3) Fig. 3.d shows that a search strategy with a perfect heuristic and $\omega \in [0,1/2)$ may expand a node (in this case the node n_2) which is off of a minimal solution path.

path). On the other hand, for $\omega \epsilon [\frac{1}{2},1]$ search with a perfect heuristic expands only nodes on a minimal solution path, but the optimality theorem is not true in $(\frac{1}{2},1]$. The only value of ω for which both the optimality theorem and the admissibility theorem are true, and search with a perfect heuristic expands only nodes on a minimal solution path is $\omega = \frac{1}{2}$.

In many problem domains one is not interested in finding a minimal solution to the problem, but in finding any solution using a minimal amount of resources. In these instances one would certainly not choose $\omega \epsilon [0,\frac{1}{2}]$, since admissibility is not required. In fact one may choose $\omega = 1$ since completeness, the assurance that every problem which has a solution will be solved given sufficient resources, may not be a desired property. However not including a cost component is not always advisable as shown by Pohl [1970]. He constructed an example where the heuristic deliberately led the search away from the goal, and showed that for this example search with $f = h$ expands more nodes than search with $f = g + h$.

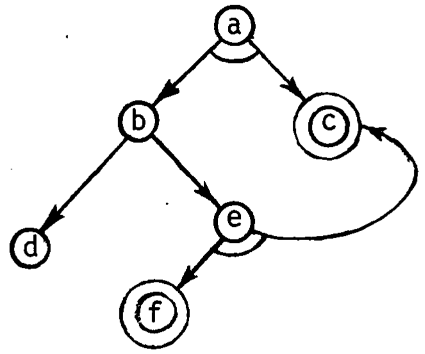
Although there are instances where $\omega \epsilon [\frac{1}{2},1]$ is best, if we have reasonable confidence in the heuristic, $\omega \epsilon [0,\frac{1}{2})$ should not be chosen, as the following corollary of the optimality theorem shows.

COROLLARY If h is consistent, then search with $f = (1-\omega)g + \omega h$ for $\omega < \frac{1}{2}$ expands all the nodes expanded by search with $f = \frac{1}{2}g + \frac{1}{2}h$.

Search with $\omega = \frac{1}{2}$ is better than search with $\omega < \frac{1}{2}$ because, as can be seen from Fig. 2.c, the area covered by the former in finding a minimal solution is a subset of that covered by the latter.

5. Problem-Solving Using the AND/OR Graph Model

AND/OR graphs are generalizations of directed graphs in that successors can be either OR-nodes or AND-nodes. If all successors are OR-nodes the AND/OR graph reduces to a directed graph. AND-successors are denoted by a bar connecting the arcs and usually mean some sort of dependence among the successor nodes. Corresponding to a solution path in a directed graph is a *solution graph* in an AND/OR graph. A solution graph is a subgraph with the following three properties: the root node is in the subgraph, if one AND-successor of a node is in the subgraph then all AND-successors of that node are in the subgraph, and all tip nodes are solved. In the following example (double circles denote solved nodes) a solution graph consists of the entire graph with the exception of the node d.



Costs are assigned to each arc of an AND/OR graph, and the cost of a solution graph is the sum of the costs of the arcs which are part of the solution graph.

AND/OR graphs can be used as a model of the problem space of *problem-reduction representations* [Nilsson, 1971]. In a problem-reduction representation the original problem (and recursively each subproblem) is divided into subproblems, where a Boolean relation applies between the solutions of the subproblems and the solution of the original problem. Problems are associated with nodes, and if all of the subproblems must be solved in order for the original problem to be solved, the nodes which represent these subproblems are AND-successors, otherwise they are OR_successors. Primitives subproblems, ones whose solution is known, are associated with solved nodes.

Many heuristic procedures for searching AND/OR graphs [Amarel, 1967; Slagle & Dixon, 1969; Nilsson, 1969; Nilsson, 1971] were developed prior to the work of Chang and Slagle [1971]. These procedures were structurally different from the best-bud procedures for searching directed graphs because they contained sections which labeled nodes solved or unsolved, and backed up the solved nodes to determine whether or not a solution graph had been found. A back up procedure is not necessary for searching a directed graph because a solution is a path and not a graph. The work of Chang and Slagle showed that a back up procedure was not necessary for searching AND/OR graphs either, and provided a unified approach to heuristically searching directed graphs and AND/OR graphs.

The best-bud approach to searching AND/OR graphs is based on the following method of formulating an AND/OR graph as directed graph. The method defines the set of nodes in the AND/OR graph which must be solved in order for a solution graph to exist as a node in the directed graph. Initially the root node of the directed graph is set equal to the root node of the AND/OR graph. The directed graph which corresponds to the above AND/OR graph is:

$$\begin{tikzcd}
 & A \arrow[d] & \\
 & BC \arrow[dl] \arrow[dr] & \\
DC & & EC \arrow[d] \\
 & & FC
\end{tikzcd}$$

FC is a goal node since the nodes f and c are the solved tip nodes of a solution graph.

