[5] J. E. Falk, "Lagrange multipliers and nonlinear programming," *J. Math. Anal. Appl.*, vol. 19, July 1967.
[6] O. L. Mangasarian and J. Ponstein, "Minimax and duality in nonlinear programming," *J. Math. Anal. Appl.*, vol. 11, pp. 504–518, 1965.
[7] J. Stoer, "Duality in nonlinear programming and the minimax theorem," *Numerische Mathematik*, vol. 5, pp. 371–379, 1963.
[8] R. T. Rockafellar, "Duality and stability in extremum problems involving convex functions," *Pacific J. Math.*, vol. 21, pp. 167–187, 1967.
[9] P. Wolfe, "A duality theorem for nonlinear programming," *Q. Appl. Math.*, vol. 19, pp. 239–244, 1961.
[10] R. T. Rockafellar, "Nonlinear programming," presented at the American Mathematical Society Summer Seminar on the Mathematics of the Decision Sciences, Stanford University, Stanford, Calif., July–August 1967.
[11] D. G. Luenberger, "Convex programming and duality in normal space," *Proc. IEEE Systems Science and Cybernetics Conf.* (Boston, Mass., October 11–13, 1967).
[12] J. M. Danskin, "The theory of max-min with applications," *J. SIAM*, vol. 14, pp. 641–665, July 1966.
[13] W. Fenchel, "Convex cones, sets, and functions," mimeographed notes, Princeton University, Princeton, N. J., September 1963.
[14] R. Fletcher and M. J. D. Powell, "A rapidly convergent descent method for minimization," *Computer J.*, vol. 6, p. 163, July 1963.
[15] L. S. Lasdon and A. D. Waren, "Mathematical programming for optimal design," *Electro-Technol.*, pp. 53–71, November 1967.
[16] J. B. Rosen, "The gradient projection method for nonlinear programming, pt. I, linear constraints," *J. SIAM*, vol. 8, pp. 181–217, 1960.
[17] R. Fletcher and C. M. Reeves, "Function minimization by conjugate gradients," *Computer J.*, vol. 7, July 1964.
[18] D. Goldfarb, "A conjugate gradient method for nonlinear programming," Ph.D. dissertation, Dept. of Chem. Engrg., Princeton University, Princeton, N. J., 1966.
[19] L. S. Lasdon, "A multi-level technique for optimization," Ph.D. dissertation, Systems Research Center, Case Institute of Technology, Cleveland, Ohio, Rept. SRC 50-C-64-19, 1964.
[20] L. S. Lasdon and J. D. Schoeffler, "A multi-level technique for optimization," Preprints, Joint Automatic Control Conf., Troy, N. Y., June 22–25, 1965, pp. 85–92.
[21] ——,"Decentralized plant control," *ISA Trans.*, vol. 5, pp. 175–183, April 1966.
[22] C. B. Brosilow and L. S. Lasdon, "A two level optimization technique for recycle processes," *1965 Proc. AICHE—Symp. on Application of Mathematical Models in Chemical Engineering Research, Design, and Production* (London, England).
[23] L. S. Lasdon, "Duality and decomposition in mathematical programming," Systems Research Center, Case Institute of Technology, Cleveland, Ohio, Rept. SRC 119-C-67-52, 1967.
[24] A. V. Fiacco and G. P. McCormick, *Sequential Unconstrained Minimization Techniques for Nonlinear Programming.* New York: Wiley, 1968.
[25] R. Fox and L. Schmit, "Advances in the integrated approach to structural synthesis," *J. Spacecraft and Rockets*, vol. 3, p. 858, June 1966.
[26] B. P. Dzielinski and R. E. Gomory, "Optimal programming of lot sizes, inventory, and labor allocations," *Management Sci.*, vol. 11, pp. 874–890, July 1965.
[27] J. E. Falk, "A relaxed interior approach to nonlinear programming," Research Analysis Corp., McLean, Va. RAC-TP-279, 1967.

A Formal Basis for the Heuristic Determination of Minimum Cost Paths

PETER E. HART, MEMBER, IEEE, NILS J. NILSSON, MEMBER, IEEE, AND BERTRAM RAPHAEL

***Abstract*—Although the problem of determining the minimum cost path through a graph arises naturally in a number of interesting applications, there has been no underlying theory to guide the development of efficient search procedures. Moreover, there is no adequate conceptual framework within which the various ad hoc search strategies proposed to date can be compared. This paper describes how heuristic information from the problem domain can be incorporated into a formal mathematical theory of graph searching and demonstrates an optimality property of a class of search strategies.**

I. INTRODUCTION

A. *The Problem of Finding Paths Through Graphs*

MANY PROBLEMS of engineering and scientific importance can be related to the general problem of finding a path through a graph. Examples of such problems include routing of telephone traffic, navigation through a maze, layout of printed circuit boards, and mechanical theorem-proving and problem-solving. These problems have usually been approached in one of two ways, which we shall call the *mathematical approach* and the *heuristic approach.*

1) The mathematical approach typically deals with the properties of abstract graphs and with algorithms that prescribe an orderly examination of nodes of a graph to establish a minimum cost path. For example, Pollock and Wiebenson[1] review several algorithms which are guaranteed to find such a path for any graph. Busacker and Saaty[2] also discuss several algorithms, one of which uses the concept of dynamic programming.[3] The mathematical approach is generally more concerned with the ultimate achievement of solutions than it is with the computational feasibility of the algorithms developed.

