INFORMATION SCIENCES **34**, 47–59 (1984)

A Result on the Computational Complexity of Heuristic Estimates for the A* Algorithm†

MARCO VALTORTA

Department of Computer Science, Duke University, Durham, North Carolina 27706

Communicated by Azriel Rosenfeld

ABSTRACT

The performance of a new heuristic search algorithm is analyzed. The algorithm uses a formal representation (*semantic representation*) that contains enough information to compute the heuristic evaluation function $h(n)$, as defined in the context of A^*, without requiring a human expert to provide it. The heuristic is computed by solving less constrained subproblems (*auxiliary problems*) of the given problem. The new algorithm is shown to be less efficient than the Dijkstra algorithm, according to the complexity measure "number of node expansions." This proves that it is not efficient to compute heuristics for A^* by solving auxiliary problems with backtracking.

INTRODUCTION

Concluding "Problem Representation and Formal Properties of Heuristic Search," in which he describes the state-space approach to problem solving, Gordon Vanderbrug states: "Efforts [in the current research on heuristic search methods] are directed towards including semantic information by concentrating on the meaning of the symbols being manipulated" [26].

In this paper I describe a way to enrich the well-known graph formalism with more information from the problem domain, and an admissible search algorithm based on the new representation. The core of the paper is the analysis of the complexity of this new algorithm; specifically, I compare it with the classic Dijkstra algorithm [2].

† The writing of this paper was supported by the U.S. Air Force Office of Scientific Research under grant AFOSR-81-0221. The paper is based on work done at the Politecnico di Milano, Milan, Italy. A shorter version of this paper was presented at the Eighth International Joint Conference on Artificial Intelligence in Karlsruhe, F.R.G. [25].

52 Vanderbilt Ave., New York, NY 10017 0020-0255/84/$03.00

Kowalski [10] notes that every computable problem (in this paper, problem always means computable problem) can be formulated as the problem of finding a path through a graph. This approach, often referred to as the "state-space approach," is described in numerous books and articles on automated problem solving [23, 15, 26, 16]; moreover, there are well-studied connections between this approach and automata theory (see [12]).

In 1968, Hart, Nilsson, and Raphael published a paper [7] in which they described an algorithm for finding the shortest path in a graph from an initial node to a final node, using heuristic information to direct the search. This algorithm, called A^*, is *admissible*, in the sense that it always finds a solution of minimal cost if such a solution exists. The heuristic information (a function of the nodes in the graph having certain properties) is provided by an expert and is dependent on the problem domain. A^* has been used in various application domains ([14, 22] are just two examples chosen from two vastly different fields), and its complexity has been investigated in detail [20, 21, 13, 4, 8, 17, 24, 19].

A^* requires less time (on a sequential computer) than the Dijkstra algorithm [2] in solving a shortest-path problem, but it requires an expert to provide information not contained in the problem representation formalism. To avoid this, Marco Somalvico and other researchers at the Politecnico di Milano [12, 6, 24] have proposed a representation (the "semantic representation") which contains the information necessary to compute the heuristics used by A^*. Reference [5] summarizes the results of this research program through 1978. The reader who feels that the definitions and results of the following section are too sketchy is invited to read [5]. The facts given are also discussed in more detail in my thesis [24].

BASIC RESULTS

I start by giving two definitions of (formalized) problem:

A *syntactic problem* is a 5-tuple

$$P = (N, E, W, i, k),$$

where:

N is a set of states, called the state space;

E is a set of directed edges;

W is a set of nonnegative costs, greater than an arbitrary small constant δ, associated to each edge;

i is a distinguished member of N, the initial state;

k is a distinguished member of N, the final state.

Given a syntactic problem $P = (N, E, W, i, k)$, $M = (N, E, W)$ is called the *problem schema* of P. Note that M is a directed graph. A *solution* of a problem P is a path in the graph $G = (N, E, W)$ from i to k. Because of the restriction on the costs associated to the edges of M, M is sometimes called a δ-graph.

