Heuristic Search Viewed as Path Finding in a Graph

Ira Pohl

IBM Thomas J. Watson Research Center, Yorktown Heights, New York

Recommended by E. J. Sandewall

ABSTRACT

This paper presents a particular model of heuristic search as a path-finding problem in a directed graph. A class of graph-searching procedures is described which uses a heuristic function to guide search. Heuristic functions are estimates of the number of edges that remain to be traversed in reaching a goal node. A number of theoretical results for this model, and the intuition for these results, are presented. They relate the efficiency of search to the accuracy of the heuristic function. The results also explore efficiency as a consequence of the reliance or weight placed on the heuristics used.

1. Introduction

Heuristic search has been one of the important ideas to grow out of artificial intelligence research. It is an ill-defined concept, and has been used as an umbrella for many computational techniques which are hard to classify or analyze. This is beneficial in that it leaves the imagination unfettered to try any technique that works on a complex problem. However, leaving the concept vague has meant that the same ideas are rediscovered, often cloaked in other terminology rather than abstracting their essence and understanding the procedure more deeply. Often, analytical results lead to more efficient procedures. Such has been the case in sorting [1] and matrix multiplication [2], and the same is hoped for this development of heuristic search.

This paper attempts to present an overview of recent developments in formally characterizing heuristic search. The model allows us to define precisely measures of search efficiency in machine-independent terms. The problems of when these techniques will work and how well they will work are interpreted as questions of constructing accurate computable heuristic functions. This formulation allows analytical results on how to construct efficient heuristic search procedures.

Artificial Intelligence 1 (1970), 193–204

This paper does not attempt a history of developments in this area [3–6]; nevertheless, a few of the main accomplishments that shaped this viewpoint are necessary for context. The heuristic search programs of Newell, Simon, and Shaw starting with the logic theorist [7, 8] generated a large amount of enthusiasm. They attempted to apply these techniques to a wide variety of problem domains, empirically demonstrating the generality of heuristic search. A more recent empirical attempt at generality coupled with an active concern for efficiency has been the Graph Traverser programs [9, 10] developed principally by Doran and Michie. Here the search mechanism becomes specifically a graph-searching program guided by a heuristic function which is an estimator of distances in the graph. The Graph Traverser program was used to compare the efficiency of different heuristic functions for solving a class of puzzles.

Another line of development was motivated by the need for a procedure for guiding the SRI robot [11] around its world. In choosing to represent the world as a directed graph, the problem of directing the robot was transformed into a shortest-path problem. However, the robot had a means for estimating Euclidean distances in his world. Hart, Nilsson, and Raphael [12] discovered how to use this information to improve the efficiency of computing the shortest path. They generalized this to the concept of heuristic estimation and provided an algorithm which would use this information and still compute shortest paths. They produced the first theorems on efficiency as a function of accuracy of the heuristic function. The results above and related techniques [13–16] in enumerative combinatorial computations have been responsible for the development presented here.

An informal definition of a heuristic is any device that aids search efficiency by either restricting the region searched or appropriately ordering the search. Closely allied techniques of branch and bound, dynamic programming and alpha-beta can be lumped together with heuristic search and called intelligent computational enumeration. A sample of the problems tackled with these methods includes puzzles [7, 9], theorem proving [8, 17], combinatorial problems [13], and games [18]. The use of evaluation functions in guiding search in a discrete problem space occurs in each procedure. The most efficient evaluation functions rely on the semantics of a particular domain. To develop a problem-solving program for performance in a particular complex area, such as chess, requires a large investment in deriving and mechanizing domain specific information. For example, recognizing when it is appropriate to launch a king side attack by pawnstorm is not useful in any obvious way for solving the traveling salesman problem. Regardless of the proportion of work that must go into devising these domain-dependent heuristics versus improving the efficiency of the general search procedures, the pervasiveness of these search procedures argue for the desirability of maximizing their efficiency.

2. The M[illegible]

To achieve an analysis, a narrow, well-defined point of view of heuristic search as a problem of finding a path in a locally finite directed graph is taken here. The reader interested in the development of these techniques should consult references [3–6, 9, 13–15, 19]. Also alternative models exist which are promising [20–23].