Placing a merit ordering on the set of nodes in the above directed graph is equivalent to placing a merit ordering on the power set of the set of nodes of the AND/OR graph. Chang and Slagle accomplish this by associating with each node n_i of the AND/OR graph a statement N_i that the problem represented by that node is solved. Formulas in propositional logic are used to indicate the relationship between a node and its successors. The sequence of formulas which represent the above AND/OR graph is $A = BC = (D \vee E)C = DC \vee EC = DC \vee FC$. Conjunctions of the N_i are called *implicants.* An implicant corresponds to a node in the directed graph formulation of an AND/OR graph. Thus, placing a merit ordering on the set of implicants of an AND/OR graph essentially places a merit ordering on the power set of the set of nodes of the AND/OR graph, or on the set of nodes of the directed graph formulation of the AND/OR graph.

Chang and Slagle give a best-bud search strategy for AND/OR graphs which is based on defining a merit ordering on the set of implicants. The search strategy is equivalent to the best-bud approach to searching the directed graph formulation of an AND/OR grpah. They prove that it, like the best-bud procedure for searching directed graphs, is admissible for heuristics which satisfy the lower bound condition and optimal for consistent heuristics.

The completeness, admissibility, and optimality properties of search with the evaluation function $f = (1-\omega)g + \omega h$ for the appropriate range of ω and the appropriate conditions on h apply to AND/OR grphs as well as directed graphs. This follows directly from the fact that the search strategy for AND/OR graphs is logically equivalent to the search strategy for directed graphs applied to the directed graph formulation of an AND/OR graph.

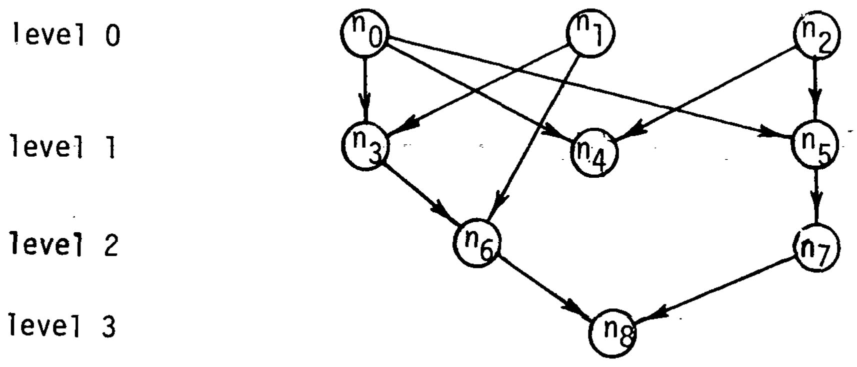
6. Problem-Solving Using the Theorem-Proving Graph Model

Some state-space representations have operators which require more than one state as input. The theorem proving problem is one such multiple-input operator state-space problem. In the state-space representation of the theorem proving problem clauses are states (the input clauses are the initial states and the empty clause is the goal state), factoring is a single-input operator and resolution is a multiple-input operator. A solution is a deduction of the empty clause from the set of input clauses.

In considering the problem of developing search strategies for theorem proving, Kowalski [1970] has generalized best-bud search strategies from graphs which model the state-space of representations with single-input operators to those which model the state-space of representations with multiple-input operators. To obtain graphs which model the state-space of representations with multiple-input operators Kowalski extends the tree representation of ordinary graphs (distinct paths to a node are represented by distinct nodes) to a directed, acyclic graph, which he calls a *theorem-proving graph*. For example, if $\{s_0,s_1,s_2\}$ are initial states, and if γ_i; for $1 \le i \le 6$ are successor operators such that

$$\begin{aligned}
\gamma_1(\{n_0,n_1\}) &= \{n_3\} \\
\gamma_2(\{n_0,n_2\}) &= \{n_4\} \\
\gamma_3(\{n_0,n_2\}) &= \{n_5\} \\
\gamma_4(\{n_1,n_3\}) &= \{n_6\} \\
\gamma_5(\{n_5\}) &= \{n_7\} \\
\gamma_6(\{n_6,n_7\}) &= \{n_8\}
\end{aligned}$$

then the theorem-proving graph for this portion of the state-space is:



In the introduction we described a search strategy as one which determines which node to expand at each stage of the search. One could equally well think of a search strategy as one which determines which node to generate at each stage of the search. The first approach was taken by Hart et al. [1968] and the second approach was taken by Pohl [1970]. The two approaches are illustrated for directed graphs in Fig. 4. The first approach results in viewing the application of an operator in the normal direction, from the domain to the range; while the second approach results in viewing the application of an operator in the inverse direction, from the range to the domain. For graphs which model representations with single-input operators one approach is as good as the other.

For the case of multiple-input operators, it is easier to think in terms of which node to generate than which set of nodes to expand, and this is the approach taken by Kowalski. This allows for the merit ordering to be defined on the set of nodes, rather than on the power set of nodes. A search strategy for theorem-proving graphs, like one for directed graphs, determines which node to generate at each stage. An efficient implementation of a search strategy for a multiple-input operator state-space problem must be able to apply operators in the inverse direction without exhaustively checking the power set of the nodes which have been generated. This is possible in the theorem proving problem when the g-component of a clause is its level and the h-component is its length, and when the process of performing a merge is assigned on explicit cost.