The *optimal solution* of a problem P is a path in the graph $G = (N, E, W)$ from i to k of minimal cost, where the cost of a path is the sum of the costs of its edges.

The notion of state in the (classic) syntactic formalism will now be enriched to define the "semantic" formalism.

Given a set $\underline{A}$ of attributes, and a set $\underline{V}$ of values for each attribute in $\underline{A}$, I define a *semantic* (or structured) *state*. A *semantic problem* is a 6-tuple

$$\underline{P} = (\underline{A}, \underline{V}, \Pi, \Lambda, \underline{i}, \underline{k}),$$

where

$\underline{A}$ is a set of attributes;

$\underline{V}$ is a set of values;

Π is a set of predicates (called *properties*, each indicated by π) whose domain is the set of all possible states;

Λ is a set of predicates of two arguments (called *legal conditions*, each indicated by λ), each one being one of the possible states;

$\underline{i}$ is a distinguished sequence of attribute-value pairs;

$\underline{k}$ is a distinguished sequence of attribute-value pairs;

A sequence of attribute-value pairs is called a *semantic* (or structured) *state*. A semantic state has a structure: the sequence of attribute-value pairs that constitute its meaning. In the classic, "syntactic" framework, instead, a state is just an atomic concept: all the information carried over from the problem domain is contained in the graph.

Two facts that support the view that the information contained in the "syntactic" graph does not permit an efficient search are the usefulness of the heuristic evaluation function used by A^* (which cannot be computed efficiently from the "syntactic" problem representation), and the success of expert systems (which rely heavily on domain-dependent heuristics).

Let us now consider how the structure of every semantic state is used to determine the *state space* and the *legal moves* in a semantic problem. The *candidate states* are all possible sequences of attribute-value pairs (I indicate them by an underlined letter, such as $\underline{n}$); the state space, $\underline{N}$, consists of all the candidate states which satisfy *all* the properties (the states in $\underline{N}$ are called *legal states*); the candidate moves are all possible pairs of legal states; the legal moves are all the moves $(\underline{n1}, \underline{n2})$ that satisfy all the legal conditions.

To every semantic problem one can associate a graph called the *skeleton* of the semantic problem. $G = (N, E, W)$ is defined by

$$N = \{ n \in N \mid (\forall \pi)((\pi \in \Pi) \Rightarrow \pi(n)) \}$$

$$E = \{ (n1, n2) \mid (\forall \lambda)((\lambda \in \Lambda) \Rightarrow \lambda(n1, n2)) \}$$

W is the set of costs associated to the edges in E. Each edge has a nonnegative cost greater than a small constant δ associated to it. (The method of determining the cost of each edge does not affect the results of this paper.)

The reader might find it useful to try to express the eight-tile puzzle [15] in the semantic problem formalism: define a set of attributes (the tiles), a set of values (the positions of the tiles on the board), a set of properties (which, by using the attributes and the values composing a state, tell when a state is legal), and set of legal conditions (which also use the rich structure of semantic states). A property might, for example, express that in a legal state (i.e., a permissible board configuration) no two attributes can have the same value (i.e., no two tiles can occupy the same position on the board). A legal condition might state that a move is legal only if the state after the move has one distinguished attribute (i.e., the one corresponding to the blank tile) whose value (i.e., its position) is different from the value of the same attribute in the state before the move (i.e., to sum up, a move is legal only if the space on the board is in a different position after the move than it was before the move). The reader can find a solution to this drill in [5], or in [12, 24]. These two latter reference also contain another example.

One could solve a semantic problem by solving the problem corresponding to its skeleton with the Dijkstra algorithm, but this method does not take any advantage from the extra information contained in the structure of the states. Reference [24] discusses some ways to exploit this information. One of them, algorithm M, will be discussed in the following. In order to introduce the algorithm of general applicability that I want to compare with the Dijkstra algorithm, a few more definitions and facts must be given. This is the purpose of the next section.

