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

ABSTRACT

The performance of a new heuristic search algorithm is analyzed in this paper. 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.

Basic Definitions and Auxiliary Problems

A syntactic problem is a 5-tuple

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

where:
N is a set of states (or nodes), called the state-space;
E is a set of directed edges;
W is a set of positive costs, greater than an arbitrary small constant, associated to each edge;
i is a distinguished member of N, the initial state;
k is a distinguished member of N. the final state.

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.

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;
II is a set of predicates (called properties, each indicated by π) of one argument, whose domain is the set of all possible states;
A is a set of predicates (called legal conditions, each indicated by λ). of two arguments, 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 (or node). 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.

The writing of this paper was supported by the 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.

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 (indicated by a letter in italics, such as n); the state space, N , consists of all the candidate states which satisfy all the properties (the states in N are called legal states); the candidate moves are ail possible pairs of legal states; the legal moves are all the moves (n1 , n2) that satisfy all the legal conditions. Therefore, to every semantic problem one can associate a graph called the skeleton of the semantic problem. The structure of the states is lost in passing from a semantic problem to its skeleton.

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 [Valtorta, 1980] discusses some ways to exploit this information. One of them, algorithm M, will be discussed below. Before introducing it, I present the key notion of auxiliary problem. Informally speaking, a problem is auxiliary to another one if it is less constrained.

A semantic problem

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

is an auxiliary problem to

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

if

$$\Lambda' \subset \Lambda.$$

The following theorem provides a basis for computing an heuristic evaluation function (h(n)), as used by A* to focus its search [Nilsson. 3 980], from the information contained in the semantic representation of a computable problem (In this paper 1 follow the convention used in [Nilsson, 1980] in indicating the heuristic estimate with h(n) and its exact value with h*(n). Note that h*(i) is the cost of the optimal solution of P.)

Theorem. If P' is auxiliary to P, where the initial state for P' and P is n, then the length of the optimal solution of P' is a possible value for an admissible heuristic estimate h(n) for the problem P.

The proof is in [Guida and Somalvico, 1979].

Algorithm M

This algorithm is a special case of both algorithm G given in [Guida and Somalvico, 1979] and algorithm S in [Valtorta, I960].

Input: a semantic problem $P = (A, V, \Pi, \Lambda, i, k)$.

Output: an optimal solution of *P.*

Method: Solve the problem corresponding to the skeleton of *P* using A*, where each necessary value of h(n) is computed by solving an auxiliary problem to *Q,* using the Dijkstra algorithm. *Q* differs from *P* only in the initial state, which for *Q* is n.

All the auxiliary problems have the same set of legal conditions—A'. This ensures that the "consistency" condition [Niisson, 1971; Nilsson, 1980] is satisfied for h(n) computed by solving the auxiliary problems to *Q* This result is shown to hold in [Valtorta, l960].

Complexity of Algorithm M

In this section, 1 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 [Nilsson, 1971; Martelli, 1977]. First recall some results from [Martelli, 1977].

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.

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

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

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

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.) The following result can be proven:

Main Theorem. Let a semantic problem

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

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

Since the Dijkstra algorithm never expands the same node twice, it follows as a corollary that algorithm M uses at least the same number of node expansions as the Dijkstra algorithm.

Proof of the theorem.

Algorithm M expands nodes in two phases:
(a) to compute h(n);
(b) to solve *P,* with the A* strategy.

I shall consider three cases, which exhaust all possibilities, 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.

Case 1

The estimate h(n) allows node *n* to be expanded in phase (b).

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 not useful 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

(4) $g(n) + h(n)*h^*(i)$.

(Note that if (4) 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, offsets at best the saving arising from not expanding *n* in phase (b).

Case 3

There are nodes *r* that are expanded by the Dijkstra algorithm, but are not expanded by the M algorithm in phase (b) because, in order to be expanded, they should be reached through a node *m* whose estimate h(m) is so large that *m* is not expanded in phase (b).

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

(5) $d(m,r) < h^*(i) - g(m)$,

because d(m,r), the distance from *m* to r, plus g(m), equals g(r) which is bounded by h*(i) by Fact 1.

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

(6) $h(m) > h^*(i) - g(m)$.

Since h(m) is computed, in phase (a), by solving an auxiliary problem to P using the Dijkstra algorithm, one must expand, according to Fact 1, all the nodes at distance less than h(rn) 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 Pmust be expanded.

Therefore, the nodes (call them s) expanded by the M algorithm in phase (b) satisfy the following inequality:

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

By comparing (7) with (5), one concludes that, even in the most favorable case, the set of nodes r which are not expanded in phase (b) because of the computation of the estimate h(m) in phase (a) is a subset of the set of the nodes (nodes s) 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.

Conclusion

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

Given that a shortest-path algorithm is *blind* if it does not use heuristic information, and it is *unidirectional* if it expands nodes at increasing distances from the initial 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.

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:

(B) $g(n) < h^*(i)$

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 [Amarel, 1968], which is considered to be beyond the state of the art. (See, for example [Lenat, 1982, pp. 237-241].) Even auxiliary problems whose solutions lead 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 [Valtorta, 1980], 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 [Nilsson, 1971; Nilsson, 1980].

Appendix I: Related Research

Work on the semantic representation, motivated by the effort to automate the computation of heuristics, was started at the Politecnico di Milano by Marco Somalvico and his assistants in the mid-seventies.

Judea Pearl and the late John Gashnig have discovered, independently from 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" [Gashnig, 1979]. Gashnig uses the syntactic formalism and he does not propose an algorithm that finds auxiliary problems automatically, using the "semantic" formalism, as algorithm M does.

Judea Pearl and Dennis Kibler [Kibler, 1982] have postulated the need for changing representation paradigm to solve auxiliary problems efficiently. Their postulation is grounded on the negative result discussed in this paper of which i had informed them in personal correspondence. They quote this result explicitly in their reports [Pearl, 1982, p.131; Kibler 1982, p.4J.

References

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

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

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

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

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

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

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

[Nilsson, 1971] Nilsson, Nils J. *Problem Solving Methods in Artificial Intelligence*. New York: McGraw-Hill, 1971.

[Nilsson, 1980] Nilsson, Nils J. *Principles of Artificial Intelligence*. Palo Alto (Calif): Tioga Publishing Company, 1980

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

[Valtorta, 1980] Valtorta, Marco. "Un Contribute alia 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).