Kowalski calls a merit ordering a *δ-finite merit ordering* if there are finitely many nodes which have better or equal merit than any given node, and proves that any search strategy for theorem-proving graphs which generates nodes according to a δ-finite merit ordering is complete. A δ-finite merit

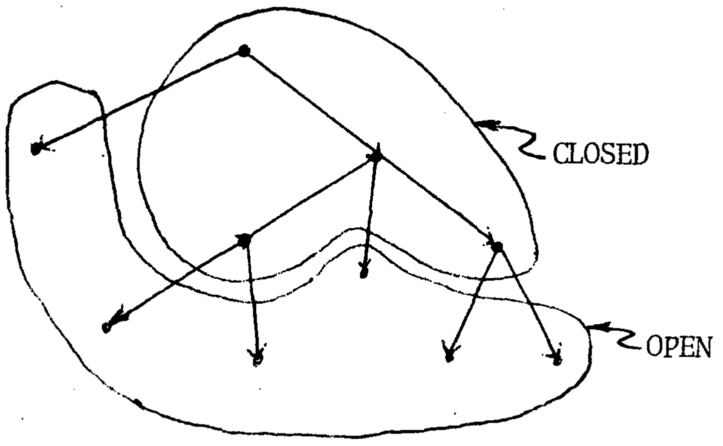


Fig. 4.a. The approach to a search strategy which divides the nodes into those which have been expanded, CLOSED; and those which are candidates for expansion, OPEN. The nodes in OPEN are the tip nodes of the graph, those in CLOSED are the nontip nodes.

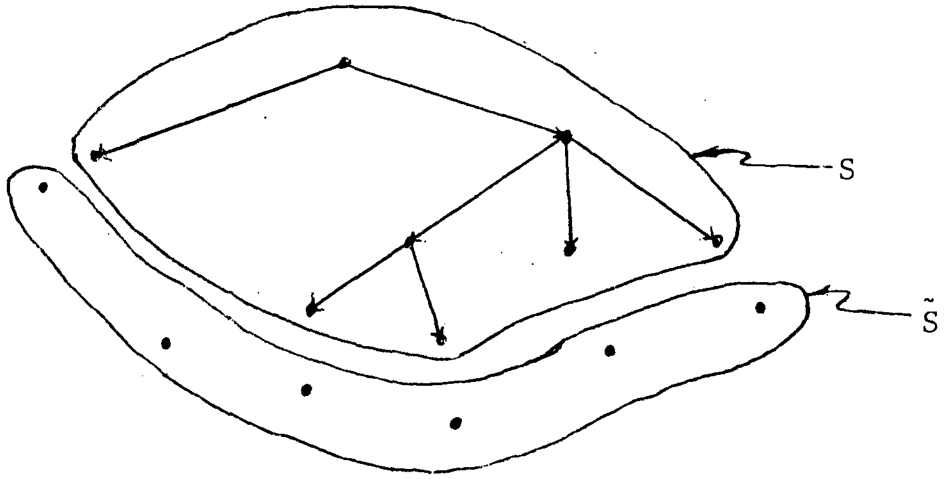


Fig. 4.b. The approach to a search strategy which divides the nodes into those which have been generated, S; and those which are candidates for generation, S .

Fig. 4. Illustrations of two approaches to a search strategy for directed graphs.

ordering is a more general condition than a δ-graph which was used by Hart et al. The former allows finitely many arcs to have zero cost, while the later specifies that all arc costs are greater than or equal to δ which is greater than zero. Kowalski also proves that with appropriate conditions on the heuristic function, diagonal and upper diagonal search strategies are admissible, and that upward diagonal search strategies are an optimal subclass of diagonal search strategies.

The completeness, admissiblity, and optimality of search with $f \doteq (1-\omega)g + \omega h$ for appropriate values of ω also apply to the theorem-proving graph model. At each stage of the process of searching a theorem-proving graph the merit ordering defined by f determines which of the nodes which are one step away from those nodes which have been generated at previous stages will be generated at the current stage. This is analogous to the directed graph case except that in searching theorem-proving graphs a node n can never be completely expanded. Some time later in the search process the node n may be needed as an input to the operator γ because a node n' was generated such that γ is applicable to a set B , where $\{n,n'\} \subseteq B$. This difference however does not affect the above properties.

Multiple-Output Operators

It is natural to ask about representations with multiple-output operators, and of strategies for searching graphs which model them. The concept of a multiple-output operator must involve some notion of dependence among the successor states, because if they are independent then the multiple-output operator can be replaced by a set of single-output operators. For example, if $\gamma(\{n_0\}) = \{n_1,n_2\}$, with n_1 and n_2 independent, then γ can be replaced by γ_1 and γ_2 , where $\gamma_1(\{n_0\}) = \{n_1\}$ and $\gamma_2(\{n_1\}) = \{n_2\}$.