AUXILIARY PROBLEMS

Informally speaking, a problem is auxiliary to another one if it is less constrained. The notion of auxiliary problem is formalized on the following definition:

DEFINITION. A semantic problem

$$\underline{P}' = (\underline{A}', \underline{V}', \Pi', \Lambda', \underline{i}', \underline{k}')$$

is an *auxiliary problem* of

$$\underline{P} = (\underline{A}, \underline{V}, \Pi, \Lambda, \underline{i}, \underline{k})$$

if

$$\underline{A}' = \underline{A},$$

$$\underline{V}' = \underline{V},$$

$$\Pi' = \Pi,$$

$$\Lambda' \subset \Lambda,$$

$$\underline{i}' = \underline{i},$$

$$\underline{k}' = \underline{k},$$

(I indicate that $\underline{P}'$ is auxiliary to $\underline{P}$ by $\underline{P}' \Leftarrow \underline{P}$.)

The following theorem provides a basis for computing an heuristic evaluation function $h(n)$, as used by A^* to focus its search [16], from the information contained in the semantic representation of a computable problem. (In this paper I follow the convention used in [16] in that I indicate the heuristic estimate with $h(n)$ and its exact value with $h^*(n)$.)

THEOREM. *If $\underline{P}' \Leftarrow \underline{P}$, where the initial state for $\underline{P}'$ and $\underline{P}$ is $\underline{n}$, then the length of the optimal solution of $\underline{P}'$ is a possible value for an admissible heuristic estimate $h(n)$ for the problem $\underline{P}$.*

The reader should convince himself that the theorem is true, or check the summary given in [5] or the proof in [12].

The following section presents an algorithm built on the previous theorem.

ALGORITHM *M*

This algorithm is a special case of both algorithm *G* given in [5] and algorithm *S* in [24].

Input: a semantic problem $\underline{P}$.

Output: an optimal solution of $\underline{P}$.

Method: Solve the problem corresponding to the skeleton of $\underline{P}$ using A^*, where each necessary value of $h(\underline{n})$ is computed by solving an auxiliary problem

of

$$\underline{Q} = (\underline{A}, \underline{V}, \Pi, \Lambda, \underline{n}, \underline{k})$$

using the Dijkstra algorithm.

All the auxiliary problems have the same set of legal conditions— Λ'. This ensures that the "consistency" condition [15, 16] is satisfied for $h(n)$ computed by solving the auxiliary problems of $\underline{Q}$. This result is now shown to hold.

THEOREM. *If $h(m)$ and $h(n)$ are computed as costs of shortest paths from m to k and from n to k respectively on the skeleton of the same auxiliary problem, then $h(m) - h(n) \leqslant d(m,n)$, where $d(m,n)$ is the cost of the shortest path from m to n on the skeleton of the problem.*

Proof. The proof consists of a *reductio ad absurdum*. (Refer to Figure 1 while following the proof.) Assume that the consistency assumption is not satisfied, i.e. that

$$h(m) > h(n) + d(m,n).$$

Note that $d(m,n)$ is an upper bound on the length of the shortest path from m to n in the skeleton of the auxiliary problem. Therefore the shortest path from m to k in the skeleton of the auxiliary problem must pass through n. And if the

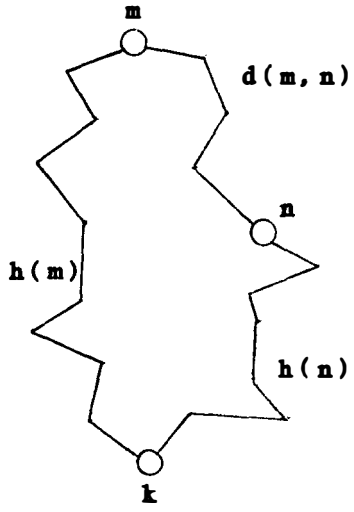


Fig. 1. Two estimates computed on the same graph satisfy the consistency assumption.

shortest path from m to k in the skeleton of the auxiliary problem passes through n, it must be that

$$h(m) \leqslant h(n) + d(m, n),$$

which can be rewritten as

$$h(m) - h(n) \leqslant d(m, n). \quad \blacksquare$$

I have shown in the previous section how to compute the heuristic function $h(n)$ from the information contained in the semantic representation. I can proceed to analyze the complexity of this totally automatic procedure. This is the purpose of the next section.

