SOME REMARKS ON HEURISTIC SEARCH ALGORITHMS

Lasz16 Mero

Computer and Automation Institute, Hungarian Academy of Sciences

In this communication the terminology and the notations of the paper of Martelli [1] are used throughout. A similar treatment is given in the book of Nilsson [2]. Familiarity with at least one of these works is assumed to read this paper.

1. An improved version of Martelli's algorithm B

Martelli in his algorithm B improved the classical algorithm A for finding a minimal cost path in a graph using heuristic estimates by introducing a global variable F which gives the greatest r values achieved up to that step by a closed node. F is used to control whether the remaining costs promised by the heuristic estimates are in accordance with the propagation of the algorithm so far. Algorithm B uses the heuristic estimates only if they promise to improve the global F value, otherwise it uses only the g-estimates. This way algorithm B avoids the possible exponential running time of the original algorithm A even if the consistency assumption does not hold. Nevertheless, some particularly wrong heuristic estimates can mislead the search of algorithm B, as well. Fig. 1. displays an example of Martelli for which algorithm A requires 0/2*[1]/ node expansion and algorithm B requires only 0/N*7 expansions.

The improved algorithm, let us call it algorithm B', is based on the observation that expanding a node may result in a possibility to improve the heuristic estimate of that node, and perhaps also of its son nodes. E.g. in the graph of Fig.1. after expanding node n, the heuristic estimate of n, could have been set 19 instead of zero still maintaining the upper bound property of the heuristic extimate, but this change would have avoided several reopenings of node n. performed both by algorithm A* and B. On the other hand, if e.g. the heuristic estimate of ru /also in the graph of Fig.1./ had been 23 instead of zero, after expanding n. the heuristic estimate of n, could have been set 12. This operation would have saved several reopenings, as well.

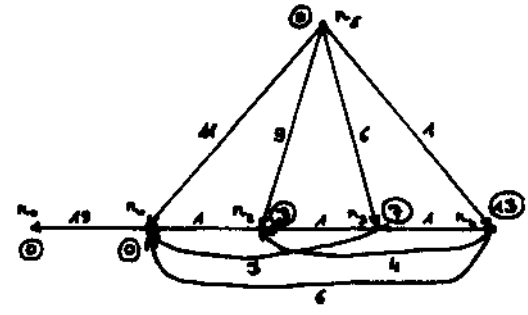

Fig.1. A graph for which algorithm A* requires O/2^N/, B O/N^2/ and B' O/N/ node expansions

Algorithm B' is a variant of B and it can be obtained from B by inserting two steps after step /3/ of algorithm B. Thus algorithm B' looks the following way:

/1/ Put the start node s on a list called OPEN. Set

$$\hat{g}(s) \leftarrow 0,\ \hat{f}(s) = \hat{h}(s),\ F \leftarrow 0$$

/2/ If OPEN is empty exit with failure; otherwise continue.

/3/ If there are some nodes in OPEN with $\hat{f} < F$, select among them the node n whose $\hat{g}$ value is smallest; otherwise, select the node n in OPEN whose $\hat{f}$ value is smallest and set $F \leftarrow \hat{f}(n)$. /Resolve ties arbitrarily, but always in favour of any goal node./ Remove n from OPEN and put it on a list called CLOSED.

/3a/ For each son m of the recently selected node n if

$$\hat{h}(m) < \hat{h}(n) - c(n,m)$$

holds, then set

$$\hat{h}(m) \leftarrow \hat{h}(n) - c(n,m)$$

/3b/ Let m be a son of n for which $\hat{h}(m)$ +

+ $c(\underline{m},\underline{n}) \stackrel{def}{=} \hat{h}_{\underline{m}}(\underline{n})$ is minimal. If $\hat{h}_{\underline{m}}(\underline{n}) \neq \hat{h}(\underline{n})$ then set $\hat{h}(\underline{n}) \leftarrow \hat{h}_{\underline{m}}(\underline{n})$

/4/ If $\underline{n}$ is a goal node, exit with the solution path obtained by tracing back through the pointers; otherwise continue.

/5/ Expand node $\underline{n}$, generating all of its successors. /If there are no successors go to /2/. For each successor $\underline{n}_i$, compute $\hat{g}_i \leftarrow \hat{g}(\underline{n}) + c(\underline{n}, \underline{n}_i)$

/6/ If a successor n_i is not already on either OPEN or CLOSED, set $\hat{g}(\underline{n}_i) \leftarrow \hat{g}_i$ and $\hat{f}(\underline{n}_i) \leftarrow \hat{g}_i + \hat{h}(\underline{n}_i)$ Put $\underline{n}_i$ on OPEN and direct a pointer from it back to $\underline{n}$.

