In orbit assembly of large structures

Artificial Intelligence and Decision Systems - Assignment #1

Authors: Luís Rei (78486), João Girão (78761)

November 3, 2017

...

1 Problem representation

Problem solving is defined as a combinatorial search problem, where an agent must find the right sequence of actions to achieve a certain goal. In this particular assignment, we are formulating a single-state search problem in a fully observable and deterministic world, where each action generates a single state transition. To formulate a search problem, a description of the level of detail, both in describing states and actions, is necessary. The assignment is detailed with

- states: orbit and ground set representing the components flown to the ISS or not, as well as the date of the last launch that allows that configuration.
- initial state: the initial configuration of our space mission has the orbit set empty, as opposed to the ground set that is filled with all the ISS components.
- operators: launches that carry a set of structures to orbit.
- goal state: all initially grounded ISS components are airborne.
- path cost: the cumulative cost of launches leading to a certain node.

2 Search algorithm design

To find the right approach to the problem one must take into account its scope. The search algorithm works on a limited state-space and we want to optimize that search to achieve the best result based on our performance metric. The problem we are facing is decomposed as a directed cycle graph that, unlike a tree-based algorithm, handles repeated nodes. This brings additional computations in the form of a 'closed' node list. Taking in mind that the state-space can be very big, building and looking at each iteration into a 'closed' list containing the nodes already expanded in the search can be extremely time consuming.

Besides time and memory, solution optimality was also a concern. Not only did we want to guarantee that a solution was found (completeness), we wanted it to minimize our cost function, we wanted the lowest path cost among all solutions.

2.1 Blind approach

The uniform-cost search fulfills our demands and was used to resolve our uninformed search problem. This blind search method does not use additional (domain-dependent) information about states beyond that provided in the problem definition. Time (how long does it take to find a solution) and space (how much memory is needed to find a solution) complexities also made a strong case in favor of this methodology as it compares nicely against other strategies mentioned in some depth at the theoretical classes.

Guided by path costs rather than depths, these complexities are not easily characterized in terms of b (branching factor) and node depths. Instead, letting C^* be the cost of the optimal solution, and assume that every action costs at least ϵ, then the algorithm's worst-case time and space complexity is

$$O(b^{1+\frac{C^*}{\epsilon}}),$$

which can be much greater than b·d, d being the depth of the shallowest goal node[1].

The algorithm properties are summed up as:

1. Complete, if step costs are strictly positive;
2. Optimal, if step costs are strictly positive, because $\forall_n \quad g(successor(n) > g(n)$, with g(n) the cost of going from the root node to node n (path cost function);
3. Complexity proportional to the number of nodes with path cost less than of the optimal solution.

2.2 Informed approach

The heuristic search uses problem-specific knowledge about the domain to "guide" the search towards more promising paths. This additional knowledge is used in the form of an heuristic function - $h(n)$ - that estimates the cost between the current node and the goal node.

In the A^* algorithm we combine this information with function g(n) to minimize an evaluation function f(n) = h(n) + g(n). If h(n) is consistent - $h(n) \leq c(n, a, n') + h(n')$ -, and no repeated nodes are discarded, the solution returned by A^* in a graph-search accomplishes our propositions:

1. Complete, as long as step costs are strictly positive;
2. Optimal, because $f(n) = g(n) + 0 > C^* \geq g(n^*) + h(n^*) = f(n^*) \implies f(n) > f(n^*)$

Fundamentally, the only requirement for admissibility is that a heuristic never over-estimates the distance to the goal. This is important, because an overestimate in the wrong place could artificially make the best path look worse than another path, and prevent it from ever being explored. Thus a heuristic that can provide overestimates loses any guarantee of optimality. Underestimating does not carry the same costs. If you underestimate the cost of going in a certain direction, eventually the edge weights will add up to be greater than the cost of going in a different direction, so you'll explore that direction too. The only problem is loss of efficiency.

In general, time and space complexity of A^* is exponential with the solution depth of $O(b^d)$ since it expands all nodes with f(n) < C^*, and some (or all) nodes with f(n) = C^*, and the number of nodes with f(n) = C^* is exponential with the solution depth (d).

The strategy introduced as a parameter in a general search algorithm will be a priority queue on g or f-values (according to the use of the uniformed or informed search method respectively).

3 Heuristic

Heuristic functions that produce negative values are not inadmissible, per se, but have the potential to break the guarantees of A^*, so we restricted ourselves with a positive value-producing heuristic function: $h(n) = c_k \sum_{i=1}^{n} w_i$, with c_k the cost of the cheapest launch in the possible launch set, and w_i the weights of the components yet on the ground. The performance of the heuristic function is summed in table 1.

Table 1: Qualitative analysis on the heuristic impact in our search problem

Files used	trivial1.txt		trivial2.txt		simple1.txt		mir.txt	
Metric	Nodes	b	Nodes	b	Nodes	b	Nodes	b
Blind	25	37.474	11	19.170	3455	4297.151	9951	16537.556
Heuristic	15	24.548	11	19.170	1317	1727.421	21931	34702.523

Guaranteed the desired A* sought after properties, we deduced that the used heuristic proved to be very useful on cases where the cheapest launch was used, and gave extremely poor results when that condition wasn't met. Should the project be further improved, our two main focus areas would be providing a more robust heuristic and optimizing the search algorithm developed.

References

[1] Russell, Stuart and Norvig, Peter, *Artificial Intelligence: A Modern Approach*, Pearson Educated Limited (3rd Edition), 2009.