One type of dependency among successor states occurs when sequences of operators must be found which transform all of the successors into a goal state. Consider, for example, a representation of the theorem proving problem which defines a state not as a clause but as a literal of a clause. In this representation all the literals of some clause must be resolved away in order for a solution to be found. Resolving away all the literals of some clause is like finding paths in a graph from all successor nodes to goal nodes. This, of course, is the case for AND-successors in AND/OR graphs. Thus because the concept of multiple-output operators must involve some notion of dependence among successors, an AND/OR graph can be used to model the search of such representations. Approaching this multiple-output operator/dependent successor relationship from a different direction, leads to the observation that the concept of multiple-output operators can be used to define the operators of a problem-reduction representation. The use of both multiple-input and multiple-output operators provides a formalism which includes both problem-reduction and state-space representation.

Another type of dependency among successor states occurs when the application of an operator to one of the successors in some way affects the other successors. The problem here is analogous to the problem of side affects in programming languages. If it is true that the application of an operator to one successor affects the other successors, then the representation may as well be redefined to make all of the successors into a single state. Consider the above example of the theorem proving problem where states are literals. All the literals of a clause which have a common variable are dependent, since a substitution of a term for that variable must be made throughout the clause. The bookkeeping involved in keeping track of these dependent states would suggest that each set of dependent states be considered as one state. This does not preclude operating on part of a state and storing the affects of this operation on the other parts of the state for subsequent use. Inference systems like SL-resolution [Kowalski & Kuehner, 1971] define state-space representations of the theorem proving problem with the above property.

Let us make one final comment about multiple-output operator representations. Unlike single-output operator representations, where at each stage a search strategy simply decides which single node to generate, multiple-output operator representations require search strategies to determine which set of nodes to generate. Thus the merit ordering must be defined on the power set of the nodes of the graph. As we have seen, this is what the procedure of Chang and Slagle for searching AND/OR graphs does.

7. Summary

This paper has discussed three types of graphs which are used to model the search process, and the completeness, admissiblity, and optimality properties of strategies for searching them. Each of the graphs model the search process for a different type of problem representation: directed graphs are used for state-space representations with single-input operators, AND/OR graphs are used for problem reduction representations, and theorem-proving graphs are used for state-space representations with multiple-input operators. It was seen that heuristic strategies for searching graph structures for modeling single-input and multiple-input operator state-space representations place a merit ordering on the set of nodes; while strategies for searching the graph structure for problem-reduction representations, which can be thought of as multiple-output operator representations, place a merit-ordering on the power set of nodes. The graph structures themselves are quite different, but the structures of the heuristic strategies for searching them differ only in the set on which the merit ordering is defined. More particularly, the properties of completeness, admissibility, and optimality of search with

$f = (1-\omega)g + \omega h$ for appropriate values of ω and appropriate conditions on h apply to all three models.

A summary of the theoretical properties of searching with the evaluation function $f = (1-\omega)g + \omega h$ is given in Fig. 5. The point, or the subinterval of $[0,1]$, over which a given property is true is indicated in the graph, with a reference and the graph model for which the property was shown appearing in the margin. The admissilbity and optimality for $\omega \varepsilon [0,\frac{1}{2}]$ for directed graphs, AND/OR graphs, and theorem-proving graphs reference the author who originally worked with procedures for searching these models. Pohl [1969] and Kowalski [1970] say that search with $\omega \varepsilon [0,\frac{1}{2})$ is admissible for directed graphs and theorem-proving graphs respectively. This author could find no mention of the optimality for directed graphs and theorem-proving graphs, and the admissilbity and optimality for AND/OR graphs in the references. The extension of these properties to the interval $[0,\frac{1}{2}]$ is looked upon mostly as an observation, and thus Fig. 5 refers to the original references.

Fig. 5 does not refer to all of the theoretical results about heuristic search. The use of $f = (1-\omega)g + \omega h$ is only one way to define a merit ordering, and any result which does not define the merit ordering in this way (such as the theorem proved by Kowalski that search with any δ-finite merit ordering is complete) cannot be shown on the graph.

A theory of heristic search is beginning to form, and we have discussed some of the initial steps in the formation of that part of the theory which concerns the use of graphs to model the search process. The theory tells us that completeness, admissibility, and optimality are properties which are possessed by heuristic search strategies which satisfy certain conditions. These results do not imply that these properties are always to be desired.