/7/ If a successor $\underline{n}_i$ is already on OPEN or CLOSED and if $\hat{g}(\underline{n}_i) > \hat{g}_i$, then update it by setting $\hat{g}(\underline{n}_i) \leftarrow \hat{g}_i$ and $\hat{f}(\underline{n}_i) \leftarrow \hat{g}_i + \hat{h}(\underline{n}_i)$ Put n_i on OPEN if it was on CLOSED and redirect to $\underline{n}$ the pointer from $\underline{n}_i$.

/8/ Go to /2/.

For this modified algorithm the following statement holds:

Theorem 1. Algorithm B' is admissible.■

This theorem can be proved in a similar way as the admissibility of algorithms A^* and B is proved in [1] and [2].

2. The complexity of algorithm B'

The complexity of a heuristic search algorithm is measured by the number of node expansions and not by the number of elementary operations required to perform the algorithm. This approach /also taken in [1] and [2]/ is motivated by the fact in most of the applications that the most time consuming operation is generating the successors of a node /i.e. the node expansion/ and calculating their heuristic estimates, all other administrative activities of the search algorithms are of negligible complexity.

Theorem 2. For every graph algorithm B' expands each node at most as many times as algorithm B, if both algorithms resolve ties in the same way.-

This theorem looks very plausible because in all cases of inconsistencies algorithm B improves its heuristic estimates, and in no other cases. However, better heuristic estimates alone do not warrant better performance. If e.g. the heuristic estimates of the graph in Fig.1. were all zero, algorithm B would expand each node only once. On the other hand, algorithm B' may change the order of expanded nodes drastically compared with algorithm B. In spite of the plausibility of Theorem 2., the author could find only a fairly long and sophisticated proof of it.

Theorem 4.1. in [1] can be sharpened the following way, using the same proof technique:

Theorem 3. Algorithm B requires at most $N^2/2 + O/N/$ node expansions. Fig.1. shows that this statements is sharp.■

The author conjectures that algorithm B' requires only $N^2/4 + O/N/$ steps in the worst case, but he was able to prove only the following statement:

Theorem 4. Algorithm B' requires at most $N^2/8 + O/N/$node expansions.■

The proof of this theorem is quite straightforward, however fairly lengthy. The main guideline is to show that if a node is expanded k times, then there must be at least k distinct nodes "before" and also "after" this node.

Fig. 2. displays a graph for which algorithm B' requires $N^2/8 + O/N/$ node expansions. /It is easy to see, how to generalize this graph to have N nodes./

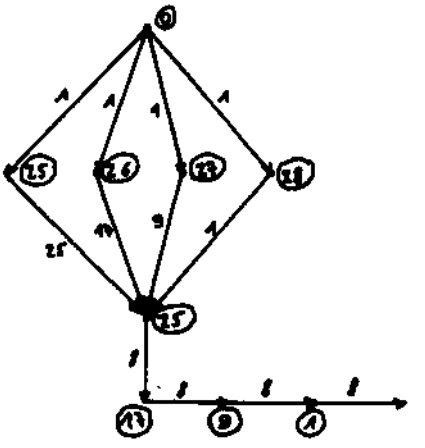

Fig. 2.

Let us remark, that if steps /3a/ and /3b/ are inserted to the original algorithm A^* instead of algorithm B, Theorem 4. will hold for this algorithm as well, thus this method turns out to be another way to avoid the possible exponential running time of algorithm A^*.

3. No overall optimal algorithm exists

If the consistency assumption holds, it can be proved that algorithm A^* /and also algorithms B and B'/ expands the possible least number of nodes, i.e. no algorithm not better informed than A^* can expand less number of nodes.

If the complexity of a search algorithm is measured by the number of distinct expanded nodes, this optimality property of A* will still hold. But in our case the complexity of a search algorithm is measured by the total number of node expansions, and in this case the following theorem can be proved:

Theorem 5. Let M be an algorithm for finding the minimal""cost path in graphs using the heuristic information attached to the nodes of the graphs. There exists an algorithms N and a graph G so, that algorithm N also finds the minimal cost path in all graphs and N requires fewer node expansions than M when applied to G.g

The proof of this theorem consists of a set of consecutive counterexamples resulting in a graph for which it is proved that at least one of its nodes must be expanded at least twice by an optimal algorithm. Then an algorithm is constructed which also finds always the minimal cost path but expands each node of this particular graph only once.

4. Conclusions

An improved version of Martelli's algorithm B has been described. This algorithm can improve its heuristic estimates during its run. The improved algorithm never requires more node expansions than the original algorithm, and its worst case behaviour is also at least twice better. Furthermore, there are cases, when the original algorithm requires O/N / node expansions for an N-node graph, while the improved version requires only 0/N/ expansions.

Another result is that if the complexity of the algorithm is measured by the total number of node expansions, no overall optimal search algorithm exists.

The proofs of the theorems stated in this communication will be given in a forthcoming paper, now in preparation.

References

1. A.Martelli: On the complexity of admissible search algorithms. Artificial Intelligence 8 *11911/,* 1-13
2. N.J.Nilssonx Principles of Artificial Intelligence. TIOGA Publishing Company, Palo Alto, California, 1980