2) The heuristic approach typically uses special knowledge about the domain of the problem being represented by a graph to improve the computational efficiency of solutions to particular graph-searching problems. For example, Gelernter's[4] program used Euclidean diagrams to direct the search for geometric proofs. Samuel[5] and others have used ad hoc characteristics of particular games to reduce

Manuscript received November 24, 1967.
The authors are with the Artificial Intelligence Group of the Applied Physics Laboratory, Stanford Research Institute, Menlo Park, Calif.

the "look-ahead" effort in searching game trees. Procedures developed via the heuristic approach generally have not been able to guarantee that minimum cost solution paths will always be found.

This paper draws together the above two approaches by describing how information from a problem domain can be incorporated in a formal mathematical approach to a graph analysis problem. It also presents a general algorithm which prescribes how to use such information to find a minimum cost path through a graph. Finally, it proves, under mild assumptions, that this algorithm is optimal in the sense that it examines the smallest number of nodes necessary to guarantee a minimum cost solution.

The following is a typical illustration of the sort of problem to which our results are applicable. Imagine a set of cities with roads connecting certain pairs of them. Suppose we desire a technique for discovering a sequence of cities on the shortest route from a specified start to a specified goal city. Our algorithm prescribes how to use special knowledge—e.g., the knowledge that the shortest road route between any pair of cities cannot be less than the airline distance between them—in order to reduce the total number of cities that need to be considered.

First, we must make some preliminary statements and definitions about graphs and search algorithms.

B. Some Definitions About Graphs

A *graph* G is defined to be a set $\{n_i\}$ of elements called nodes and a set $\{e_{ij}\}$ of directed line segments called arcs. If e_{pq} is an element of the set $\{e_{ij}\}$, then we say that there is an *arc* from node n_p to node n_q and that n_q is a *successor* of n_p. We shall be concerned here with graphs whose arcs have *costs* associated with them. We shall represent the cost of arc e_{ij} by c_{ij}. (An arc from n_i to n_j does not imply the existence of an arc from n_j to n_i. If both arcs exist, in general $c_{ij} \neq c_{ji}$.) We shall consider only those graphs G for which there exists $\delta > 0$ such that the cost of every arc of G is greater than or equal to δ. Such graphs shall be called δ *graphs*.

In many problems of interest the graph is not specified explicitly as a set of nodes and arcs, but rather is specified implicitly by means of a set of source nodes $S \subset \{n_i\}$ and a successor operator Γ, defined on $\{n_i\}$, whose value for each n_i is a set of pairs $\{(n_j, c_{ij})\}$. In other words, applying Γ to node n_i yields all the successors n_j of n_i and the costs c_{ij} associated with the arcs from n_i to the various n_j. Application of Γ to the source nodes, to their successors, and so forth as long as new nodes can be generated results in an explicit specification of the graph thus defined. We shall assume throughout this paper that a graph G is always given in implicit form.

The subgraph G_n from any node n in $\{n_i\}$ is the graph defined implicitly by the single source node n and some Γ defined on $\{n_i\}$. We shall say that each node in G_n is *accessible* from n.

A *path* from n_1 to n_k is an ordered set of nodes $(n_1, n_2, \ldots, n_k)$ with each n_{i+1} a successor of n_i. There exists a path from n_i to n_j if and only if n_j is accessible from n_i. Every path has a cost which is obtained by adding the individual costs of each arc, $c_{i,i+1}$, in the path. An *optimal path* from n_i to n_j is a path having the smallest cost over the set of all paths from n_i to n_j. We shall represent this cost by $h(n_i, n_j)$.

This paper will be concerned with the subgraph G_s from some single specified *start node* s. We define a nonempty set T of nodes in G_s as the *goal nodes*.[1] For any node n in G_s, an element $t \in T$ is a *preferred* goal node of n if and only if the cost of an optimal path from n to t does not exceed the cost of any other path from n to any member of T. For simplicity, we shall represent the unique cost of an optimal path from n to a preferred goal node of n by the symbol $h(n)$; i.e., $h(n) = \min_{t \in T} h(n,t)$.

C. Algorithms for Finding Minimum Cost Paths

We are interested in algorithms that search G_s to find an optimal path from s to a preferred goal node of s. What we mean by searching a graph and finding an optimal path is made clear by describing in general how such algorithms proceed. Starting with the node s, they generate some part of the subgraph G_s by repetitive application of the successor operator Γ. During the course of the algorithm, if Γ is applied to a node, we say that the algorithm has *expanded* that node.

We can keep track of the minimum cost path from s to each node encountered as follows. Each time a node is expanded, we store with each successor node n both the cost of getting to n by the lowest cost path found thus far, and a pointer to the predecessor of n along that path. Eventually the algorithm terminates at some goal node t, and no more nodes are expanded. We can then reconstruct a minimum cost path from s to t known at the time of termination simply by chaining back from t to s through the pointers.

We call an algorithm *admissible* if it is guaranteed to find an optimal path from s to a preferred goal node of s for any δ graph. Various admissible algorithms may differ both in the order in which they expand the nodes of G_s and in the number of nodes expanded. In the next section, we shall propose a way of ordering node expansion and show that the resulting algorithm is admissible. Then, in a following section, we shall show, under a mild assumption, that this algorithm uses information from the problem represented by the graph in an optimal way. That is, it expands the smallest number of nodes necessary to guarantee finding an optimal path.