COMPLEXITY OF ALGORITHM M

In this section, I compare algorithm M with the Dijkstra algorithm. It would be senseless to compare M with the A^* algorithm, since, to focus its search, A^* relies on information [i.e., $h(n)$] that is outside the problem-representation formalism used (i.e., the syntactic graph). Dijkstra's algorithm can be considered a special case of A^* where $h(n) = 0$ for all n, i.e., no heuristic information is given.

I compare these algorithm according to the criterion "number of node expansions," which is discussed and generally accepted in the published literature [15, 13]. A remarkable shortcoming of this criterion in our case is that it considers the cost of expanding a node in an auxiliary problem to be the same as the cost of expanding a node in the original problem. Still, I think that the result I obtain using this approach is sufficiently interesting to justify the simplification.

In order to prove our fundamental result, I shall make use of some results which can be found in [24, 13, 4]. I now recall these results in a compact form.

Let a (directed) graph $G = (N, E, W)$ be given. Let $g(n)$ be the length of the path from node i, the initial node, to node n, in graph G, passing through already expanded nodes. [This is the "standard" definition of $g(n)$, as given in most of the referenced literature.]

FACT 1. *The Dijkstra algorithm will find a shortest path in G by expanding only the nodes n that satisfy the following inequality*:

$$g(n) < h^*(i). \tag{1}$$

[Note that here and in Fact 2, $h^*(i)$ is the shortest path from i to k because of the definition of $h^*(n)$.]

FACT 2. *The A* algorithm will find a shortest path in G by expanding only the nodes n that satisfy the following inequality*:

$$g(n)+h(n) < h^*(i) \tag{2}$$

and some of the nodes that satisfy

$$g(n)+h(n) = h^*(i). \tag{3}$$

I define the distance from node m to node p in the graph $G=(N, E, W)$ to be the length of the shortest path from m to p in G. (If no path from m to p exists in G, then the distance is conventionally assumed to be infinite.) I can now prove the following result:

MAIN THEOREM. *Let a semantic problem*

$$\underline{P}=(\underline{A}, \underline{V}, \Pi, \Lambda, \underline{i}, \underline{k})$$

be given. To solve the problem $\underline{P}$, *algorithm M expands at least every node expanded by the Dijkstra algorithm to solve the syntactic problem corresponding to its skeleton.*

A corollary to this theorem descends from the fact that algorithm M computes consistent and admissible estimates: algorithm M uses at least the same number of node expansions as the Dijkstra algorithm. I can conclude this because the number of node expansions and the number of expanded nodes are the same when consistent and admissible estimates are used in the A algorithm* [15, 16].

Proof of the theorem. The proof is long, but straightforward.

Algorithm M expands nodes in two phases:

(a) to compute $h(n)$;
(b) to solve $\underline{P}$, with the same strategy used by A^*.

The estimates computed by M, of the form $h(n)$, can be divided in three classes; I shall therefore consider three cases, and show that for each case the computation of the estimate plus the solution of the problem using it is more expensive than the solution of the problem by using the Dijkstra algorithm, which does not require any estimate to be computed.

The first two cases are very simple.

Case 1. The estimate $h(n)$ *does not disallow node n to be expanded in phase* (b). An example is given in Figure 2. The computation of the heuristic, in this

case, does not allow us to save even a single node while using it in phase (b). Since to compute $h(n)$ by solving an auxiliary problem one needs to expand at least a node (in the nontrivial case in which n is the final node, when it is obviously unuseful to compute the heuristic!), it would have been better not to compute the estimate at all in the first place.

Case 2. *The estimate $h(n)$ is such that n is not expanded because*

$$g(n)+h(n) \geq h^*(i).$$

(Note that if the above is true with " $>$ ", node n will not be expanded for sure; if it is true with " $=$ ", it might.) If the only effect of $h(n)$ is that node n will not be expanded, the cost of the estimate computation in phase (a), which necessitates at least the expansion of node n itself, is not sufficient to compensate the saving arising from not expanding n in phase (b).