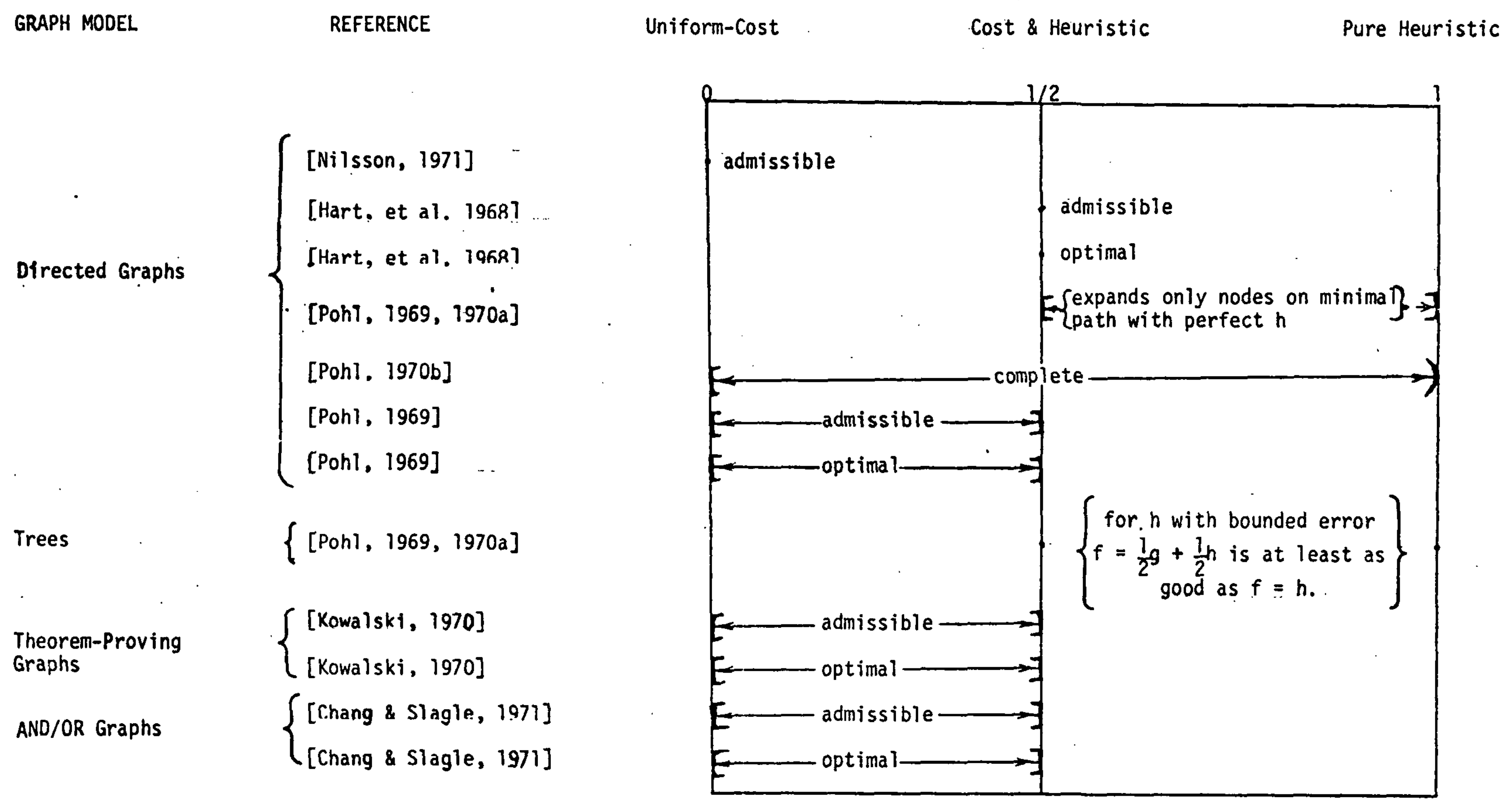


Figure 5. A summary of the properties of heuristic search strategies which use $f = (1-\omega)g + \omega h$ to define a merit ordering. Admissibility requires the heuristic to satisfy the lower bound condition and optimality requires a consistent heuristic. Completeness places no restrictions on the heuristic.

Admissible search strategies tend to be more conservative in that they stay nearer to the start node — they take less chances. They stay nearer to the start node and take fewer chances than inadmissible strategies because they guarantee finding a minimal solution before any other solution. In some problem environments one is not interested in obtaining a minimal solution. In other environments minimal solutions are considered nice, but finding a solution quickly is more important. In these cases one would, of course, not necessarily employ an admissible strategy. The same thing is true for completeness only to a lesser extent. Complete strategies are more conservative than incomplete strategies because they guarantee finding a solution if one exists. They cannot continue the search indefinitely down a path (or a number of paths simulataneously) because the goal node may not lie along that path. The relative conservativeness of admissible and complete strategies can be seen from the amount of emphasis that they place on the heuristic component. Admissible strategies cannot place more emphasis on the heuristic component than the cost component, whereas all that is necessary in order for a strategy to be complete is that there be a nonzero cost component. But even though search strategies with the completeness, admissibility, and optimality properties are not always to be desired, the characterization of strategies which do possess them is important. The characterization is important because there are problem-solving environments where they are a concern, and where they are not, the characterization can aid in the design of powerful heuristic search strategies.

APPENDIX

The appendix contains proofs of the completeness, admissibility, and optimality theorems for search strategies which use the evaluation function $f = (1-\omega)g + \omega h$. We call such a search strategy A^*_ω . The proofs are similar to those of Hart et al. [1968] and Pohl [1970]. $\hat{g}$ and $\hat{h}$ are approximations to g and h .