II. An Admissible Searching Algorithm

A. Description of the Algorithm

In order to expand the fewest possible nodes in searching for an optimal path, a search algorithm must constantly make as informed a decision as possible about which node to expand next. If it expands nodes which obviously cannot be on an optimal path, it is wasting effort. On the other hand, if it continues to ignore nodes that might be on an

[1] We exclude the trivial case of $s \in T$.

optimal path, it will sometimes fail to find such a path and thus not be admissible. An efficient algorithm obviously needs some way to evaluate available nodes to determine which one should be expanded next. Suppose some *evaluation function* $\hat{f}(n)$ could be calculated for any node n. We shall suggest a specific function below, but first we shall describe how a search algorithm would use such a function.

Let our evaluation function $\hat{f}(n)$ be defined in such a way that the available node having the smallest value of $\hat{f}$ is the node that should be expanded next. Then we can define a search algorithm as follows.

Search Algorithm A:*

1) Mark s "open" and calculate $\hat{f}(s)$.

2) Select the open node n whose value of $\hat{f}$ is smallest. Resolve ties arbitrarily, but always in favor of any node $n \in T$.

3) If $n \in T$, mark n "closed" and terminate the algorithm.

4) Otherwise, mark n closed and apply the successor operator Γ to n. Calculate $\hat{f}$ for each successor of n and mark as open each successor not already marked closed. Remark as open any closed node n_i which is a successor of n and for which $\hat{f}(n_i)$ is smaller now than it was when n_i was marked closed. Go to Step 2.

We shall next show that for a suitable choice of the evaluation function $\hat{f}$, the algorithm A^* is guaranteed to find an optimal path to a preferred goal node of s and thus is admissible.

B. The Evaluation Function

For any subgraph G_s and any goal set T, let $f(n)$ be the actual cost of an optimal path *constrained to go through* n, from s to a preferred goal node of n.

Note that $f(s) = h(s)$ is the cost of an unconstrained optimal path from s to a preferred goal node of s. In fact, $f(n) = f(s)$ for every node n on an optimal path, and $f(n) > f(s)$ for every node n not on an optimal path. Thus, although $f(n)$ is not known a priori (in fact, determination of the true value of $f(n)$ may be the main problem of interest), it seems reasonable to use an estimate of $f(n)$ as the evaluation function $\hat{f}(n)$. In the remainder of this paper, we shall exhibit some properties of the search algorithm A^* when the cost $f(n)$ of an optimal path through node n is estimated by an appropriate evaluation function $\hat{f}(n)$.

We can write $f(n)$ as the sum of two parts:

$$f(n) = g(n) + h(n) \tag{1}$$

where $g(n)$ is the actual cost of an optimal path from s to n, and $h(n)$ is the actual cost of an optimal path from n to a preferred goal node of n.

Now, if we had estimates of g and h, we could add them to form an estimate of f. Let $\hat{g}(n)$ be an estimate of $g(n)$. An obvious choice for $\hat{g}(n)$ is the cost of the path from s to n having the smallest cost so far found by the algorithm. Notice that this implies $\hat{g}(n) \geq g(n)$.

A simple example will illustrate that this estimate is easy to calculate as the algorithm proceeds. Consider the

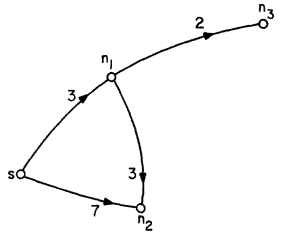

Fig. 1.

subgraph shown in Fig. 1. It consists of a start node s and three other nodes, n_1, n_2, and n_3. The arcs are shown with arrowheads and costs. Let us trace how algorithm A^* proceeded in generating this subgraph. Starting with s, we obtain successors n_1 and n_2. The estimates $\hat{g}(n_1)$ and $\hat{g}(n_2)$ are then 3 and 7, respectively. Suppose A^* expands n_1 next with successors n_2 and n_3. At this stage $\hat{g}(n_3) = 3 + 2 = 5$, and $\hat{g}(n_2)$ is lowered (because a less costly path to it has been found) to $3 + 3 = 6$. The value of $\hat{g}(n_1)$ remains equal to 3.

Next we must have an estimate $\hat{h}(n)$ of $h(n)$. Here we rely on information from the problem domain. Many problems that can be represented as a problem of finding a minimum cost path through a graph contain some "physical" information that can be used to form the estimate $\hat{h}$. In our example of cities connected by roads, $\hat{h}(n)$ might be the airline distance between city n and the goal city. This distance is the shortest possible length of any road connecting city n with the goal city; thus it is a lower bound on $h(n)$. We shall have more to say later about using information from the problem domain to form an estimate $\hat{h}$, but first we can prove that if $\hat{h}$ is any lower bound of h, then the algorithm A^* is admissible.

C. The Admissibility of A*

We shall take as our evaluation function to be used in A^*

$$\hat{f}(n) = \hat{g}(n) + \hat{h}(n) \tag{2}$$

where $\hat{g}(n)$ is the cost of the path from s to n with minimum cost so far found by A^*, and $\hat{h}(n)$ is any estimate of the cost of an optimal path from n to a preferred goal node of n. We first prove a lemma.

Lemma 1

For any nonclosed node n and for any optimal path P from s to n, there exists an open node n' on P with $\hat{g}(n') = g(n')$.

