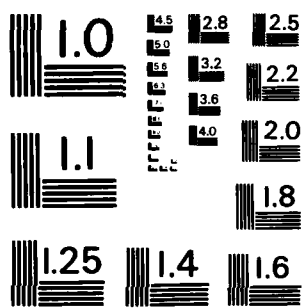MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU OF STANDARDS-1963-A

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR- 83-0343	2. GOVT ACCESSION NO. $AD\ A127\ 901$	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* A RESULT ON THE COMPUTATIONAL COMPLEXITY OF HEURISTIC ESTIMATES FOR THE A* ALGORITHM		5. TYPE OF REPORT & PERIOD COVERED TECHNICAL
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(*s*) Marco Valtorta		8. CONTRACT OR GRANT NUMBER(*s*) AFOSR-81-0221
9. PERFORMING ORGANIZATION NAME AND ADDRESS Computer Science Department Duke University Durham NC 27706		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS PE61102F; 2304/A2
11. CONTROLLING OFFICE NAME AND ADDRESS Mathematical & Information Sciences Directorate Air Force Office of Scientific Research Bolling AFB DC 20332		12. REPORT DATE Jan 83
		13. NUMBER OF PAGES 24
14. MONITORING AGENCY NAME & ADDRESS(*if different from Controlling Office*)		15. SECURITY CLASS. *(of this report)* UNCLASSIFIED
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*
Approved for public release; distribution unlimited.

17. DISTRIBUTION ST. 4ENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY TES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*
Heuristic search; problem representation; auxiliary problems; computational complexity.

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*
The performance of a new heuristic search algorithm is analyzed. The algorithm
uses a formal representation that contains enough information to compute the
heuristic evaluation function h(n), without requiring a human expert to provide
it. The new algorithm is shown to be less efficient than the Dijkstra algorithm,
according to the complexity measure "number of node expansion." Researchers
attempt an interpretation of this strong negative result. Other properties of
the new algorithm are discussed in references [Valt80, GSoma79, GSValt80]. A
short discussion of related research, some of which is in progress, (CONTINUED)

DD $_{1\ JAN\ 73}^{FORM}$ 1473

83 - 05 06 - 112

ITEM #20, CONTINUED: concludes the paper.

Accession For		
NTIS GRA&I	☒	
DTIC TAB	☐	
Unannounced	☐	
Justification		
By		
Distribution/		
Availability Codes		
Dist	Avail and/or Special	
A		

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, USA

Telephone Number: (919) 684-5110.

Submission Category: Long Paper.

Preferred Subarea: Search.

Length of Paper: 4790 words, all included.

Abstract

The performance of a new heuristic search algorithm is
analyzed in this paper. The algorithm uses a formal
representation that contains enough information to compute
the heuristic evaluation function h(n), without requiring a
human expert to provide it. The new algorithm is shown to
be less efficient than the Dijkstra algorithm, according to
the complexity measure "number of node expansions." I at-
tempt an interpretation of this strong negative result.
Other properties of the new algorithm are discussed in
references [Valt80, GSoma79, GSValt80]. A short discus-
sion of related research, some of which is in progress, con-
cludes the paper.

DTIC
ELECTE
MAY 10 1983
D
D

83 05 06-112

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 are directed towards including semantic information by concentrating on the meaning of the symbols being manipulated"[Vander76].

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 to the classic Dijkstra [Dijk59] algorithm.

Kowalski [Kowa79] 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 [Soma73, Nils71, Vander76, Nils80]; moreover, there are well-studied connections between this approach and dynamic systems theory, in particular automata theory (see [MVSoma76]).

In 1968, Hart, Nilsson, and Raphael published a paper [HNRaph68] 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 for "practical" purposes [Mont70], and its complexity has been investigated [Pohl70, Pohl77, Mart77, Gelper77, HDPear80, Pear80, Valt80].

A* requires less time than the Dijkstra algorithm [Dijk59] 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 [MVSoma76, GSValt80, Valt80] have proposed a representation (the "semantic representation") which contains the information necessary to compute the heuristics used by A* [GSoma79]. I direct the reader to the just-mentioned paper, which logically precedes this one, for a description of the semantic representation. In the following section, I shall give some definitions and results in a slightly informal and sketchy way. The facts given are discussed in more detail in the referenced literature and in my thesis in particular [Valt80].

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 of attributes-- A, and a set of values for each attribute in A-- V, I define a semantic (or structured)

state.

