Final Notes for Intro to Artificial Intelligence Final Exam

Andrés Ponce

June 19, 2020

Topics covered in this final exam include search algorithms, constraint satisfaction, and logical satisfiability.

Search Algorithms

Breadth-First Search

With this algorithm, remember we have to search for the *shallowest unexpanded node*. This means that we will check the tree/graph layer by layer before moving on to the next yet unexpanded node. The data structure used for this algorithm will probably just be a FIFO structure, since we first want to examine the nodes that are shallower than the current one, so we should pop the node that has been the longest in the structure.

1. **Completeness**: Yes, on finite graphs.
2. **Time Complexity**: $O(b^d)$
3. **Space Complexity**: $O(bd)$
4. **Optimality**: Yes, for unit costs. Why? If there is a unit cost, then the first time we encounter the node, its value will be guaranteed to have the lowest value. If it did not have the lowest value, we would have encountered it previously, right?

Depth-First Search

Depth-First Search works quite differently than Breadth-First search. With DFS, we expand the deepest, unexpanded node. What this means is that we traverse the tree/graph in a single path, following the children of the current child until we can do so no more. Once this happens, we then backtrack up to the deepest unexpanded node, possibly along the same general path that we were travelling before.

1. **Completeness**: Nope. Might repeat given loops in the state space.
2. **Time Complexity**: $O(b^m)$, because we could potentially reach the maximum state space distance before finding the target node.
3. **Space Complexity**: $O(bm)$, since we have to sotre all nodes in memory

4. **Optimality**:Nope, it may search other paths, which might cause it to terminate before actually reaching the best solution.

It is complete for finite state spaces, probably becuase it searches all the nodes eventually, thus finding a solution if there is one.

Depth-Limited Search

This algorithm is essentially DFS but having a maximum depth be l. What does this avoid? The so-called loops which may result are avoided, however we would need to be careful to not limit ourselves to not finding the wrong answer if we set a small l.

Iterative Deepening Search

Here, we consistently run depth-limited search on several of the nodes. We increase the maximum depth l every time we run the algorithm.

1. **Completeness**: Yes, for finite d.
2. **Time complexity**: $O(b^d)$ [1]
3. **Space Complexity**: $O(bd)$
4. **Optimality**: Yes.

[1] Remember d is the length of the maximum length of the solution.

Uniform-Cost Search

With this algorithm, we expand the least-cost unexpanded node. This means that as we get closer to the target node, we might also see a reduction in the value of the node.

1. **Completeness**: Yes, if cost $\geq \epsilon$
2. **Time Complexity**:
3. **Space Complexity**
4. **Optimality**: Yes

Informed Search Strategies

If we remember, *un*-informed search strategies would use the With informed search, we can use more information at each stage other than just the parent and children, for example, we can have an idea of how much we still have to go until the target node, for example.

With this new ability, we have a couple more tools to use:

1. **Best-First Search**: Now, we can expand the node that looks to lead us to the best solution (e.g. the node whose heuristic value is the lowest).
2. **Evaluation Function**: We now can have a function $f(n)$ which tells us which node to choose for expansion. (Useful in A*, for example).
3. **Heuristic Function**: We can now have an educated guess of how much we have to reach the

For example, the first algorithm we can study is a simple **Greedy Best First** algorithm. Using this algorithm, we merely select the node that has the best heuristic value out of the bunch. However, this method is neither complete nor optimal, why? [2]

[2] I guess there might be a scenario where at one stage n, we choose the one with the lowest heuristic value, but at a lager stage $n + j$ it turns out the combined sum was lower for a second path.

A Search*

Here, the algorithm relies on one formula

$$f(n) = g(n) + h(n) \tag{1}$$

The $f(n)$ value, as mentioned, represents the total combined value for the node. This value is the sum of $g(n)$, i.e. the cost of moving from the starting node to the current node, and $h(n)$, which is the cost of going from the current node's children to the target node.

1. **Completeness**: Yes, unless there are infinitely many nodes with cost $f(n) < C$, where C is the optimal cost to goal state.
2. **Time Complexity**: Exponential
3. **Space Complexity**: Keeps all the nodes in memory, so not pretty efficient
4. **Optimality**: Yes, given some conditions on the heuristic:

For A* to really live up to its potential, we need an **admissible** heuristic. An admissible heuristic is one which does not overestimate the true cost of the node to the target. [3]

[3] Ohterwise, if it overestimated sometimes, we could choose a less efficient route by accident?

Another condition for A* optimality is a **consistent** heuristic. A consistent heuristic is one which satisfies the **triangle inequality**:

$$h(n) \leq c(n, a, n') + h(n)' \tag{2}$$

In this equation, $c(n, a, n')$ is the cost of going from the current node n to the next one in the path n' by using action a. [4]

[4] Consistent Heuristics are always admissible, not so the other way around.