Proof: Let $P = (s = n_0, n_1, n_2, \cdots, n_k = n)$. If s is open (that is, A^* has not completed even one iteration), let $n' = s$, and the lemma is trivially true since $\hat{g}(s) = g(s) = 0$. Suppose s is closed. Let Δ be the set of all closed nodes n_i in P for which $\hat{g}(n_i) = g(n_i)$. Δ is not empty, since by assumption $s \in \Delta$. Let n^* be the element of Δ with highest index. Clearly, $n^* \neq n$, as n is nonclosed. Let n' be the successor of n^* on P. (Possibly $n' = n$.) Now $\hat{g}(n') \leq \hat{g}(n^*) + c_{n^*,n'}$ by definition of $\hat{g}$; $\hat{g}(n^*) = g(n^*)$ because n^* is in Δ, and $g(n') = g(n^*) + c_{n^*,n'}$ because P is an optimal path. Therefore, $g(n') \leq \hat{g}(n')$. But in

general, $\hat{g}(n') \geq g(n')$, since the lowest cost $g(n')$ from s to n' discovered at any time is certainly not lower than the optimal cost $g(n')$. Thus $\hat{g}(n') = g(n')$, and moreover, n' must be open by the definition of Δ.

Corollary

Suppose $\hat{h}(n) \leq h(n)$ for all n, and suppose A^* has not terminated. Then, for any optimal path P from s to any preferred goal node of s, there exists an open node n' on P with $\hat{f}(n') \leq f(s)$.

Proof: By the lemma, there exists an open node n' on P with $\hat{g}(n') = g(n')$, so by definition of $\hat{f}$

$$\begin{aligned}\hat{f}(n') &= \hat{g}(n') + \hat{h}(n') \\ &= g(n') + \hat{h}(n') \\ &\leq g(n') + h(n') = f(n').\end{aligned}$$

But P is an optimal path, so $f(n') = f(s)$ for all $n' \epsilon P$, which completes the proof. We can now prove our first theorem.

Theorem 1

If $\hat{h}(n) \leq h(n)$ for all n, then A^* is admissible.

Proof: We prove this theorem by assuming the contrary, namely that A^* does not terminate by finding an optimal path to a preferred goal node of s. There are three cases to consider: either the algorithm terminates at a nongoal node, fails to terminate at all, or terminates at a goal node without achieving minimum cost.

Case 1

Termination is at a nongoal node. This case contradicts the termination condition (Step 3) of the algorithm, so it may be eliminated immediately.

Case 2

There is no termination. Let t be a preferred goal node of s, accessible from the start in a finite number of steps, with associated minimum cost $f(s)$. Since the cost on any arc is at least δ, then for any node n further than $M = f(s)/\delta$ steps from s, we have $\hat{f}(n) \geq g(n) \geq g(n) > M\delta = f(s)$. Clearly, no node n further than M steps from s is ever expanded, for by the corollary to Lemma 1, there will be some open node n' on an optimal path such that $\hat{f}(n') \leq f(s) < \hat{f}(n)$, so, by Step 2, A^* will select n' instead of n. Failure of A^* to terminate could then only be caused by continued reopening of nodes within M steps of s. Let $\chi(M)$ be the set of nodes accessible within M steps from s, and let $\nu(M)$ be the number of nodes in $\chi(M)$. Now, any node n in $\chi(M)$ can be reopened at most a finite number of times, say $\tilde{\rho}(n, M)$, since there are only a finite number of paths from s to n passing only through nodes within M steps of s. Let

$$\rho(M) = \max_{n \epsilon \chi(M)} \tilde{\rho}(n, M),$$

the maximum number of times any one node can be reopened. Hence, after at most $\nu(M)\rho(M)$ expansions, all nodes in $\chi(M)$ must be forever closed. Since no nodes outside $\chi(M)$ can be expanded, A^* must terminate.

Case 3

Termination is at a goal node without achieving minimum cost. Suppose A^* terminates at some goal node t with $\hat{f}(t) = \hat{g}(t) > f(s)$. But by the corollary to Lemma 1, there existed just before termination an open node n' on an optimal path with $\hat{f}(n') \leq f(s) < \hat{f}(t)$. Thus at this stage, n' would have been selected for expansion rather than t, contradicting the assumption that A^* terminated.

The proof of Theorem 1 is now complete. In the next section, we shall show that for a certain choice of the function $\hat{h}(n)$, A^* is not only admissible but optimal, in the sense that no other admissible algorithm expands fewer nodes.

III. On the Optimality of A^*

A. Limitation of Subgraphs by Information from the Problem

In the preceding section, we proved that if $\hat{h}(n)$ is any lower bound on $\hat{h}(n)$, then A^* is admissible. One such lower bound is $\hat{h}(n) = 0$ for all n. Such an estimate amounts to assuming that any open node n might be arbitrarily close to a preferred goal node of n. Then the set $\{G_n\}$ is unconstrained; anything is possible at node n, and, in particular, if $\hat{g}$ is a minimum at node n, then node n must be expanded by every admissible algorithm.

Often, however, we have information from the problem that constrains the set $\{G_n\}$ of possible subgraphs at each node. In our example with cities connected by roads, no subgraph G_n is possible for which $h(n)$ is less than the airline distance between city n and a preferred goal city of n. In general, if the set of possible subgraphs is constrained, one can find a higher lower bound of $h(n)$ than one can for the unconstrained situation. If this higher lower bound is used for $\hat{h}(n)$, then A^* is still admissible, but, as will become obvious later, A^* will generally expand fewer nodes. We shall assume in this section that at each node n, certain information is available from the physical situation on which we can base a computation to limit the set $\{G_n\}$ of possible subgraphs.

Suppose we denote the set of all subgraphs from node n by the symbol $\{G_{n,\omega}\}$ where ω indexes each subgraph, and ω is in some index set Ω_n. Now, we presume that certain information is available from the problem domain about the state that node n represents; this information limits the set of subgraphs from node n to the set $\{G_{n,\theta}\}$, where θ is in some smaller index set $\Theta_n \subset \Omega_n$.

