D1

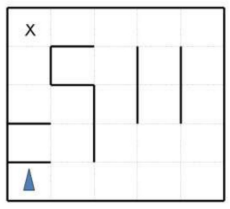
Thursday, September 27, 2018 9:28 AM

section_1_0hzy6TFupb1Z3bckfRXdC5KYpsdZOE

CS188 Spring 2014 Section 1: Search

1 Search and Heuristics

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 $V_{\max}$ 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.

   Counting Problem: 4MN*(Vmax+1)
   Location: M x N = MN
   Directions: |NSEW| = 4
   Velocity: |{0,1,2… Vmax}| = Vmax+1

2. What is the maximum branching factor of this problem? You may assume that illegal actions are simply not returned by the successor function. Briefly justify your answer.

   When stationary, the car can turn 90 degrees left / right OR accelerate, b = 3
   When moving, the car can accelerate or decellerate: b = 2

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

Manhattan distance is not admisslbe for large grids because they will UNDERESTIMATE
The cost.

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

Heuristic: Assume the car can make a straight line, with no turning, towards the goal.
Analysis: With constant acceleration, the car must spend the first half of the distance accelerating and the second half decellerating to a stop. We solve for steps to cover half distance accelerating, then double this cost.
A = 1
V = 1*steps+v0
X = 1/2*steps**2 + v0*steps + c0.
MD = manhattan distance
Halfcost: MD/2 = 0.5*steps**2
Half-cost: steps = sqrt(md)
Heuristic = 2*sqrt(MD)

5. If we used an inadmissible heuristic in A* tree search, could it change the completeness of the search?

No. Heuristics just let us prioritize which nodes to explore next, but eventually we will explore all nodes. We will eventually explore the tree down to the goal state.

6. If we used an inadmissible heuristic in A* tree search, could it change the optimality of the search?

In tree search, our optimality could be jeopardized. We may conclude we have our solution while the true optimal path APPEARS to not be promising and have an expected high future cost.

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

Easier to generate. Though we lose optimality. May provide FAST subotimal solution.

2 Expanded Nodes

Consider tree search (i.e. no closed set) on an arbitrary search problem with max branching factor b. Each search node n has a backward (cumulative) cost of $g(n)$, an admissible heuristic of $h(n)$, and a depth of $d(n)$. Let c be a minimum-cost goal node, and let s be a shallowest goal node.

For each of the following, you will give an expression that characterizes the set of nodes that are expanded before the search terminates. For instance, if we asked for the set of nodes with positive heuristic value, you could say $h(n) \geq 0$. Don't worry about ties (so you won't need to worry about $>$ versus $\geq$). If there are no nodes for which the expression is true, you must write "none."

1. Give an expression (i.e. an inequality in terms of the above quantities) for which nodes n will be expanded in a breadth-first tree search.

   - All nodes with depth < s will be expanded.
     - Graph must explore all nodes at a certain depth before continuing deeper. Therefore,
       - s-1, s-2, s-3…. To depth 1 are all explored before termination.

2. Give an expression for which nodes n will be expanded in a uniform cost search.

   - All nodes with g(n) < c will be expanded
     - UCS is guaranteed optimality by exploring all nodes by cost priority. Therefore, the first goal state it reaches MUST be the lowest cost goal state, because all other nodes of lower cost were explored and were VERIFIED to be not a goal state

3. Give an expression for which nodes n will be expanded in an A* tree search with heuristic $h(n)$.

   - All nodes will with h(n)+g(n) < optimal path cost are guaranteed to be expanded.
     - A* search will NOT terminate if they still exist in the graph.

4. Let h_1 and h_2 be two admissible heuristics such that $\forall n, h_1(n) \geq h_2(n)$. Give an expression for the nodes which will be expanded in an A* tree search using h_1 but not when using h_2.

   Notes, h1 provides lower bound on cost from node to goal (admissibility precondition guarantees this). Therefore, the "tighter" our lower bound is, the less time we will explore paths that don't lead to goal.

F1(n) = g(n)+h1(n)	F2(n)=g(n)+h2(n)

   This is an empty set. Using h2, we MUST expand all nodes expanded when using h1. From the termination condition w/ h1. all nodes with g(n)+h1(n) < s are explored. For h2, all nodes with g(n)+h2(n)<s are explored. Nodes g(n)+h2(n) < g(n)+h1(n), so using h2, we explore all nodes explored by h1 (and hten some)

5. Give an expression for the nodes which will be expanded in an A* tree search using h_2 but not when using h_1.

   Nodes expanded by h2, not by h1, satisfy the condition: g(n)+h2(n) < s. and g(n)+h1(n) > s
   The first conditional is all nodes expanded using h2. The second conditional is for nodes NOT expanded using h1