Case 3. *There are nodes g_i that are expanded by the Dijkstra algorithm, but are not expanded by algorithm M in phase* (b) *because, in order to be expanded, they have to be reached through a node m whose estimate $h(m)$ is so large that m is not expanded in phase* (b). Figure 3 provides an example of this case: note that $h(m)$ is so large that the nodes g_i are not expanded in phase (b) of algorithm M.

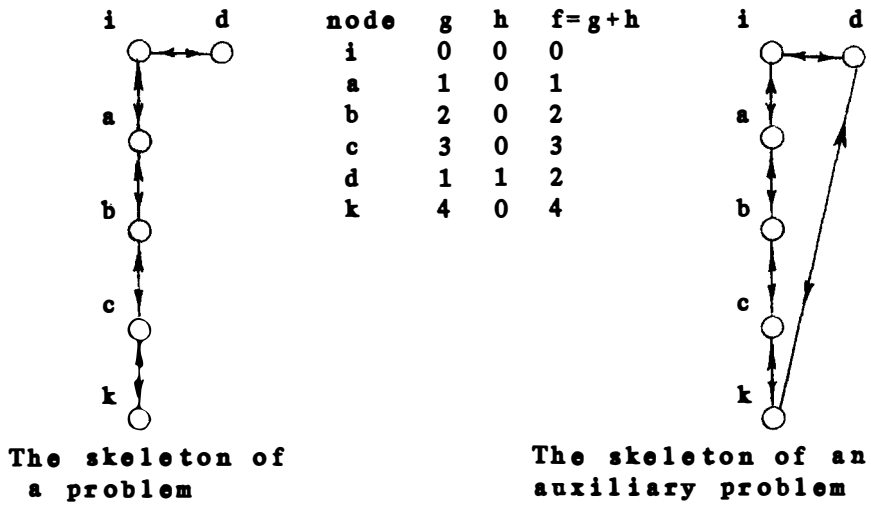


Fig. 2. The estimate for d is not useful.

The nodes g_i are, at most, the ones for which the following holds:

$$g(m) + d(m, g_i) < h^*(i), \tag{4}$$

which can be rewritten as follows:

$$d(m, g_i) < h^*(i) - g(m). \tag{5}$$

This is the property that characterizes the set of nodes that, at most, are not expanded if $h(m)$ is large enough to avoid expanding m.

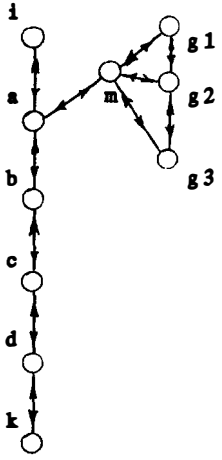
By Fact 2, m is not expanded if $h(m)$ is at least so large that the following holds:

$$g(m) + h(m) = h^*(i), \tag{6}$$

which can be rewritten as

$$h(m) = h^*(i) - g(m). \tag{7}$$

Since $h(m)$ is computed, in phase (b), by solving an auxiliary problem of P using the Dijkstra algorithm, one must expand, according to Fact 1, all the nodes at distance less than $h(m)$ from m on the skeleton of the auxiliary



node	g	h	f = g + h
i	0	0	0
a	1	0	1
b	2	0	2
c	2	0	2
m	2	4	6
g1	3	(h is not computed)	
g2	3	(h is not computed)	
g3	3	(h is not computed)	
d	4	0	4
k	5	0	5

Fig. 3. $h(m)$ is so large that $h(g1)$, $h(g2)$, $h(g3)$ are not computed. The figure shows the skeleton of a problem. It does not show the skeleton of the auxiliary problem used to compute $h(m)$. (Can you draw it?)

problem.