A semantic problem is a 6-tuple

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

where:

A is a set of attributes;

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 all possible states;

i is a distinguished sequence of attribute-value pairs;

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:

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

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

W is the set of costs associated to the edges in E. Each edge has a non negative 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 [Nils71] 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), a 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 [GSoma79], or in [MVSoma76, Valt80]. These two latter references 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 [Valt80] 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 to 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 in the following 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

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

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

$$\Pi' = \Pi ;$$

$$\Lambda' \subseteq \Lambda ;$$

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

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

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

The following theorem provides a basis for computing an heuristic evaluation function ($h(n)$), as used by A* to focus its search [Nils80], from the information contained in the semantic representation of a computable problem. (In this paper I follow the convention used in [Nils80] in that I indicate the heuristic estimate with $h(n)$ and its exact value with $h^*(n)$.)

Theorem. If $\underline{P}' <= \underline{P}$, 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 $h(\underline{n})$ for the problem $\underline{P}$.

The reader should convince himself that the theorem is true, or check the summary given in [GSoma79] or the proof in[MVSoma76].

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 [GSoma79] and algorithm S in [Valt80].

Input: a semantic problem P.

Output: an optimal solution of 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" condi-

tion [Nils71,Nils80] 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) <= 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." Keep an eye on figure 1 while following the proof
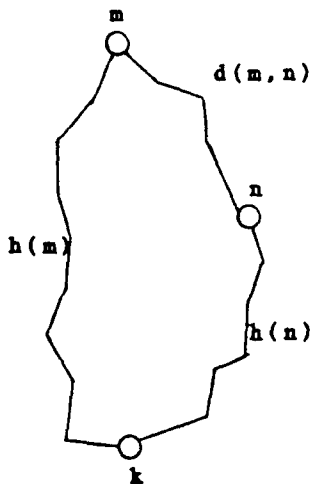

Figure 1: Two estimates computed on the same graph satisfy the consistency assumption.

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 would pass through n. And if the shortest path from m to k in the skeleton of the auxiliary problem passes through n, it must be that:

h(m) <= h(n) + d(m,n)

that can be rewritten as:

h(m) - h(n) <= d(m,n).

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 to the Dijkstra algorithm. It would be senseless to compare M to 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).

I compare these algorithms according to the criterion "number of node expansions," which is discussed and generally accepted in the published literature [Nils71, Mart77]. 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 use this simple criterion.

In order to prove our fundamental result, I shall make use of some results, which can be found in [Valt80, Mart77, Gelper77]. 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:

(1) $g(n) < h^*(i)$.

(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:

(2) $g(n) + h(n) < h^*(i)$

and some of the nodes that satisfy:

(3) $g(n) + h(n) = h^*(i)$.

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:

<u>Main</u> <u>Theorem</u>. Let a semantic problem

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

be given. To solve 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 [Nils71,Nils80].

<u>Proof</u> 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 $\underline{n}$ to be ex-

panded in phase (b). An example is given in figure 2 below.

node	g	h	f=g+h
i	0	0	0
a	1	0	1
b	2	0	2
c	3	0	3
d	1	1	2
k	4	0	4
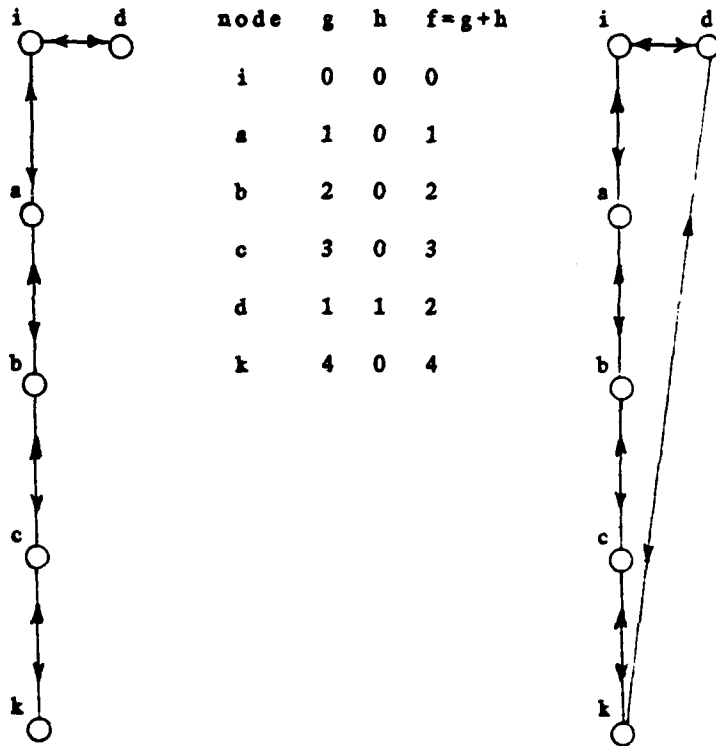
The skeleton of
a problem

The skeleton of an
auxiliary problem

Figure 2: The estimate for d is not useful.