A problem space is a locally finite directed graph G.

G: $X = \{x_1, x_2, \ldots\}$, X is the set of *nodes* and can be infinite

$E = \{(x_i, x_j) | x_i, x_j \in X, x_j \in \Gamma(x_i)\}$, E is the set of *edges* or *arcs* and can be infinite if $|X|$ is infinite (the cardinality or number of objects in a set is denoted |set name|)

Γ is the successor mapping.

Γ: $X \to 2^X$ the mapping of X into its power set, and Γ is finite, i.e., for all x, $|\Gamma(x)| \in N$, the integers

In using directed graphs to specify domains, a data structure is stored with each node which contains a problem description. The successor mapping defines the structural character of the problem space, identifying for each particular node its immediate neighbors. A problem consists of some node (or set of nodes) which is to be the initial node, and another node (or set of nodes) which is to be the terminal node. A solution of the problem is a path from the (an) initial node to the (a) terminal node. There are many variants [15] of this basic problem involving constraints on the paths acceptable as solutions.

A path in a graph is written as $\mu = (x_1, x_2, \ldots, x_k)$, where $(x_i, x_{i+1}) \in E$. The cardinality length of this path is $l(\mu) = k-1$, the number of edges traversed in going from x_1 to x_k. Length or distance will be the cardinality length, unless explicitly stated otherwise. The initial node of a problem will be denoted s, and the terminal node will be denoted t. It is easy for an algorithm to keep track of the length of a path from s to some node x which it has reached, and this length, $l(\mu_{sx})$, will be denoted $g(x)$. Heuristic search algorithms will attempt to estimate $l(\mu_{xt})$ by a function $h(x)$ and use this information to improve search efficiency. A search algorithm would begin with node s and, applying the successor mapping, would produce an enumeration of nodes in the graph until it encountered t or failed. Failure could come from exhausting the computational resources available or from exhausting all nodes reachable from s without finding t.

Exhaustive methods are impractical in large spaces. In these spaces the number of nodes grows exponentially with distance from the initial node. An algorithm's work is directly proportional to the number of nodes it must enumerate, and for such an algorithm to find solution paths of great length it must try nonexhaustive methods. In our model, the heuristics attempt to

get an efficient directed search by hopefully following some geodesic in our discrete space. These heuristic functions use features of the problem description stored at a node to estimate the distance remaining to the terminal node. Though an actual problem solver would compute $h(x)$ by looking at the features of the problem state stored there and comparing them to what is desired, it is a convenient mathematical fiction to just think of h as a function on the nodes.

Along with a graph model of problem solving, the class of algorithms which will be used for producing solutions must be described. This class of heuristic path algorithms (HPA) will conduct enumerative searches of graphs, using the h and g functions to guide the search.

Heuristic Path Algorithm (HPA)

s = start node, t = terminal node, x = any node

g: $X \to N$, the number of edges from s to x enumerated by HPA—distance-to-date term

h: $X \to R^+$ (the nonnegative reals), an estimate of the number of edges on a shortest path from x to t—heuristic function

$f(x) = (1 - \omega)g(x) + \omega h(x)$, $0 \leqslant \omega \leqslant 1$—evaluation function

S = set of nodes already visited and expanded

$\tilde{S}$ = set of nodes one edge removed from those in S, but not in S—candidate set

1. Place s in S and calculate $\Gamma(s)$, placing them in $\tilde{S}$. If $x \in \Gamma(s)$, then $g(x) = 1$ and $f(x) = (1 - \omega) + \omega h(x)$.

2. Select $n \in \tilde{S}$ such that $f(n)$ is a minimum.

3. Place n in S and $\Gamma(n)$ in $\tilde{S}$, discarding any nodes already in $S \cup \tilde{S}$. Calculate f for these new successors of n. If $x \in \Gamma(n)$, then $g(x) = 1 + g(n)$ and $f(x) = (1 - \omega)g(x) + \omega h(x)$.