THEOREM 1. [Completeness] If $\omega \in [0,1)$, then A^*_ω is complete for all δ-graphs.

PROOF. Let n be an arbitrary node. We will show that only finitely many nodes have better merit than n . Completeness follows because if there is a solution then all of the nodes on a solution path have the property that finitely many nodes have better merit, and thus the goal node will eventually be expanded.

Let B be the set of nodes which have better merit than n , and let $n' \in B$. Then since $\hat{h}(n) \geq 0$, and n' has better merit than n ,

$$(1-\omega)\hat{g}(n') \leq (1-\omega)\hat{g}(n') + \omega\hat{h}(n') = \hat{f}(n') \leq \hat{f}(n) \ .$$

Let d be the number of nodes along the longest path from the start node s to n' . Then $d\delta \leq \hat{g}(n')$, and

$$(1-\omega)d\delta \leq (1-\omega)\hat{g}(n') \leq \hat{f}(n) \quad \text{or} \quad d \leq \frac{\hat{f}(n)}{(1-\omega)\delta} \quad \text{for} \quad \omega \neq 1 \ .$$

Hence, if $n' \in B$, it must be within $\frac{\hat{f}(n)}{(1-\omega)\delta}$ nodes of s . Since each node of the graph has a finite number of successors, there are finitely many nodes within $\frac{\hat{f}(n)}{(1-\omega)\delta}$ nodes of s , and the set B is finite. Thus there are only finitely many nodes which have better merit than n , and A^* for $\omega \in [0,1)$ is complete.

THEOREM 2. [Admissiblity] If $\hat{h}(n) \leq h(n)$ for all n and $\omega \in [0,\frac{1}{2}]$, then A^*_ω is admissible for all δ-graphs.

PROOF. Since Theorem 1 showed that A^*_ω is complete, what remains to be shown is that if there exists a minimal solution, then A^*_ω will find it. Assume that there is a minimal solution. Let n be the node expanded by A^*_ω just prior to termination, and let m be any other goal node. Suppose, on the contrary, that A^*_ω has not found a minimal solution, i.e., $g(m) < g(n)$. Let $s=m_0,m_1,\ldots,m_j,\ldots,m_k=m$ be a minimal solution path to m, and m_j be the last node on this path which has been generated. The following chain of inequalities shows that the node m_j had better merit than n.

$$\begin{aligned}
\hat{f}(m_j) &= (1-\omega)\hat{g}(m_j) + \omega\hat{h}(m_j) & & \\
&= (1-\omega)g(m_j) + \omega\hat{h}(m_j) & & [\hat{g}(m_j) = g(m_j), \text{ since a minimal path to } m_j \text{ has been found}] \\
&\le (1-\omega)g(m_j) + \omega h(m_j) & & [\text{from the lower bound condition}] \\
&\le (1-\omega)[g(m_j) + h(m_j)] & & [\text{since } \omega \le 1/2] \\
&= (1-\omega)g(m) & & [g(m) = g(m_j) + h(m_j), \text{ since g and h are exact}] \\
&< (1-\omega)g(n) & & [g(m) < g(n) \text{ is assumed}] \\
&\le (1-\omega)\hat{g}(n) & & \\
&= \hat{f}(n) .
\end{aligned}$$

This is a contradiction since A^*_ω selected n over m_j when $\hat{f}(m_j) < \hat{f}(n)$. Hence A^*_ω is admissible.

Before proving optimality we first prove two lemmas which are analogous to those found in Hart, et al.

If the evaluation function is $f = g + h$, then the f-value of all nodes on a minimal solution path is the actual cost of a minimal solution, and if the evaluation funciton is $f = \frac{1}{2}g + \frac{1}{2}h$ then the f-value of all nodes on a minimal solution path is one half of the actual cost of a minimal solution. However, when the search is directed by $f = (1-\omega)g + \omega h$ then the f-value of nodes on a minimal solution path varies between the $\min\{\omega,1-\omega\}$ times the actual cost of a minimal solution and the $\max\{\omega,1-\omega\}$ times the actual cost of a minimal solution. The actual cost of a minimal solution is given by $g(t)$ where t is a goal node to which there is a minimal solution path. The algorithm A^* never expands a node whose merit is larger than $f(s)$. A bound on the worst merit of a node expanded by A^*_ω is given in the following lemma.

LEMMA 1. Let $\hat{h}$ satisfy the lower bound condition, t be a goal node to which there is a minimal solution path, and $\gamma = \max\{1-\omega,\omega\}$. If A^*_ω expands a node n , then $\hat{f}(n) \le \gamma g(t)$.

PROOF. Let $s=n_0,n_1,\ldots,n_j,\ldots,n_k=t$ be a minimal solution path, and suppose that A^*_ω has generated but not yet expanded n_j . Assume, on the contrary, that A^*_ω expands a node n , where $\hat{f}(n) > \gamma g(t)$. Then

$$\begin{aligned}\hat{f}(n_j) &= (1-\omega)\hat{g}(n_j) + \omega h(n_j)\\ &= (1-\omega)g(n_j) + \omega\hat{h}(n_j) \quad [\hat{g}(n_j) = g(n_j)\text{ , since } A^*_\omega \text{ has found a minimal path to } n_j]\\ &\le (1-\omega)g(n_j) + \omega h(n_j) \quad [\text{from the lower bound condition}].\end{aligned}$$

We now divide the proof into two cases.