The computation of the heuristic, in this case, does
not allow 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 non trivial 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 be-
cause

$g(n) + h(n) >= h*(i)$.

(Note that if the above is true with ">", node $\underline{n}$ 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 neces-
sitates at least the expansion of node $\underline{n}$ itself, is not suf-
ficient to compensate the saving arising from not expanding
$\underline{n}$ in phase (b).

Case 3

There are nodes $\underline{gi}$ that are expanded by Dijkstra algo-
rithm, but are not expanded by the M algorithm in phase (b)
because, in order to be expanded, they should be reached
through a node $\underline{m}$ whose estimate h(m) is so large that m is
not expanded in phase (b).

The following figure 3 provides an example of this
case: note that h(m) is so large that the gi nodes are not

expanded in phase (b) of the M algorithm.
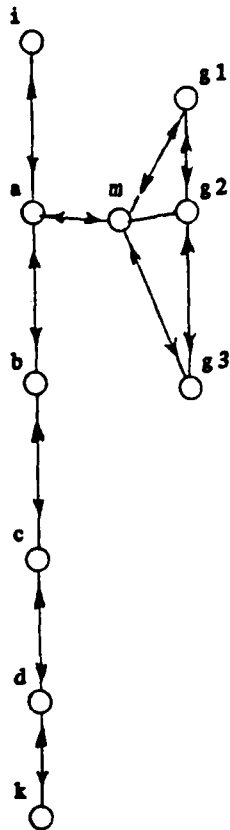
node	g	h	f = g + h
i	0	0	0
a	1	0	1
b	2	0	2
c	2	0	2
m	2	4	6
g1	3	(h is not computed)	
g2	3	(h is not computed)	
g3	3	(h is not computed)	
d	4	0	4
k	5	0	5

Figure 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?)

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

(3) g(m) + d(m,gi) < h*(i)

that can be rewritten as follows:

(4) $d(m,gi) < h*(i) - g(m)$.

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

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

(5) $g(m) + h*(m) = h(i)$

that can be rewritten as:

(6) $h(m) = h*(i) - g(m)$.

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 problem.

But we know that $h(m)$ is at least so large that (6) 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 of 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, the nodes expanded by the M algorithm in phase (b) satisfy the following inequality:

(7) $d(m,hi) < h^*(i) - g(m)$.

By comparing (7) with (4), one concludes that, even in the most favorable case, the set of nodes gi 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 (hi) expanded to compute the estimate in phase (b).

Therefore, even in this last case, it is better not to compute the heuristic at all and solve P by using the Dijkstra algorithm directly.

$\square$

Conclusion

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

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).

An algorithm to find the minimum cost path in a graph is _unidirectional_ if it expands nodes at non decreasing 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, uni-directional, deterministic algorithms.

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:

(8) g(n) < h*(i)

Then, the adversary can find a problem such that there is an edge from node n to node f of such a small cost that the minimum cost path from i to f passes through n.

This means that the B algorithm 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 [Amarel68], which is con-

sidered to be beyond the state of the art. (See, for example
[Lenat82, pp. 237-241].) Even auxiliary problems whose solu-
tion 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 in-
teresting example of this phenomenon is described in [MVSo-
ma76, Valt80], 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 [Nilss71, Nilss80].

Related Research

Judea Pearl and the late John Gashnig have discovered,
independently from the Milan team, that admissible heuris-
tics for A* can be computed by solving auxiliary problems.
Judea Pearl calls the auxiliary problems "relaxed models."
John Gashnig calls them "edge supergraphs" [Gashn79]. Gash-
nig 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 and Dennis Kibler[Kibler82] have postulated
the need for changing representation paradigm to solve auxi-
liary problems efficiently. Their postulation is grounded on
the negative result discussed in this paper. They quote this

result explicitly in their reports [Pearl82, p.131; Kibler82,p.4].

References

[Amarel] Amarel, Saul. "On Representations of Problems of Reasoning About Actions." Machine Intelligence 3, Ed. D. Michie. Edinburgh: Edinburgh University Press, 1968, 131-171.