For each $G_{n,\theta}$ in $\{G_{n,\theta}\}$ there corresponds a cost $h_\theta(n)$ of the optimum path from n to a preferred goal node of n. We shall now take as our estimate $\hat{h}(n)$, the greatest lower bound for $h_\theta(n)$. That is,

$$\hat{h}(n) = \inf_{\theta \epsilon \Theta_n} h_\theta(n). \tag{3}$$

We assume the infimum is achieved for some $\theta \epsilon \Theta_n$.

In actual problems one probably never has an explicit representation for $\{G_{n,\theta}\}$, but instead one selects a pro-

cedure for computing $\hat{h}(n)$, known from information about the problem domain, to be a lower bound on $h(n)$. This selection itself induces the set $\{G_{n,\theta}\}$ by (3). Nevertheless, it is convenient to proceed with our formal discussion as if $\{G_{n,\theta}\}$ were available and as if $\hat{h}(n)$ were explicitly calculated from (3). For the rest of this paper, we assume that the algorithm A^* uses (3) as the definition of $\hat{h}$.

B. A Consistency Assumption

When a real problem is modeled by a graph, each node of the graph corresponds to some state in the problem domain. Our general knowledge about the structure of the problem domain, together with the specific state represented by a node n, determines how the set Ω_n is reduced to the set Θ_n. However, we shall make one assumption about the uniformity of the manner in which knowledge of the problem domain is used to impose this reduction. This assumption may be stated formally as follows. For any nodes m and n,

$$h(m,n) + \inf_{\theta \in \Theta_n} h_\theta(n) \geq \inf_{\theta \in \Theta_m} h_\theta(m). \tag{4}$$

Using the definition of $\hat{h}$ given in (3), we can restate (4) as a kind of triangle inequality:

$$h(m, n) + \hat{h}(n) \geq \hat{h}(m). \tag{5}$$

The assumption expressed by (4) [and therefore (5)] amounts to a type of consistency assumption on the estimate $\hat{h}(n)$ over the nodes. It means that any estimate $\hat{h}(n)$ calculated from data available in the "physical" situation represented by node n alone would not be improved by using corresponding data from the situations represented by the other nodes. Let us see what this assumption means in the case of our example of cities and roads. Suppose we decide, in this example, to use as an estimate $\hat{h}(n)$, the airline distance from city n to its closest goal city. As we have stated previously, such an estimate is certainly a lower bound on $h(n)$. It induces at each node n a set $\{G_{n,\theta}\}$ of possible subgraphs from n by (3). If we let $d(m, n)$ be the airline distance between the two cities corresponding to nodes n and m, we have $h(m, n) \geq d(m, n)$ and, therefore, by the triangle inequality for Euclidean distance

$$h(m, n) + \hat{h}(n) \geq d(m, n) + \hat{h}(n) \geq \hat{h}(m),$$

which shows that this $\hat{h}$ satisfies the assumption of (5).

Now let us consider for a moment the following $\hat{h}$ for the roads-and-cities problem. Suppose the nodes of the graph are numbered sequentially in the order in which they are discovered. Let $\hat{h}$ for cities represented by nodes with odd-numbered indexes be the airline distance to a preferred goal city of these nodes, and let $\hat{h} = 1$ for nodes with even-numbered indexes. For the graph of Fig. 2, $\hat{f}(s) = \hat{h}(n_1) = 8$. Nodes n_2 and n_3 are the successors of n_1 along arcs with costs as indicated. By the above rule for computing $\hat{h}$, $\hat{h}(n_2) = 1$ while $\hat{h}(n_3) = 5$. Then $\hat{f}(n_2) = 6 + 1 = 7$, while $\hat{f}(n_3) = 3 + 5 = 8$, and algorithm A^* would erroneously

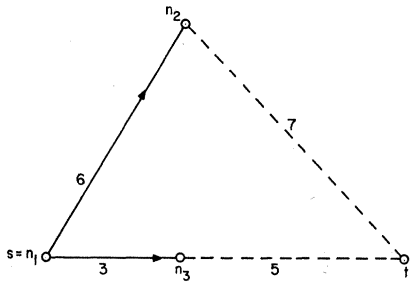

Fig. 2.

choose to expand node n_2 next. This error occurs because the estimates $\hat{h}(s) = 8$ and $\hat{h}(n_2) = 1$ are inconsistent in view of the fact that n_2 is only six units away from s. The information that there cannot exist a path from s to a goal with total cost less than eight was somehow available for the computation of $\hat{h}(s)$, and then ignored during the computation of $\hat{h}(n_2)$. The result is that (5) was violated, i.e.,

$$h(s, n_2) + \hat{h}(n_2) = 6 + 1 < 8 = \hat{h}(s).$$

For the rest of this paper, we shall assume that the family $\{\Theta_n\}$ of index sets satisfies (4) or, equivalently, the procedures for computing the estimates $\hat{h}$ always lead to values that satisfy (5). We shall call this assumption the *consistency assumption*. Note that the estimate $\hat{h}(n) = 0$ for all n trivially satisfies the consistency assumption. Intuitively, the consistency assumption will generally be satisfied by a computation rule for $\hat{h}$ that uniformly uses measurable parameters of the problem state at all nodes; it will generally be violated if the computation rule depends upon any parameter that varies between nodes independently of the problem state (such as a parity count or a random variable), or if the computations at some nodes are more elaborate than at others.

