CS188 Fall 2018 Section 2: Graph Search

1 Maze Navigation

Imagine a car-like agent wishes to exit a maze like the one shown below:

The agent is directional and at all times faces some direction \( d \in \{N,S,E,W\} \). With a single action, the agent can either move forward at an adjustable velocity \( v \) or turn. The turning actions are left and right, which change the agent’s direction by 90 degrees. Turning is only permitted when the velocity is zero (and leaves it at zero). The moving actions are fast and slow. Fast increments the velocity by 1 and slow decrements the velocity by 1; in both cases the agent then moves a number of squares equal to its NEW adjusted velocity. Any action that would result in a collision with a wall crashes the agent and is illegal. Any action that would reduce \( v \) below 0 or above a maximum speed \( V_{\text{max}} \) is also illegal. The agent’s goal is to find a plan which parks it (stationary) on the exit square using as few actions (time steps) as possible.

As an example: if the agent shown were initially stationary, it might first turn to the east using (right), then move one square east using fast, then two more squares east using fast again. The agent will of course have to slow to turn.

1. If the grid is \( M \) by \( N \), what is the size of the state space? Justify your answer. You should assume that all configurations are reachable from the start state.

   The size of the state space is \( 4 MN(V_{\text{max}} + 1) \). The state representation is (direction facing, \( x, y \), speed). Note that the speed can take any value in \( \{0, ..., V_{\text{max}}\} \).

2. Is the Manhattan distance from the agent’s location to the exit’s location admissible? Why or why not?

   No, Manhattan distance is not an admissible heuristic. The agent can move at an average speed of greater than 1 (by first speeding up to \( V_{\text{max}} \) and then slowing down to 0 as it reaches the goal), and so can reach the goal in less time steps than there are squares between it and the goal. A specific example: the target is 6 squares away, and the agent’s velocity is already 4. By taking only 4 slow actions, it reaches the goal with a velocity of 0.

3. State and justify a non-trivial admissible heuristic for this problem which is not the Manhattan distance to the exit.

   There are many answers to this question. Here are a few, in order of weakest to strongest:

   (a) The number of turns required for the agent to face the goal.

   (b) Consider a relaxation of the problem where there are no walls, the agent can turn and change speed arbitrarily. In this relaxed problem, the agent would move with \( V_{\text{max}} \), and then suddenly stop at the goal, thus taking \( d_{\text{manhattan}}/V_{\text{max}} \) time.

   (c) We can improve the above relaxation by accounting for the deceleration dynamics. In this case the agent will have to slow down to 0 when it is about to reach the goal. Note that this heuristic will always return a greater value than the previous one, but is still not an overestimate of the true cost to reach the goal. We can say that this heuristic dominates the previous one.
2 Admissible and Consistent Heuristics

In this question we’ll explore the relationship between admissible and consistent heuristics.

Recall the definitions for admissible and consistent heuristics. Let \( s' \) denote the successor of \( s \), and \( \text{FutureCost}(s) \) denote the minimum cost from \( s \) to the goal.

- **Admissible**: \( h(s) \leq \text{FutureCost}(s) \).
- **Consistent**: \( h(s) - h(s') \leq \text{Cost}(s, s') \) and \( h(\text{Goal}) = 0 \).

1. If we used an inadmissible heuristic in A* graph search, would the search be complete? Would it be optimal?

   If the heuristic function is bounded, then A* graph search would visit all the nodes eventually, and would find a path to the goal state if there exists one. An inadmissible heuristic does not guarantee optimality as it can make the good optimal goal look as though it is very far off, and take you to a suboptimal goal.

2. If we used an admissible heuristic in A* graph search, is it guaranteed to return an optimal solution?

   No. Admissible heuristics guarantee optimality only for tree-search problems but not general graph search problems. A heuristic must be consistent to be optimal for graphs, which we’ll prove later.

3. Give a general advantage that an inadmissible heuristic might have over an admissible one.

   The time to solve an A* search problem is a function of two factors: the number of nodes expanded, and the time spent per node. An inadmissible heuristic may be faster to compute, leading to a solution that is obtained faster due to less time spent per node. It can also be a closer estimate to the actual cost function (even though at times it will overestimate!), thus expanding less nodes. Sometimes we may be okay with finding a suboptimal solution to a search problem at a cheaper cost.

4. Show that A* search always find the optimal solution when using a consistent heuristic. Let’s look at the cumulative cost of a length-T path from state \( s_1 \) to state \( s_{T-1} \). Let Cost denote the original edge cost, and \( \text{Cost}' \) denote the modified A* edge cost (A* is equivalent to running UCS on a graph with modified edge costs \( \text{Cost}'(s, s') = h(s') - h(s) + \text{Cost}(s, s') \)).

   \[
   \sum_{i=1}^{T-1} \text{Cost}'(s_i, s_{i+1}) = \sum_{i=1}^{T-1} \text{Cost}(s_i, s_{i+1}) + h(s_{i+1}) - h(s_i) = \left( \sum_{i=1}^{T-1} \text{Cost}(s_i, s_{i+1}) \right) + (h(s_{T-1}) - h(s_1))
   \]

   This sort of cancellation is often called a telescoping sum. Notice that term \( h(s_{T-1}) - h(s_1) \) is constant for all paths between \( s_1 \) and \( s_{T-1} \) - a consistent heuristic effectively leaves path costs unchanged (up to a constant). Since a constant offset can never change which path is optimal, A* using a consistent heuristic must find the optimal solution.

5. Consistency is a stronger condition than admissibility. Show that every consistent heuristic is admissible. *(Note that this is not true the other way around: there are admissible heuristics that are not consistent)*

   We can show this by induction on the states, starting from the goal. Our base case for the goal is true since one of the requirements for consistency is that \( h(\text{Goal}) = 0 \).

   As the inductive hypothesis, let’s assume that \( h(s') \leq \text{FutureCost}(s') \) for the best successor \( s' \) from \( s \). We need to show that \( h(s) \leq \text{FutureCost}(s) \). Then we have:

   \[
   h(s) \leq h(s') + \text{Cost}(s, s') \leq \text{FutureCost}(s') + \text{Cost}(s, s') = \text{FutureCost}(s)
   \]

   The first inequality comes from the definition of consistency. The second from our inductive hypothesis. The last equality comes from our assumption that \( s' \) was the optimal next state to visit from \( s \).