[Dijks59] Dijkstra, Edger W. "A Note on two Problems in Connection with Graphs." Numerische Matematik. 1 (1959), 269-271.

[Gashn79] Gashnig, John. "A Problem Similarity Approach to Devising Heuristics: First Results." Proc. 6th IJCAI, (1979) 301-307.

[Gelp77] Gelperin, David. "On the Optimality of A*." Artificial Intelligence, 8 (1977), 69-76.

[GSoma79] Guida, Giovanni and Marco Somalvico. "A Method for Computing Heuristics in Problem Solving." Information Sciences, 19 (1979), 251-259.

[GSValt80] Guida, Giovanni, Marco Somalvico and Marco Valtorta. "Alcune Proprieta' Algebriche dei Problemi Ausiliari: Un Contributo alla Teoria dei Problemi."

Proceedings of AICA 1980, Bari, Italy, 177-193 (in Italian).

[HNRaph68] Hart, Peter A., Nils J. Nilsson, and Bertram Raphael. "A Formal Basis for the Heuristic Determination of Minimal Cost Paths." IEEE Transactions on Systems Science and Cybernetics, 4, 2 (July 1968), 100-107.

[HDPear80] Huyn, Nam, Rina Dechter and Judea Pearl. "Probabilistic Analysis of the Complexity of A*." Artificial Intelligence, 15 (1980), pp. 241-254.

[Kibler82] Kibler, Dennis. "Natural Generation of Admissible Heuristics." Technical Report TR-188, Information and Computer Science Department, University of California at Irvine, Irvine, California, 1982.

[Kowa79] Kowalsky, Robert A. Logic for Problem Solving. Amsterdam: North-Holland, 1979.

[Lenat82] Lenat, Douglas B. "The Nature of Heuristics." Artificial Intelligence, 19, 2 (October 1982), 189-249.

[MVSoma76] Mandrioli, Dino, Alberto Sangiovanni Vincentelli and Marco Somalvico. "Toward a Theory of Problem Solving." Topics in Artificial Intelligence. Vienna: Springer, 1976.

[Mart77] Martelli, Alberto. "On the Complexity of Admissible Search Algorithms." Artificial Intelligence, 8, 1 (1977), pp. 1-13.

[Mont70]   Montanari, Ugo. "Heuristically Guided Search and
    Chromosome Matching." Artificial Intelligence, 1, 4
    (1970), pp. 227-245.

[Nils71] Nilsson, Nils J.   Problem Solving Methods in Artif-
    icial Intelligence. New York: McGraw-Hill, 1971.

[Nils80] Nilsson, Nils J. Principles of Artificial Intelli-
    gence.   Palo Alto (Calif.): Tioga Publishing Company,
    1980.

[Pearl80]   Pearl, Judea. "The Utility of Precision in Search
    Heuristics." Memo UCLA-ENG-CSL-8065 (Revision II), Cog-
    nitive Systems Laboratory at the School of Engineering
    and Applied Science, UCLA, December 1980.

[Pearl82] Pearl, Judea. "On the Discovery and Generation of
    Certain Heuristics." The UCLA Computer Science Depart-
    ment Quarterly, 10, 2 (Spring 1982), 121-132.

[Pohl70] Pohl, Ira. "First Results on the Effect of Error in
    Heuristic Search." Machin Intelligence 5. Edinburgh:
    Edinburgh University Press, 1970, pp.219-236.

[Pohl77] Pohl, Ira. "Practical and Theoretical Considera-
    tions in Heuristic Search Algorithms." Machine Intelli-
    gence 8. Edinburgh: Edinburgh University Press, 1977,
    pp.55-72.

[VSoma75] Sangiovanni Vincentelli, Alberto and Marco Somal-
    vico.   "Formulazione Teorica del Metodo dello Spazio

degli Stati per la Risoluzione Automatica dei Problemi." _Alta Frequenza_, March 1975 (in Italian).

[Valt80] Valtorta, Marco. "Un Contributo alla Teoria della Risoluzione dei Problemi: Rappresentazione Semantica, Proprieta' Algebriche e Algoritmi di Ricerca." Tesi di Laurea, Istituto di Ingegneria Elettrotecnica ed Elettronica, Politecnico di Milano, Milan, Italy, 1980 (in Italian).

[Vander76] Vanderbrug, Gordon J. "Problem Representations and Formal Properties of Heuristic Search." _Information Sciences_, 11 (1976), pp.279-307.

DATE
ILME