C. Proof of the Optimality of A*

The next lemma makes the important observation about the operation of A^* that, under the consistency assumption, if node n is closed, then $\hat{g}(n) = g(n)$. This fact is important for two reasons. First, it is used in the proof of the theorem about the optimality of A^* to follow, and second, it states that A^* need never reopen a closed node. That is, if A^* expands a node, then the optimal path to that node has already been found. Thus, in Step 4 of the algorithm A^*, the provision for reopening a closed node is vacuous and may be eliminated.

Lemma 2

Suppose the consistency assumption is satisfied, and suppose that node n is closed by A^*. Then $\hat{g}(n) = g(n)$.

Proof: Consider the subgraph G_s just before closing n, and suppose the contrary, i.e., suppose $\hat{g}(n) > g(n)$. Now there exists some optimal path P from s to n. Since $\hat{g}(n) > g(n)$, A^* did not find P. By Lemma 1, there exists an open node n' on P with $\hat{g}(n') = g(n')$. If $n' = n$, we have proved the lemma. Otherwise,

$$g(n) = g(n') + h(n',n)$$
$$= \hat{g}(n') + h(n',n).$$

Thus,

$$\hat{g}(n) > \hat{g}(n') + h(n',n).$$

Adding $\hat{h}(n)$ to both sides yields

$$\hat{g}(n) + \hat{h}(n) > \hat{g}(n') + h(n',n) + \hat{h}(n).$$

We can apply (5) to the right-hand side of the above inequality to yield

$$\hat{g}(n) + \hat{h}(n) > \hat{g}(n') + \hat{h}(n')$$

or

$$\hat{f}(n) > \hat{f}(n'),$$

contradicting the fact that A^* selected n for expansion when n' was available and thus proving the lemma.

The next lemma states that $\hat{f}$ is monotonically nondecreasing on the sequence of nodes closed by A^*.

Lemma 3

Let $(n_1, n_2, \cdots, n_t)$ be the sequence of nodes closed by A^*. Then, if the consistency assumption is satisfied, $p \leq q$ implies $\hat{f}(n_p) \leq \hat{f}(n_q)$.

Proof: Let n be the next node closed by A^* after closing m. Suppose first that the optimum path to n does not go through m. Then n was available at the time m was selected, and the lemma is trivially true. Then suppose that the optimum path to n does, in fact, go through m. Then $g(n) = g(m) + h(m, n)$. Since, by Lemma 2, we have $\hat{g}(n) = g(n)$ and $\hat{g}(m) = g(m)$,

$$\hat{f}(n) = \hat{g}(n) + \hat{h}(n)$$
$$= g(n) + \hat{h}(n)$$
$$= g(m) + h(m, n) + \hat{h}(n)$$
$$\geq g(m) + \hat{h}(m)$$
$$= \hat{g}(m) + \hat{h}(m)$$

where the inequality follows by application of (5). Thus we have

$$\hat{f}(n) \geq \hat{f}(m).$$

Since this fact is true for any pair of nodes n_k and n_{k+1} in the sequence, the proof is complete.

Corollary

Under the premises of the lemma, if n is closed then $\hat{f}(n) \leq f(s)$.

Proof: Let t be the goal node found by A^*. Then $\hat{f}(n) \leq \hat{f}(t) = f(t) = f(s)$.

We can now prove a theorem about the optimality of A^* as compared with any other admissible algorithm A that uses no more information about the problem than does A^*. Let ${\Theta_n}^A$ be the index set used by algorithm A at node n. Then, if ${\Theta_n}^{A^*} \subset {\Theta_n}^A$ for all nodes n in G_s, we shall say that algorithm A is *no more informed* than algorithm A^*.

The next theorem states that if an admissible algorithm A is no more informed than A^*, then any node expanded by A^* must also be expanded by A. We prove this theorem for the special case for which ties never occur in the value of $\hat{f}$ used by A^*. Later we shall generalize the theorem to cover the case where ties can occur, but the proof of the no-ties theorem is so transparent that we include it for clarity.

Theorem 2

Let A be any admissible algorithm no more informed than A^*. Let G_s be any δ graph such that $n \neq m$ implies $\hat{f}(n) \neq \hat{f}(m)$, and let the consistency assumption be satisfied by the $\hat{h}$ used in A^*. Then if node n was expanded by A^*, it was also expanded by A.

Proof: Suppose the contrary. Then there exists some node n expanded by A^* but not by A. Let t^* and t be the preferred goal nodes of s found by A^* and A, respectively. Since A^* and A are both admissible,

$$\hat{f}(t^*) = \hat{g}(t^*) + \hat{h}(t^*) = g(t^*) + 0 = f(t^*) = f(t) = f(s).$$

Since A^* must have expanded n before closing t^*, by Lemma 3 we have

$$\hat{f}(n) < \hat{f}(t^*) = f(t).$$

(Strict inequality occurs because no ties are allowed.)

There exists some graph $G_{n,\theta}$, $\theta \in \Theta_n$, for which $\hat{h}(n) = h(n)$ by the definition of $\hat{h}$. Now by Lemma 2, $\hat{g}(n) = g(n)$. Then on the graph $G_{n,\theta}$, $\hat{f}(n) = f(n)$. Since A is no more informed than A^*, A could not rule out the existence of $G_{n,\theta}$; but A did not expand n before termination and is, therefore, not admissible, contrary to our assumption and completing the proof.

Upon defining $N(A,G_s)$ to be the total number of nodes in G_s expanded by the algorithm A, the following simple corollary is immediate.