Case 1. $(1-\omega) \ge \omega$

$$\begin{aligned}f(n_j) &\le (1-\omega)g(n_j) + \omega h(n_j)\\ &\le (1-\omega)[g(n_j) + h(n_j)]\\ &= (1-\omega)g(t) \qquad [\text{since g and h are exact}]\end{aligned}$$

Case 2. $(1-\omega) < \omega$

$$\begin{aligned}f(n_j) &\le (1-\omega)g(n_j) + \omega h(n_j)\\ &\le \omega[g(n_j) + h(n_j)]\\ &= \omega g(t) \qquad [\text{since g and h are exact}]\end{aligned}$$

Thus $\hat{f}(n_j) \le \gamma g(t)$, while $\hat{f}(n) > \gamma g(t)$. Therefore A^*_ω did not select the node with the best merit. Hence, if A^*_ω selects n , $\hat{f}(n) \le \gamma g(t)$. This completes the proof.

The following lemma shows that if $\omega \varepsilon [0,\frac{1}{2}]$ and the heuristic function satisfies the consistency condition, then A^*_ω does not expand a node before finding a minimal path to that node.

LEMMA 2. Let $\hat{h}$ be consistent and $\omega \varepsilon [0,\frac{1}{2}]$. If A^*_ω expands n, then $\hat{g}(n) = g(n)$.

PROOF. Assume, in the contrary, that $\hat{g}(n) > g(n)$. Then A^*_ω could not have found a minimal path to n because $\hat{g}(n) \neq g(n)$. Let m be the first node on a minimal path to n which A^*_ω has not expanded. The following sequence of inequalities leads to the conclusion that the node m has better merit than the node n.

$$\hat{g}(n) > g(n)$$

$$(1-\omega)\hat{g}(n) > (1-\omega)g(n)$$

$$= (1-\omega)[g(m) + k(m,n)] \qquad \text{[}k(m,n) \text{ is the cost of minimal path from } m \text{ to } n\text{]}$$

$$= (1-\omega)g(m) + (1-\omega)k(m,n)$$

$$\geq (1-\omega)g(m) + \omega k(m,n) \qquad \text{[since } \omega \leq 1/2\text{]}$$

$$= (1-\omega)\hat{g}(m) + \omega k(m,n) \qquad \hat{g}(m) = g(m)\text{, since a minimal path to } m \text{ has been found]}$$

$$\geq (1-\omega)\hat{g}(m) + \omega(\hat{h}(m) - \hat{h}(n)) \quad \text{[since } \hat{h} \text{ is consistent]}$$

Therefore, $(1-\omega)\hat{g}(n) + \omega\hat{h}(n) > (1-\omega)\hat{g}(m) + \omega\hat{h}(m)$

$$\hat{f}(n) > \hat{f}(m)$$

This is a contradiction because A^*_ω selected n when m had a better merit. Hence $\hat{g}(n) = g(n)$. This completes the proof of Lemma 2.

THEOREM 3. [Optimality] Let $\hat{h}_1$ be consistent, $\omega \varepsilon [0,\frac{1}{2}]$, and $\hat{h}_2(n) < \hat{h}_1(n) \leq h(n)$ for all n. Then A^*_ω using $\hat{f}_2 = \hat{g}_2 + \hat{h}_2$ expands every node that A^*_ω using $\hat{f}_1 = \hat{g}_1 + \hat{h}_1$ expands for all δ-graphs that have a minimal solution.

PROOF. Let $n_0, n_1, \ldots, n_i, \ldots$ be the nodes expanded by A^*_ω using $\hat{f}_1$. Assume there is a node expanded by A^*_ω using $\hat{f}_1$, but not by A^*_ω using $\hat{f}_2$, and let the first such node be n_i. Since A^*_ω using $\hat{f}_1$ expands n_i, we know that

$$\hat{f}_1(n_i) \le (1-\omega)g(t) \quad \text{[from Lemma 1, since } (1-\omega) = \max\{1-\omega,\omega\}]$$

$$(1-\omega)\hat{g}_1(n_i) + \omega\hat{h}_1(n_i) \le (1-\omega)g(t)$$

(1) $$(1-\omega)g(n_i) + \omega\hat{h}_1(n_i) \le (1-\omega)g(t) \quad [\hat{g}_1(n_i) = g(n_i) \text{ from Lemma 2}]$$

Since A^*_ω using $\hat{f}_2$ expands all the ancestors of n_i but does not expand n_i, and since it is admissible,

$$\hat{f}_2(n_i) \ge (1-\omega)g(t)$$

$$(1-\omega)\hat{g}_2(n_i) + \omega\hat{h}_2(n_i) \ge (1-\omega)g(t).$$

We are assuming that the node n_i is not expanded by A^*_ω using $\hat{f}_2$, but it is generated by A^*_ω using $\hat{f}_2$, and since $\omega \in [0,\frac{1}{2}]$ at some stage $\hat{g}_2(n_i) = g(n_i)$. Hence,