4. If n is the goal state, then halt, otherwise go to step 2.

HPA is a typical path-finding algorithm of the Moore maze finding variety [16]. It would be the Moore algorithm for the cardinality metric if $\omega = 0$. If $\omega = 1$, it would be the Graph Traverser algorithm [9, 10] or, if $\omega = 0{\cdot}5$, it would be similar to the Hart, Nilsson, and Raphael algorithm [12]. The distance-to-date term, which is ordinarily used in combinatorial problems, and the heuristic estimator, which is ordinarily used in artificial intelligence problems, are naturally combined into an evaluation function which is their weighted sum. The combinatorial algorithms hesitate to use the heuristic term because they may throw away shortest (optimal) solutions, and the artificial intelligence algorithms hesitate to use the distance-to-date term because it may exponentially broaden the search. However, an appropriate

analysis in both instances leads to desirable gains in solving both styles of problem.

3. Theoretical Results

The path problem in a directed locally finite graph being solved by HPA is an explicit, well-defined model of heuristic search. A particular domain will have a use for specific heuristic estimators, but some general results relating the accuracy of the estimator to efficiency will be derived. The proofs are only sketched, as the details are either available elsewhere [3, 12, 19] or are clear from the sketch. The reader is advised to look at some of the other descriptions of these algorithms and related results, especially the Hart, Nilsson and Raphael paper [12] whose more general results are reinterpreted for our model.

First, we make some preliminary remarks and more definitions:

$h_p(x) = \min_{\mu_{xt}} (l(\mu_{xt}))$, over all paths from x to t, h_p is the minimum length, called the *perfect* estimator

h is in error ε at node x, if $|h_p(x) - h(x)| = \varepsilon$

h is of *bounded error* ε when

$h_p - \varepsilon \leqslant h \leqslant h_p + \varepsilon$ is satisfied throughout the graph

By "h is a computable total function from X to R^+" is meant that, for all $x \in X$, $h(x)$ is the result of a program that always terminates with a value in R^+. Ordinarily, references to h and h_p are to the class of computable total functions.

The first concern in examining this algorithm is to describe when it will find a solution path. As is the case with any sufficiently general proof procedure, HPA cannot always provide a decision procedure.

THEOREM 1. (*Undecidability.*) *For some problem domains there does not exist any total computable* h_p.

Proof. The Herbrand search problem for the first-order predicate calculus is undecidable. While there is some metric on this space, it could not be total computable or else a decision procedure for the first-order predicate calculus would exist.

Remark. Any finite space has a total computable h_p, namely, enumerate the space and make a table of the values found. This is impractical for large problems without nice mathematical properties. The graph for the traveling salesman problem for n cities has $(n-1)!$ nodes.

HPA, however, does provide a recursive enumeration of the nodes in its domain. The only condition that must be insisted on is that the evaluation function be *not* completely reliant on h, i.e., $\omega < 1$.

THEOREM 2. (*Recursively Enumerable.*) *If some solution path exists,* HPA *with* $\omega < 1$ *will find one.*

Proof. Let n = the length of the shortest path from s to t. G is locally finite, so that within n edges of s there are only a finite number of nodes. It is necessary to show that these nodes will be reached in some finite amount of computation.

h: $X \to R^+$, so that nodes in $\tilde{S}$ with $g(x) = 1$ will have some maximum value of $f(x)$ bounded by M, an integer. Therefore HPA will either find a solution path or will expand all nodes with $g(x) = 1$ before it grows any path with

$$g(x) > \left(\frac{1}{1-\omega}\right) \cdot M$$

This argument can be applied inductively to show that an exhaustive search will be carried out for any n. Q.E.D.

Remark. For $\omega = 1$, $h = 1/(1 + h_p)$, HPA will not find solution paths in an infinite graph.

In order that HPA will be a decision procedure for a given domain, criteria on the accuracy of h as a distance estimator give the following necessary and sufficient conditions.

THEOREM 3. (*Decidability.*) *If there exists a total computable h of bounded error for some domain, then* HPA *provides a decision procedure for this domain. Conversely, any problem domain for which there exists a decision procedure has a computable h of bounded error.*

Proof. If the error is bounded, then $h_p(s) \leqslant h(s) + \varepsilon$. So HPA enumerating the nodes within cardinality distance $h(s) + \varepsilon$ of s will have searched the finite subgraph which contains any solution path. If t is not found within this subgraph, it must not be connected to s and the search can terminate with the answer that no path exists.