Corollary

Under the premises of Theorem 2,

$$N(A^*, G_s) \leq N(A, G_s)$$

with equality if and only if A expands the identical set of nodes as A^*.

In this sense, we claim that A^* is an optimal algorithm. Compared with other no more informed admissible algorithms, it expands the fewest possible nodes necessary to guarantee finding an optimal path.

In case of ties, that is if there exist two or more open nodes $n_1, \cdots, n_k$ with $\hat{f}(n_1) = \cdots = \hat{f}(n_k) < \hat{f}(n)$ for every other open node n, A^* arbitrarily chooses one of the n_i. Consider the set $\mathfrak{A}^*$ of all algorithms that act identically to A^* if there are no ties, but whose members resolve ties differently. An algorithm is a member of $\mathfrak{A}^*$ if it is simply the original A^* with any arbitrary tie-breaking rule.

The next theorem extends Theorem 2 to situations where ties may occur. It states that for any admissible algorithm A, one can always find a member A^* of $\mathfrak{A}^*$ such that each node expanded by A^* is also expanded by A.

Theorem 3

Let A be any admissible algorithm no more informed than the algorithms in $\mathfrak{A}^*$, and suppose the consistency assumption is satisfied by the $\hat{h}$ used in the algorithms in $\mathfrak{A}^*$. Then for any δ graph G_s there exists an $A^* \epsilon \mathfrak{A}^*$ such that every node expanded by A^* is also expanded by A.

Proof: Let G_s be any δ graph and A_1^* be any algorithm in $\mathfrak{A}^*$. If every node of G_s that A_1^* expands is also expanded by A, let A_1^* be the A^* of the theorem. Otherwise, we will show how to construct the A^* of the theorem by changing the tie-breaking rule of A_1^*. Let L be the set of nodes expanded by A, and let $P = (s, n_1, n_2, \cdots, n_k, t)$ be the optimal path found by A.

Expand nodes as prescribed by A_1^* as long as all nodes selected for expansion are elements of L. Let n be the first node selected for expansion by A_1^* which is not in L. Now $\hat{f}(n) \leq f(s)$ by the corollary to Lemma 3. Since $\hat{f}(n) < f(s) = f(t)$ would imply that A is inadmissible (by the argument of Theorem 2), we may conclude that $\hat{f}(n) = f(s)$. At the time A_1^* selected n, goal node t was not closed (or A_1^* would have been terminated). Then by the corollary to Lemma 1, there is an open node n' on P such that $\hat{f}(n') \leq f(s) = \hat{f}(n)$. But since n was selected for expansion by A_1^* instead of n', $\hat{f}(n) \leq \hat{f}(n')$. Hence $\hat{f}(n) \leq \hat{f}(n') \leq \hat{f}(n)$, so $\hat{f}(n) = \hat{f}(n')$. Let A_2^* be identical to A_1^* except that the tie-breaking rule is modified just enough to choose n' instead of n. By repeating the above argument, we obtain for some i an $A_i^* \epsilon \mathfrak{A}^*$ that expands only nodes that are also expanded by A, completing the proof of the theorem.

Corollary 1

Suppose the premises of the theorem are satisfied. Then for any δ graph G_s there exists an $A^* \epsilon \mathfrak{A}^*$ such that $N(A^*, G_s) \leq N(A, G_s)$, with equality if and only if A expands the identical set of nodes as A^*.

Since we cannot select the most fortuitous tie-breaking rule ahead of time for each graph, it is of interest to ask how all members of $\mathfrak{A}^*$ compare against any admissible algorithm A in the number of nodes expanded. Let us define a *critical* tie between n and n' as one for which $\hat{f}(n) = \hat{f}(n') = f(s)$. Then we have the following as a second corollary to Theorem 3.

Corollary 2

Suppose the premises of the theorem are satisfied. Let $R(A^*, G_s)$ be the number of critical ties which occurred in the course of applying A^* to G_s. Then for any δ graph G_s and *any* $A^* \epsilon \mathfrak{A}^*$,

$$N(A^*, G_s) \leq N(A, G_s) + R(A^*, G_s).$$

Proof: For any noncritical tie, all alternative nodes must be expanded by A as well as by A^* or A would not be admissible. Therefore, we need merely observe that each node expanded by A^* but not by A must correspond to a different critical tie in which A^*'s tie-breaking rule made the inappropriate choice.

Of course, one must remember that when A does expand fewer nodes than some particular A^* in $\mathfrak{A}^*$, it is only because A was in some sense "lucky" for the graph being searched, and that there exists a graph consistent with the information available to A and A^* for which A^* would not search more nodes than A.

Note that, although one cannot keep a running estimate of R while the algorithm proceeds because one does not know the value of $f(s)$, this value is established as soon as the algorithm terminates, and R can then be easily computed. In most practical situations, R is not likely to be large because critical ties are likely to occur only very close to termination of the algorithm, when $\hat{h}$ can become a perfect estimator of h.

IV. Discussion and Conclusions

A. Comparisons Between A* and Other Search Techniques

Earlier we mentioned that the estimate $\hat{h}(n) \equiv 0$ for all n trivially satisfies the consistency assumption. In this case, $\hat{f}(n) = \hat{g}(n)$, the lowest cost so far discovered to node n. Such an estimate is appropriate when no information at all is available from the problem domain. In this case, an admissible algorithm cannot rule out the possibility that the goal might be as close as δ to that node with minimum $g(n)$. Pollack and Wiebenson[1] discuss an algorithm, proposed to them by Minty in a private communication, that is essentially identical to our A^* using $\hat{f}(n) = \hat{g}(n)$.