But we know that $h(m)$ is at least so large that (7) holds. Therefore, at least the nodes at distance less than $h^*(i)-g(m)$ from m in the auxiliary problem must be expanded. *A fortiori*, since the distance from i to m in the auxiliary problem is not greater than the distance from i to m in P, at least the nodes at distance less than $h^*(i)-g(m)$ from m in P must be expanded. Therefore, at least the nodes satisfying the following inequality must be expanded by algorithm M in phase (b):

$$d(m, h_i) < h^*(i) - g(m). \tag{8}$$

By comparing (8) with (5), one concludes that, even in the most favorable case, the set of nodes g_i which are not expanded in phase (a) because of the computation of the estimate $h(m)$ in phase (b) is a subset of the set of the nodes h_i expanded to compute the estimate in phase (b). Therefore, even in this last case, it is better not to compute the heuristic at all, but to solve P by using the Dijkstra algorithm directly. ■

CONCLUSION

In this conclusion, I state two definitions and a theorem, and I present an interpretation of the Main Theorem.

DEFINITION. An algorithm to find the minimum-cost path in a graph is *blind* if it relies only on the information contained in the syntactic graph to find the minimum-cost path (i.e., blind algorithms do not use heuristic information).

DEFINITION. An algorithm to find the minimum-cost path in a graph is *unidirectional* if it expands nodes at nondecreasing distances from the initial node.

(This definition is arbitrary: what should one call algorithms which expand nodes at increasing distances from the final node?)

The following result can be shown to hold:

THEOREM. *The Dijkstra algorithm is the algorithm that uses the least number of node expansions among blind, unidirectional, deterministic algorithms.*

Proof. The proof of this result consists of an "adversary" (or "oracle") based argument. Assume that another algorithm, B, can find a shortest path from i to k without expanding a node n for which the following holds:

$$g(n) < h^*(i). \tag{8}$$

Then the adversary can find a problem such that there is an edge from node n to node f of such small cost that the minimum-cost path from i to f passes through n. This means that algorithm B does not find the minimum-cost solution. ■

The above result, together with the Main Theorem, indicates that it is not efficient to compute heuristics by solving auxiliary problems with a trial-and-error strategy (i.e., a strategy involving backtracking).

Recognizing that an auxiliary problem can be solved by means of a method that does not require backtracking seems to be an extremely difficult task, strictly related to the "change of representation" problem [1], which is considered to be beyond the state of the art. (See, for example [11, pp. 237–241].) Even auxiliary problems whose solution leads to the computation of simple heuristics do not display any apparent structure (as far as their skeleton is concerned) which may lead to their simple solution. An interesting example of this phenomenon is described in [12, 24], where the auxiliary problem whose solutions compute the heuristic "number of misplaced tiles" for the eight-tile puzzle is presented. This heuristic is described in [15, 16].

RELATED RESEARCH

Judea Pearl and the late John Gashnig have discovered, independently of the Milan team, that admissible heuristics for A^* can be computed by solving auxiliary problems. Judea Pearl calls the auxiliary problems "relaxed models." John Gashnig calls them "edge supergraphs" [3]. Gashnig uses the syntactic formalism, and he does not propose an algorithm that finds auxiliary problems automatically, the way algorithm M does, thanks to the "semantic" formalism.

Judea Pearl [17] and Dennis Kibler [9] have postulated the need for changing the representation paradigm to solve auxiliary problems efficiently. Their postulation is grounded on the negative result discussed in this paper. They quote this result explicitly in their reports [17, p. 131; 9, p. 4].

REFERENCES

1. Saul Amarel, On representations of problems of reasoning about actions, in *Machine Intelligence 3*. Edinburgh U. P. Edinburgh, 1968, pp. 131–171.
2. Edger W. Dijkstra, A note on two problems in connection with graphs, *Numer. Math*. 1: 269–271 (1959).
3. John Gashnig, A problem similarity approach to devising heuristics: First results, in *Proceedings of the 6th International Joint Conference on Artificial Intelligence*, 1979, pp. 301–307.
4. David Gelperin, On the Optimality of A^*, *Artificial Intelligence* 8: 69–76 (1977).
5. Giovanni Guida and Marco Somalvico, A method for computing heuristics in problem solving, *Inform. Sci*. 19: 251–259 (1979).