If a decision procedure D exists for the domain, then it can be turned into an algorithm for a total computable function of bounded error.

$$D(t) = \begin{cases} \text{false—halt with } \infty \\ \text{true—execute HPA with } \omega = 0 \text{ and use the} \\ \qquad \text{length of the path found} \end{cases}$$

By Theorem 2, the true alternative provides a finite algorithm for finding h_p, certainly bounded in error. Q.E.D.

Corollary 1. *Some infinite domains exist for which the error on any computable heuristic term is unbounded for an infinite number of nodes.*

Proof. Use Theorem 3 and the fact that domains exist which have no decision procedure. Q.E.D.

Quite often the solution path is used as a schedule or plan that will be repeatedly executed at a cost proportional to its length. At these times it is desirable or mandatory that the shortest solution path be found. It was for this case that Hart, Nilsson, and Raphael worked out their results.

The Hart, Nilsson, and Raphael algorithm handled the more general metric where an edge length could be any positive real. It also required a different handling of step 3 of HPA to guarantee minimal path length. The algorithm with this change will be called HPA+.

HPA+ is HPA modified in step 3 to read:

Place n in S and check all $x \in \Gamma(n)$ for the following possibilities. If $x \in \Gamma(n) \cap \bar{S}$ and the new value of $f(x)$ is smaller than its old value, replace the old value by the new value. If $x \in \Gamma(n) \cap S$ and the new value of $f(x)$ is smaller than its old value, remove x from S and place it in $\bar{S}$. Otherwise place x in $\bar{S}$.

THEOREM 4. (*Hart, Nilsson, and Raphael.*) HPA+ *using as its evaluation function* $f(x) = g(x) + h(x)$ (*or equivalently* $\omega = 0{\cdot}5$), *where* $h \leqslant h_p$, *finds a shortest solution path if one exists* [12].

Proof. When the goal node is incorporated in S, it must have the current minimum value of $f(x)$ over all $x \in \bar{S}$. Call this value l_1 and assume that it is not the real minimum. Let the minimum path be μ so that $l_1 > l(\mu)$. Some node y along μ must be in $\bar{S}$ at some time in the computation (by induction).

$$f(y) = g(y) + h(y) \leqslant g(y) + h_p(y) \leqslant l(\mu) < l_1.$$

Contradiction: y was in $\bar{S}$ when t was chosen by HPA+ with value l_1.

Heuristic functions satisfying boundedness were already known in branch and bound applications [13]. They allow the branch and bound enumeration to halt before enumerating the whole search space.

Since the evaluation function is a distance estimator, it is natural to think of heuristic estimators satisfying a form of the triangle inequality. This is the idea of the "consistency assumption" of Hart, Nilsson, and Raphael.

$h_p(x, y)$ = the minimum cardinality distance over the paths from x to y

Then the consistency assumption is

$$h_p(x, y) + h(y) \geqslant h(x)$$

A heuristic function satisfying this condition never returns nodes to set $\bar{S}$. This condition certifies that HPA will behave as HPA+.

THEOREM 5. (*Hart, Nilsson, and Raphael.*) *Given* h_1 *and* h_2 *as heuristic functions satisfying consistency, and* $h_2 < h_1 < h_p$ *throughout the domain, then* HPA *using* $f_1 = g + h_1$ *will only expand some subset* (*possibly proper*) *of nodes that are expanded using* $f_2 = g + h_2$ [12].

Proof. It is sufficient to show that, if HPA with h_1 visits a node, then HPA with h_2 visits the same node. A node will not be expanded if its value exceeds $l(\mu)$, where μ is the shortest path. If $l(\mu) < g + h_2$, then $l(\mu) < g + h_1$. By the inductive use of the consistency assumption the g values of the same node are the same for h_1 and h_2. Any node with

$$l(\mu) > (g + h_1)(x) > (g + h_2)(x)$$

which is connected to s will be reached and expanded. Q.E.D.

Remarks. Some nodes may have $(g + h_1)(x) > l(\mu) > (g + h_2)(x)$; HPA with h_1 will then only expand a proper subset of the nodes that HPA with h_2 expands.

