Acta Universitatis Apulensis No. 30/2012
ISSN: 1582-5329 pp. 147-152

SEARCHING AND KNOWLEDGE REPRESENTATION

Angel Garrido

ABSTRACT. The procedures of searching of solutions of problems, in Artificial Intelligence can be applied in some situations without knowledge of the Domain, and in other situations, with knowledge of it. This last procedure is usually called Heuristic Search. In such methods, for instance, the matricial and graph theory techniques reveal essential. Its introduction can give us an easy and precise way searching the solution. A very efficient procedure must be the A^* algorithm, and some other derived from him. For this reason, it will be analyzed here.

(2010)Mathematics Subject Classification: 68T30; 94C15; 68T20; 68T27.

1. Introduction

The problems analyzed in AI may be classified according their level [1, 3, 5].

With regard to the *techniques,* we can distinguish between the dedicated to statements and the devoted to procedures. The former alludes to allow to describe the known aspects of the problem. Their usual name is *heuristical treatment.*

Right representation will be a crucial respect, because wrong election of tool can lead inexorably to project failure. So, it is a very active research area.

Some general representations may be:

- *Semantic Networks* (or Nets; SNs, by acronym); between them, Frames and Conceptual Graphs.

- *Production Rules.*

- *Logical representations.*

- *Description Logic.*

As we known, Artificial Intelligence agents deal with data (or equivalently expressed, by knowledge).

It is developed by:

- *Facts* (observe and believe knowledge).

- *Meaning* (define and relate knowledge).

and

- *Procedures* (how to knowledge).

2. Knowledge representation

Logic is a language with concrete Rules. And also that *Syntax* are the Rules for constructing legal sentences in such Logic. Whereas *Semantics* means how toread (or interpret) sentences in the Logic.

We can mentionate [3, 4] between the more useful logical constructs developed through the time:

- *Propositional Logic,* whose Syntax includes propositions, connectives, brackets, T (true) and F (false), etc.; with a semantic Boolean, which defines how connectives affect Truth. We uses Truth Tables to establish the truth of statements.

- *Predicate Logic,* which combines atoms. Recall that an atom contains no propositional connectives, and it have no structure. In Predicate Logic, each atom is a predicate.

- *Fist-Order Logic (FOL, by acronym).* It is more expressive than the precedent construct, Propositional Logic. The constants are objects, and predicates are properties and relations. Functions transform objects. And Qualifiers gives the qualification assigned to values of variables.

- *Higher-Order Logic (HOL, by acronym)* is more expressive than FOL. Functions and predicates are also objects.

- *Multi-Valued Logic (MVL, by acronym),* where more than two values are present.

- For instance, in three-valued *Fuzzy Logic* (of Lukasiewicz), we have three truth values, True, False, and Unknown, or Possible, with respective values of 1, 0, and 1/2. As a particular case, the infinite-valued logic, or *Fuzzy Logic,* introduced by Lofti A. Zadeh, which uses a membership function whose range may be the real unit interval, [0, 1].

- *Modal Logic,* where the modal operators, necessity ($\Box$) and possibility ($\Diamond$) defines two different modes for propositions.

- *Temporal Logic,* associated with the time factor. So, we can express by $\Box p$ that *always p;* and by $\Diamond p$ that *eventually p.*

And we omit here some other interesting logical tools, as Paraconsistent Logic, Modal Fuzzy Logic, Non-Monotonic Logic, Default Reasoning, and so on.

3.HEURISTIC SEARCH

Relative to the searching with knowledge of the domain [3, 6], in an initial phase, were general thinking that all the paths can be explored by the computer. But it

is too optimist. Such exploration can frequently be very difficult, because of the phenomenon of "in combinatory explosion" of the ramifications, when we expand. Their spatial and temporal complexity can advise us against their realisation. For this, we need to select, firstly, the more promising trajectories. In this way, we can not obtain the best solution (optima), but an efficient approach to her.

Now, we introduce one new mathematical tool, the *heuristic evaluation function,* denoted by *f*.

By such function, we assign a value to each node, n. So, $f(n)$ give us the estimation of the real distance (unknown), from the actual node, n, until the final node, m.

4. ABOUT THE A* ALGORITHM

Now, we introduce one new and very interesting mathematical tool, the so-called *heuristic evaluation function,* denoted by *f*. By such function, we assign a value to each node, n. So, $f(n)$ give us the estimation of the real distance (unknown), from the actual node, n, until the final node, m.

There are *critics on the Heuristic Search*, because their *unpredictability*. They *found good solutions, but not necessary the best.* This made convenient the introduction of the *algorithm* A^*, with their useful properties of *completeness* and *admissibility*.