6. Giovanni Guida, Marco Somalvico, and Marco Valtorta, "Alcune proprietà algebriche dei problemi ausiliari: Un contributo alla teoria dei problemi, (in Italian), in *Proceedings of AICA 1980*, Bari, Italy, pp. 177–193.
7. Peter A. Hart, Nils J. Nilsson, and Bertram Raphael, A formal basis for the heuristic determination of minimal cost paths, *IEEE Trans. Systems Sci. Cybernet.* 4(2): 100–107 (July 1968).
8. Nam Huyn, Rina Dechter, and Judea Pearl, Probabilistic analysis of the complexity of *A**, *Artificial Intelligence* 15: 241–254 (1980).
9. Dennis Kibler, Natural generation of admissible heuristics, Technical Report TR-188, Information and Computer Science Dept. Univ. of California at Irvine, Irvine, Calif., 1982.
10. Robert A. Kowalski, *Logic for Problem Solving*, North-Holland, Amsterdam, 1979.
11. Douglas B. Lenat, The nature of heuristics, *Artificial Intelligence* 19(2): 189–249 (Oct. 1982).
12. Dino Mandrioli, Alberto Sangiovanni Vincentelli, and Marco Somalvico. Toward a theory of problem solving, in *Topics in Artificial Intelligence* (A. Marzollo, Ed.), Springer, Vienna, 1976.
13. Alberto Martelli, On the complexity of admissible search algorithms, *Artificial Intelligence* 8(1): 1–13 (1977).
14. Ugo Montanari, Heuristically guided search and chromosome matching, *Artificial Intelligence* 1(4): 227–245 (1970).
15. Nils J. Nilsson, *Problem Solving Methods in Artificial Intelligence*, McGraw-Hill, New York, 1971.
16. Nils J. Nilsson, *Principles of Artificial Intelligence*, Tioga, Palo Alto, Calif., 1980.
17. Judea Pearl, The utility of precision in search heuristics, Memo UCLA-ENG-CSL-8065 (Revision II), Cognitive Systems Laboratory at the School of Engineering and Applied Science, UCLA, Dec. 1980.
18. Judea Pearl, On the discovery and generation of certain heuristics, *UCLA Comput. Sci. Dept. Quart.* 10(2): 121–132 (Spring 1982).
19. Judea Pearl, *Heuristics: Intelligence Search Strategies for Computer Problem Solving*, Addison-Wesley, Reading, Mass., 1984.
20. Ira Pohl, First results on the effect of error in heuristic search, in *Machine Intelligence 5*, Edinburgh U. P., Edinburgh, 1970, pp. 219–236.
21. Ira Pohl, Practical and theoretical considerations in heuristic search algorithms, in *Machine Intelligence 8*, Edinburgh U.P., Edinburgh, 1977. pp. 55–72.
22. Alberto Sangiovanni-Vincentelli and Mauro Santomauro, A heuristic guided algorithm for optimal backboard ordering, in *Proceedings of the 13th Annual Allerton Conference on Circuit and System Theory*, 1975, pp. 916–921.
23. Alberto Sangiovanni-Vincentelli and Marco Somalvico, Formulazione teorica del metodo dello spazio degli stati per la risoluzione automatica dei problemi (in Italian), *Alta Frequenza*, Mar. 1975.
24. Marco Valtorta, Un contributo alla teoria della risoluzione dei problemi: Rappresentazione semantica, proprietà algebriche e algoritmi di ricerca (in Italian), Tesi di Laurea, Istituto di Ingegneria Elettrotecnica ed Elettronica, Politecnico di Milano, Milan, Italy, 1980.
25. Marco Valtorta, A result on the computational complexity of heuristic estimates for the *A** algorithm, in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, 1983, pp. 777–779.
26. Gordon J. Vanderbrug, Problem representations and formal properties of heuristic search, *Inform. Sci.* 11: 279–307 (1976).

Received 25 June 1984; revised 30 July 1984.