The inner loop of HPA does the basic work of the algorithm. It is executed once for each node expanded, and thus the work of solving a problem is proportional to the number of nodes expanded. This provides a convenient machine-independent measure of work. In Theorem 5, the efficiency of the use of one function which is a uniformly better estimator throughout the domain dominates the efficiency of the poorer estimator. This result applies to functions guaranteed to find a shortest path. In artificial intelligence, most often one is only too glad to get any solution path. This is the difference between an optimal solution and a feasible solution. In looking for feasible solutions cheaply, the constraints on the heuristic functions of Theorem 5 may be abandoned. Many heuristics do not satisfy these constraints, and $\omega \neq 0{\cdot}5$ may also be of benefit. Remember that the Graph Traverser [9] has done well with $\omega = 1$.

In relating the efficiency of search to the accuracy of the heuristic estimator the first question is: "What if h_p, the perfect estimator, is used?"

THEOREM 6. HPA *search with* h_p *is optimal*, i.e., *expands only the nodes along the shortest path for* $1 \geqslant \omega \geqslant 0{\cdot}5$ [19].

Proof.

$$f = (1 - \omega)(g + h) + (2\omega - 1)h$$

Along the shortest path μ, $g + h_p = l(\mu)$ and $(2\omega - 1)h_p$ is monotonic decreasing. At any time the candidate along the shortest path can be seen to have the smallest f. Q.E.D.

Remark. For $\omega < 0{\cdot}5$ the search tends toward a *breadth first* search. A pure breadth first search is HPA with $\omega = 0$. Then each node is expanded in order of its distance from s. With $\omega < 0{\cdot}5$ nodes off, the shortest path may be expanded since $(2\omega - 1)h_p \leqslant 0$ is now monotonic, increasing along the shortest path.

As the accuracy of the heuristic function declines, the number of nodes expanded by HPA can be expected to increase. The search can be inefficient because it can expand nodes very distant from the terminal node or because it is exhaustive within the search radius. The search radius is defined as the $\max_{x \in s}(g(x))$ and must be at least $h_p(t)$ when t is expanded. The search radius in finding a node at distance k from s with HPA, using $\omega = 0$, is just k. This search is exhaustive within this radius because all nodes x with $g(x) < k$ will have been expanded. When using h and nonzero values of ω, the algorithm possibly increases the search radius but hopefully gains by being non-exhaustive within that larger radius.

The relation between the weighting on the heuristic term and search

efficiency was examined in Theorem 6 for the perfect estimator. This is the specific case of a heuristic function of bounded error ($\varepsilon = 0$). More generally, the heuristic functions of nonzero bounded error are of interest. The characterization of efficiency will be in terms of the worst possible behavior allowable for HPA, using an h of bounded error. This idea has its parallel in the worst case error [24] for numerical computations. There a particular single step is executed with a limited amount of precision. Each step of the computation may introduce a certain amount of error. The maximum conceivable error in performing some computation is the worst case error. Though achievable, it is very unlikely that this would be attained in a particular computation, but it does provide bounds. In the same spirit, a worst case analysis of the number of nodes HPA will expand with respect to a heuristic of given error bound can be performed. In order to achieve concrete results in a nontrivial case, the infinite undirected binary graph is used.

Let B be the infinite undirected binary graph:

$X = \{1, 2, 3, \ldots\}$

$E = \{(i, j) | j \in \Gamma(i) = \{2i, 2i + 1, \lfloor i/2 \rfloor\}\}$, $\lfloor a \rfloor \equiv$ the largest integer less than a

THEOREM 7. *Let k be the distance from the root node ($x = 1$ in the binary tree) to the goal node in B and let h be of bounded error ε (an integer). Then, for $\omega = \frac{1}{2}$, the worst case search will expand:*

$$2^{\varepsilon}k + 1 \quad \textit{nodes}\ [19]$$

Proof. Theorem 6 gives the case for $\varepsilon = 0$, which is $k + 1$. In the tree case, it is proved [19] that labeling nodes on the solution path $h_p + \varepsilon$ and off the solution path $h_p - \varepsilon$ gives the worst efficiency for HPA with an h having bounded error ε. Then the count of nodes expanded gives

$$\underset{\varepsilon = 0}{k + 1} + \underset{\varepsilon = 1}{k + 2k} + \underset{\varepsilon = 2}{2^2k} + \ldots + \underset{\varepsilon = 3}{2^{\varepsilon - 1}k}$$

$$= 1 + 2^{\varepsilon}k \qquad \text{Q.E.D.}$$

Similarly the worst case can be found for $\omega = 1$.