Best-First Search (BFS, by acronym) is a search algorithm which explores a graph by expanding only the most promising nodes, being these chosen according to a specific rule [3, 6, 8]. The A^* search algorithm is an extension of BFS. They are commonly used for path finding in combinatorial search.

A is a good extension of Dijkstra´s algorithm. This precedent was created in 1959.

A algorithm uses a BFS (best-first search)* procedure, so *finding the least-cost path* from the root-node to the goal node. It is widely used in pathfinding and graph traversal [2, 5].

Its supported technique is to use a distance-plus-cost *heuristic function, f:*

$$f = g + h$$

i.e.

$$f\,(n) = g(n) + h(n), \text{ for all node } n$$

So, it may be expressed as a sum of two auxiliar functions, g and h, where g measures the real cost from the root-node to the current node (hence, a known quantity), and h will be an admissible heuristic estimate of the distance to the goal node (hence, a unknown quantity, only estimated, using the heuristic, or expert knowledge), i.e. an estimation to such optimum final path.

About the *properties of the A* algorithm* [1, 2, 5], we can observe that

1) If there exists a path solution, it find it. For this reason, the search method will be so-called *Complete.*

2) If h is a minorant of h*, i.e. if $h \leq h^*$, which is equivalent to

$$h(n) \leq h^*(n)$$

for each node n, then A^* will found the optimal solution path. This may be expressable as the *Condition of Admissibility,* satisfied for A^*. Here, h^* will represent the real cost of the best path from the current node to the goal node (one of them, if there exists more of one).

3) We will say that an heuristic function, h_1, is best informed than another heuristic function, h_2, if both following conditions holds:

(i) $h_1 \leq h^*$, and $h_2 \leq h^*$; i.e., both are minorants functions of h,

and

(ii) $h_1 \geq h_2$

Hence, if h is *admissible,* then A^* will always find a least cost path to the goal node.

A^* is complete as long as:

1) Branching factor is always finite; every operator adds cost at least $\varepsilon > 0$;

and

2) temporal and spatial complexity will be in the worst case of the order $O(b^m)$.

By a good heuristic, we can find optimal problems in a very reasonable time.

How would be the behavior of the heuristic evaluation functions with A^*, according to they are best or worst informed?

For instance, let h_1 one heuristic evaluation function more informed than h_2, another heuristic evaluation function. Then, for each node n, generated by A^*, with the heuristic h_1, it is also generated by such algorithm with the heuristic h_2.

It is not necessarily true in converse sense. Not in general, unless the coincidence of both heuristic evaluation functions.

We says then that the algorithm A^* is more efficient with the heuristic evaluation function h_1 (the best informed) than with the heuristic evaluation function h_2 (the worst informed of both).

We can resume saying that:

to more information, more efficiency

and reciprocally,

to lesser information, lesser efficiency

About the *Monotonicity Property* of both the heuristic evaluation function, *h,* and the algorithm A^*, we need to define that

h will be *monotonic* if

$$h(n) \leq c(n,\ n') + \ h(n,\ n')$$

for every pair of nodes n and n', being n' a successor node of n, with $c(n,\ n')$ the cost function value from n to n'.

If the heuristic evaluation function, *h, is monotonic,* then the algorithm A^* find the optimal path solution, for every expanded nodes. So, it would be unnecessary to revise each time if it is compulsory to be reoriented their links.

References

[1] Fernández Galán et al.: *Problemas resueltos de Inteligencia Artificial Aplicada. Búsqueda y Representación.* Addison-Wesley Iberoamericana. Madrid, 1998.

[2] Garrido, The 8-puzzle problem. *AUA (Acta Universitatis Apulensis),* No. 12/2006, pp. 25-32.

[3] Ginsberg: *Essentials of artificial intelligence.* Morgan Kaufmann Publ. San Francisco, Cal., 1993.

[4] Klir and Yuan: *Fuzzy Sets and Fuzzy Logic. Theory and Applications.* Prentice-Hall Inc., 1995.

[5] Mira et al.: *Aspectos Básicos de la Inteligencia Artificial.* Ed. Sanz y Torres. Madrid, 1995.

[6] Nilsson: *Principles of Artificial Intelligence.* Tioga. Palo Alto, California, 1980.

[7] Pazos: *Inteligencia Artificial*. Edit. Paraninfo. Madrid, 1987.

[8] Winston: *Artificial Intelligence*. Addison-Wesley Publ. Co., 1994. Third Edition.

Angel Garrido
Faculty of Sciences
University UNED
Senda del Rey 9, 28040-Madrid
email: *agarrido@mat.uned.es*