Many algorithms, such as Moore's "Algorithm D"[6] and Busacker and Saaty's implementation of dynamic programming, keep track of $\hat{g}(n)$ but do not use it to order the expansion of nodes. The nodes are expanded in a "breadth-first" order, meaning that all nodes one step away from the start are expanded first, then all nodes two steps away, etc. Such methods must allow for changes in the value of $\hat{g}(n)$ as a node previously expanded is later reached again by a less costly route.

It might be argued that the algorithms of Moore, Busacker and Saaty, and other equivalent algorithms (sometimes known as "water flow" or "amoeba" algorithms) are advantageous because they first encounter the goal by a path with a minimum number of steps. This argument merely reflects an imprecise formulation of the problem, since it implies that the number of steps, and not the cost of each step, is the quantity to be minimized. Indeed, if we set $c_{ij} = 1$ for all arcs, this class of algorithms is identical to A^* with $\hat{h} = 0$. We emphasize that, as is always the case when a mathematical model is used to represent a real problem, the first responsibility of the investigator is to ensure that the model is an adequate representation of the problem for his purposes.

It is beyond the scope of the discussion to consider how to define a successor operator Γ or assign costs c_{ij} so that the resulting graph realistically reflects the nature of a specific problem domain.[2]

B. The Heuristic Power of the Estimate $\hat{h}$

The algorithm A^* is actually a family of algorithms; the choice of a particular function $\hat{h}$ selects a particular algorithm from the family. The function $\hat{h}$ can be used to tailor A^* for particular applications.

As was discussed above, the choice $\hat{h} = 0$ corresponds to the case of knowing, or at least of using, absolutely no information from the problem domain. In our example of cities connected by roads, this would correspond to assuming a priori that roads could travel through "hyperspace," i.e., that any city may be an arbitrarily small road distance from any other city regardless of their geographic coordinates.

Since we are, in fact, "more informed" about the nature of Euclidean space, we might increase $\hat{h}(n)$ from 0 to $\sqrt{x^2 + y^2}$ (where x and y are the magnitudes of the differences in the x, y coordinates of the city represented by node n and its closest goal city). The algorithm would then still find the shortest path, but would do so by expanding, typically, considerably fewer nodes. In fact, A^* expands no more nodes[3] than any admissible algorithm that uses no more information from the problem domain; viz., the information that a road between two cities might be as short as the airline distance between them.

Of course, the discussion thus far has not considered the cost of computing $\hat{h}$ each time a node is generated on the graph. It could be that the computational effort required to compute $\sqrt{x^2 + y^2}$ is significant when compared to the effort involved in expanding a few extra nodes; the optimal procedure in the sense of minimum number of nodes expanded might not be optimal in the sense of minimum total resources expended. In this case one might, for example, choose $\hat{h}(n) = (x + y)/2$. Since $(x + y)/2 < \sqrt{x^2 + y^2}$, the algorithm is still admissible. Since we are not using "all" our knowledge of the problem domain, a few extra nodes may be expanded, but total computational effort may be reduced; again, each "extra" node must also be expanded by other admissible algorithms that limit themselves to the "knowledge" that the distance between two cities may be as small as $(x + y)/2$.

Now suppose we would like to reduce our computational effort still further, and would be satisfied with a solution path whose cost is not necessarily minimal. Then we could choose an $\hat{h}$ somewhat larger than the one defined by (3). The algorithm would no longer be admissible, but it might be more desirable, from a heuristic point of view, than any admissible algorithm. In our roads-and-cities example, we might let $\hat{h} = x + y$. Since road distance is usually substantially greater than airline distance, this $\hat{h}$ will usually, but not always, result in an optimal solution path. Often, but not always, fewer nodes will be expanded and less arithmetic effort required than if we used $\hat{h}(n) = \sqrt{x^2 + y^2}$.

Thus we see that the formulation presented uses one function, $\hat{h}$, to embody in a formal theory all knowledge available from the problem domain. The selection of $\hat{h}$, therefore, permits one to choose a desirable compromise between admissibility, heuristic effectiveness, and computational efficiency.

[2] We believe that appropriate choices for Γ and c_{ij} will permit many of the problem domains in the heuristic programming literature[7] to be mapped into graphs of the type treated in this paper. This could lead to a clearer understanding of the effects of "heuristics" that use information from the problem domain.

[3] Except for possible critical ties, as discussed in Corollary 2 of Theorem 3.

References

[1] M. Pollack and W. Wiebenson, "Solutions of the shortest-route problem—a review," *Operations Res.*, vol. 8, March–April 1960.

[2] R. Busacker and T. Saaty, *Finite Graphs and Networks: An Introduction with Applications.* New York: McGraw–Hill, 1965, ch. 3.

[3] R. Bellman and S. Dreyfus, *Applied Dynamic Programming.* Princeton, N. J.: Princeton University Press, 1962.

[4] H. Gelernter, "Realization of a geometry-theorem proving machine," in *Computers and Thought.* New York: McGraw–Hill, 1963.

[5] A. Samuel, "Some studies in machine learning using the game of checkers," in *Computers and Thought.* New York: McGraw–Hill, 1963.

[6] E. Moore, "The shortest path through a maze," *Proc. Internat'l Symp. on Theory of Switching* (April 2–5, 1957), pt. 2. Also, *The Annals of the Computation Laboratory of Harvard University*, vol. 30. Cambridge, Mass.: Harvard University Press, 1959.

[7] E. Feigenbaum and J. Feldman, *Computers and Thought.* New York: McGraw–Hill, 1963.