(2) $$(1-\omega)g(n_i) + \omega\hat{h}_2(n_i) \ge (1-\omega)g(t).$$

Together (1) and (2) imply,

$$(1-\omega)g(n_i) + \omega\hat{h}_1(n_i) \le (1-\omega)g(n_i) + \omega\hat{h}_2(n_i) .$$

Subtracting $(1-\omega)g(n_i)$ and dividing by ω gives,

(3) $$\hat{h}_1(n_i) \le \hat{h}_2(n_i) .$$

But $\hat{h}_2(n) < \hat{h}_1(n)$ for all n. Therefore, A^*_ω using $\hat{f}_2$ expands all of the nodes expanded by A^*_ω using $\hat{f}_1$. This completes the proof of the optimality theorem.

References

(1) Amarel, S. [1967]: "An Approach to Heuristic Problem-Solving and Theorem Proving in the Propositional Calculus," In J. Hart and S. Takasu (eds.). Systems and Computer Science, U. of Toronto Press, Toronto, 1967.

(2) Chang, C. L. and Slagle, J. R. [1971]: "An Admissible and Optimal Algorithm for Searching AND/OR Graphs," Artificial Intelligence Journal, vol. 2, no. 2, Fall 1971, pp. 117-128.

(3) Doran, J. E. and Michie, D. [1966]: "Experiments with the Graph Traverser Program," Proc. R. Soc. (A), vol. 294, pp. 235-259.

(4) Doran, J. E. [1967]: "An Approach Toward Automatic Problem-Solving," In: Collins, N. L. and Michie, D. (eds.). Machine Intelligence 1, Oliver & Boyd, Edinburgh, 1967, pp. 105-123.

(5) Ernst, G. W. and Newell, A. [1969]: GPS - A Case Study in Generality and Problem Solving, Academic Press, New York, 1969.

(6) Hart, P. N., Nilsson, N., and Raphael, B. [1968]: "A Formal Basis for the Heuristic Determination of a Minimum Cost Path," IEEE Trans. Sys. Sci. Cybernetics, vol. S.S.C-4, no. 2, July 1968, pp. 100-107.

(7) Kowalski, R. [1970]: "Search Strategies for Theorem-Proving," In: Meltzer, B. & Michie, D. (eds.). Machine Intelligence 5, American Elsevier, New York, 1970, pp. 181-201.

(8) Kowalski, R. and Kuehner, D. [1971]: "Linear Resolution with Selection Function," Artificial Intelligence Journal, vol. 2, no. 3/4, Winter 1971, pp. 221-260.

(9) Michie, D. [1970]: "Heuristic Search," Computer Journal, vol. 14, no. 1, pp. 96-102.

(10) Michie, D. [1971]: "Formation and Execution of Plans by Machine," In: Findler, N. V. & Meltzer, B. (eds). Artificial Intelligence and Heuristic Programming, American Elsevier, New York, 1971, pp. 101-124.

(11) Nilsson, N. J. [1969]: "Searching Problem-Solving and Game-Playing Trees for Minimal Cost Solutions," In: A.J.H. Morrell (ed.). Inf. Proc. 68, vol. 2, pp. 1556-1562, North-Holland Publ. Co., Amsterdam, 1969.

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

(13) Pohl, I. [1969]: "Bi-Directional and Heuristic Search in Path Problems," Ph.D. Thesis, Computer Science Center, Stanford Univ., May 1969.

(14) Pohl, I. [1970a]: "First Results on the Effect of Error in Heuristic Search," In: Meltzer, B. & Michie, D. (eds.). Machine Intelligence 5, American Elsevier, New York, 1970, pp. 219-236.

(15) Pohl, I. [1970b]: "Heuristic Search Viewed as Path Finding in a Graph," Artificial Intelligence Journal, vol. 1, no. 3, Fall 1970, pp. 193-204.

(16) Pohl, I. [1971]: "Bi-directional Search," In: Meltzer, B. & Michie, D. (eds.). Machine Intelligence 6, American Elsevier, New York, 1971, pp. 127-140.

(17) Sandewall, E. [1971]: "Heuristic Search: Concepts and Methods," In: Findler, N.V. & Meltzer, B. (eds.). Artificial Intelligence and Heuristic Programming, American Elsevier, New York, 1971, pp. 81-100.

(18) Simon, H.A. [1971]: "The Theory of Problem Solving," Proc. IFIP Congress 71, North Holland, Amsterdam, August 1971, Invited Papers, pp. 249-266.

(19) Slagle, J.R. and Dixon, J. [1969]: "Experiments with Some Programs that Search Game Trees," J.ACM, vol. 16, no. 2, pp. 189-207, April 1969.

(20) Slagle, J.R. [1971]: Artificial Intelligence: The Heuristic Programming Approach, McGraw-Hill, New York, 1971.