THEOREM 8. *As in Theorem 7, except that $\omega = 1$. Then the worst case search will expand:*

$$\begin{cases} k + 1 & \varepsilon = 0 \\ \dfrac{4^{\varepsilon}}{2}k + 1 & \varepsilon \geq 1 \end{cases} \quad [19]$$

The values $\omega = \frac{1}{2}$ and $\omega = 1$ were selected because they are the extremum and the behavior at the intermediate values is monotonic. If $\omega < \frac{1}{2}$, then a problem of sufficient length k could be found such that the broadening of the search would make it more expensive then for $\omega \geq \frac{1}{2}$ regardless of the error bound. The spirit of the result above is the same for general problem

graphs with a unique solution path of interest; only the counting would be more difficult.

THEOREM 9. *If* HPA *is searching a graph G with a unique solution path, then*

(i) *for any given h, $\omega = \frac{1}{2}$ will visit at most the same number of nodes as $\omega = 1$ in the worst case;*

(ii) *if h_1 and h_2 are of bounded error ε_1, ε_2 with $\varepsilon_2 > \varepsilon_1$, then, in the worst case for a given ω, h_1 will visit at most the same number of nodes as h_2* [19].

4. Some Final Observations

The last results refer to unique solution paths, but most problem spaces have many alternative solution paths. It is also not clear how reasonable are the results that bound the worst case. The latter question has its parallel in statistical error analysis versus worst case in numerical algorithms. The problem in characterizing statistical error in a graph space is very difficult insofar as the metrics are not well behaved and graph counting problems are difficult. Nevertheless, the model remains as a useful guide in developing heuristic procedures. Empirical performance measures can be collected to decide on appropriate weightings and features. Consider for a given heuristic function and problem the variables

N_ω = number of nodes expanded

k_ω = length of the solution path found

ζ_ω = branch rate of the search tree [3]

The branch rate is a derived quantity which gives the average degree of a tree of N nodes and diameter k_ω:

$$N_\omega = \frac{\zeta_\omega}{\zeta_\omega - 1}\left(\zeta_\omega^{k_\omega} - 1\right) \tag{4.1}$$

Empirically for the 15 puzzle with several heuristic functions used, increasing ω increases the path length of solutions and decreases the branch rate. In the ideal case using $h = h_p$ the exhaustive search ($\omega = 0$) finds the shortest path by expanding all nodes within that radius. As ω increases, the number of nodes decreases until at $\omega = \frac{1}{2}$ only the nodes along the shortest path are visited. This remains the case for $\frac{1}{2} < \omega \leqslant 1$. Where h is a less than perfect function, $\omega = 0$ still finds the shortest path. This is guaranteed to continue if $[\omega/(1 - \omega)]h \leqslant h_p$, the lower bound condition, is satisfied. However, when larger values of ω are used, more confidence is placed in the heuristic term. The searches continue to greater depth before being abandoned. The paths found can be quite circuitous and long. This type of search is very narrow in branching rate. As is seen in (4.1), the path length comes into the exponent of the branching rate. In either case, if one is halved the other can be approximately doubled, maintaining the same search performance.

Empirically ω can be varied and its performance checked on sample problems within a domain for a particular h. In general, automatic learning of the Samuel's type [10, 18] can be performed with respect to ω.

The evaluation function can be extended to a general linear form or convex combination,

$$f(x) = \left(1 - \sum_{i=1}^{k} \omega_i\right) g(x) + \sum_{i=1}^{k} \omega_i h_i(x)$$

$$\omega_i \geqslant 0, \qquad \sum_{i=1}^{k} \omega_i \leqslant 1$$

with even the ω_i as functions on the nodes. This would allow a learning system to measure performance of each term as a contribution to the accuracy of distance estimation. When the heuristics provide a lower bound to guarantee a shortest path, the weights could be adjusted to effect this; or, alternatively, a function of the form

$$f(x) = g(x) + \max_i (h_i(x))$$

where each $h_i(x) \leqslant h_p(x)$ could be used.

It remains to try more experiments that use these ideas in theorem proving, combinatorial problems, and general problem solving.

REFERENCES

1. Frazer, W. D., and McKeller, A. C. Samplesort: A sampling approach to minimal storage tree sorting. TR 74, Depart. of Electrical Engineering, Princeton University, Nov. 1968.
2. Winograd, S. On the number of multiplications required to compute certain functions. *Proc. Natl. Acad. Sci.* (5) **58** (Nov. 1967), 1840–1842.
3. Nilsson, N. Problem-solving methods in artificial intelligence, McGraw-Hill, New York (in preparation, Spring, 1971).
4. Newell, A., and Ernst, G. *The Search for Generality. Proceedings* 1965 *IFIP Congr.* Spartan Books, New York, 1965, pp. 17–24.
5. Michie, D. Heuristic search. Experimental Programming Reports: No. 19, Dept. of Machine Intelligence and Perception, University of Edinburgh, Edinburgh, Scotland, Dec. 1969.
6. Sandewall, E. Concepts and methods for heuristic search. *Proc. Internat. Joint Conf. Artificial Intelligence* (Walker, D., and Norton, L. eds.). MITRE Corporation, Bedford, Mass., 1969, pp. 199–218.
7. Newell, A., and Simon, H. GPS, a program that simulates human thought. *Computers and Thought* (Feigenbaum, E., and Feldman, J., eds.). McGraw-Hill, New York, 1963, pp. 279–293.
8. Newell, A., Shaw, J. C., and Simon, H. Empirical explorations with the logic theory machine. *Proc. Western Joint Computer Conf.* **11** (1957), 218–239.
9. Doran, J., Michie, D. Experiments with the Graph Traverser program. *Proc. Roy. Soc. A* (1437) **294** (Sept. 1966), 235–259.

10. Michie, D., and Ross, R. Experiments with the adaptive graph traverser *Machine Intelligence* 5 (1969), 301–318.
11. Nilsson, N. A mobile automation: An application of artificial intelligence techniques. *Proc. IJCAI* (Walker, D., and Norton, L., eds.). MITRE Corporation, Bedford, Mass., 1969, pp. 509–520.
12. Hart, P., Nilsson, N., and Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. System Sci. Cybernetics* (2) 4 (July 1968), 100–107.
13. Lawler, E., and Wood, D. Branch-and-bound methods: A survey. *Operations Research* (4) 14 (July–August 1966), 699–719.
14. Pohl, I. A theory of bi-directional search in path problems. RC 2713, IBM Research Center, Yorktown Heights, N.Y., Nov. 1969.
15. Dreyfus, D. An appraisal of some shortest path algorithms. *Operations Research* (3) 17 (May–June 1969), 395–412.
16. Moore, E. The shortest path through a maze. *Proc. Internatl. Symp. Theory of Switching*, Part II, April 1957. Harvard University Press, Cambridge, Mass., 1959, pp. 285–292.
17. Kowalski, R. Search strategies for theorem-proving. *Machine Intelligence* 5 (1969), 181–202.
18. Samuel, A. Some studies in machine learning using the game of checkers. *Computers and Thought* (1963), 71–105.
19. Pohl, I. First results on the effect of error in heuristic search. *Machine Intelligence* 5 (1969), 219–236.
20. Sandewall, E. A planning problem solver based on look-ahead in stochastic game trees. *J. ACM* (3) 16 (July 1969), 364–383.
21. Slagle, J., and Bursky, P. Experiments with a multipurpose theorem-proving heuristic program. *J. ACM* (1) 15 (Jan. 1968), 85–99.
22. Quinlan, J., and Hunt, E. A formal deductive problem solving system. *J. ACM* (4) 15 (Oct. 1968), 625–646.
23. Ernst, G. Sufficient conditions for the success of GPS. *J. ACM* (4) 16 (Oct. 1969), 517–533.
24. Wilkinson, J. *Rounding Errors in Algebraic Processes*, Prentice-Hall, Englewood Cliffs, N.J., 1963.

Accepted June, 1970