AI - HW 1

Daniel Yu

September 21, 2024

Contents

1 Problem 1

1. Maze-solving would be considered fully observable environment when the agent has complete knowledge of the layout or configuration of the environment, the agent's position, the positions of other objects or agents, etc. The type of algorithms best suited to solving problems in this setting would be **Search Algorithms** such as A^*, Dijistra and other shortest path algorithms.
2. Maze-solving would be considered partially observable environment when the agent has incomplete knowledge about the layout, configuration, position, other objects, etc. in the environment. For example, consider a robot attempting to navigate a maze using a camera to perceive the environment. Clearly, the robot would have to navigate with a restricted field of vision. The agents percepts would be the limit information it could perceive about its surroundings. In terms of implementation, I would have the agent build a model in its memory and update the model with new incoming information from the surroundings, implementing logic to have the agent navigate by exploring all possible paths, backtracking if stuck, using randomized directions, etc. as the agent has no idea which path is correct.
3. No, consider the game of solitaire where the rules of the game can be known to the agent so the environment is known. However, the agent doesn't have full information (doesn't know what the order of the cards left in the deck are for example) so the environment is only partially observable!
4. No, refer to the eariler example. Since solitaire's rules are known, the environment is known even though the agent has incomplete information and is operating in a partially observable environment.
5. An unknown environment is an environment in which the rules are not known. Consider for example, the classic reinforcement learning example, where there is only a goal that the agent is given, but no information about how to achieve the goal only penalties for bad behavior and rewards for good behavior. In this case, the agent has to learn the rules of the game such as selling equipment for money. On the other hand, a non-deterministic environment is an environment where outcomes are probabilistic. For example, consider that same game where now there is a certain chance to get robbed while selling goods. Previously, the game was unknown and deterministic, but with this addition it is unknown and non-deterministic.

2 Problem 2

1. If the "w == goal: terminate" condition is removed, the number of nodes explored would be increased by the number of neighbors of the original node v excluding w. For example, if the node A had two 2 neighbors: B,C and B was the goal and gets explored first. Instead, of BFS immediately terminating, it would continue to C. Then on the next iteration, B would be dequeued and it would terminate. However, if instead node A had two neighbors: C,B and C gets explored first. Then after B gets explored (and doesn't terminate), it would wrap back to C whereupon, it would continue exploring all of C's neighbors excluding B, and then finally, B would get dequeued.
2. The asympoptic notation would be dominated by the last layer i.e. the layer at the depth limit, as the higher layers would be absorbed into the complexity of the lower layer. So while, there will be $O(b^{d-1})$ redudanant calculations from the previous depths, the number of nodes expands at each iteration would be $O(b^d)$ at the bottom layer where b is the branching factor and d is the depth dominating the prior terms.
3. Yes, because essentially, this will just become a graph where each node is $A_1, \ldots, A_5, \ldots, E_5$ and the neighbors of the nodes would just be nodes that are x distance away, so for example, D_1 would just have neighbors D_4, A_1 which would then in turn have their own neighbors x distance away. So this becomes a directed graph problem with the same cost at each node which then either BFS, DFS can solve for the shortest path!
4. We will prove this:

   Proof. To prove that $H(s, g, n) = \frac{|s_x - g_x| + |s_y - g_y|}{n-1}$ where s is an arbitrary start node, g is any goal node, and n is the size of the square grid, is consistent, we have to show that $H(s, g, n) \leq c(s \to s') + H(s', g, n)$. We know that the cost from s to any neighbor s' is constant cost of 1. So we have to show

   $$H(s, g, n) \leq c(s \to s') + H(s', g, n) \to H(s, g, n) \leq 1 + H(s', g, n).$$

.

$$\frac{\mid s_x - g_x \mid + \mid s_y - g_y \mid}{n-1} \leq 1 + \frac{\mid s'_x - g_x \mid + \mid s'_y - g_y \mid}{n-1}$$
$$\frac{\mid s_x - g_x \mid + \mid s_y - g_y \mid}{n-1} \leq \frac{n-1+ \mid s'_x - g_x \mid + \mid s'_y - g_y \mid}{n-1}$$

Since we know that any move that s makes to s' will be only in one direction, i.e. x or y direction, without loss of generality, let that move be in the direction of y-axis. Then, $s_x = s'_x$ and the terms cancel.

$$\frac{\mid s_y - g_y \mid}{n-1} \leq \frac{n-1+ \mid s'_y - g_y \mid}{n-1}.$$

The above inequality must hold because the maximum value of $\mid s_y - g_y \mid = n-1$ and $\mid s'_y - g_y \mid \geq 0$, so

$$\arg \max_{|s_y - g_y|} [\frac{\mid s_y - g_y \mid}{n-1} \leq \frac{n-1+ \mid s'_y - g_y \mid}{n-1}] = \frac{n-1}{n-1} \leq \frac{n-1+ \mid s'_y - g_y \mid}{n-1}.$$

□

5. A^* algorithm:
   - Step 2:
     (a) Pop D_2 since the priority are the same and it is first in queue.
     (b) Since it is not the goal state continue:
     (c) Neighbors: $\{D_1, C_2, D_3\}$
     (d) cost for all: 2
     (e) priority:
     $$D_1 = 1 + 1 + \frac{3}{3} = 3.$$
     $$C_2 = 1 + 1 + \frac{3}{3} = 3.$$
     $$D_3 = 1 + 1 + \frac{1}{3} = \frac{7}{3}.$$
     (f) PQ: $\{\{B_4, \frac{5}{3}, 1, [B_2]\}, \{D_3, \frac{7}{3}, 2, [B_2, D_3]\}, \{D_1, 3, 2, [B_2, D_1]\}, \{C_2, 3, 2, [B_2, C_2]\}\}$
   - Step 3:
     (a) Pop B_4 since it has lowest priority
     (b) it is not goal, continue
     (c) Neighbors: $\{B_3, A_4, C_4\}$
     (d) costs are all 2
     (e) priority:
     $$A_4 = 1 + 1 + 1 = 3.$$
     $$B_3 = 1 + 1 + 1 = 3.$$
     $$C_4 = 1 + 1 + \frac{1}{3} = \frac{7}{3}.$$
     (f) PQ: $\{\{C_4, \frac{7}{3}, 2, [B_2, B_4]\}, \{D_3, \frac{7}{3}, 2, [B_2, D_3]\}, \{A_4, 3, 2, [B_2, B_4]\}, \{B_3, 3, 2, [B_2, B_4]\}, \{D_1, 3, 2, [B_2, D_1]\}, \{C$
   - Step 4:
     (a) Pop C_4 lowest and first in queue
     (b) Not answer, continue
     (c) Neighbors: $\{C_2, A_4\}$

(d) cost: 3

(e) priority: both are already in queue and with lower priority numbers, so don't add to queue

(f) PQ: $\{D_3, \frac{7}{3}, 2, [B_2, D_3]\}, \{A_4, 3, 2, [B_2, B_4]\}, \{B_3, 3, 2, [B_2, B_4]\}, \{D_1, 3, 2, [B_2, D_1]\}, \{C_2, 3, 2, [B_2, C_2]\}\}$

- Step 5:
  (a) Pop D_3 lowest Priority
  (b) Not answer, continue
  (c) Neighbors: $\{A_3\}$
  (d) cost: 3
  (e) priority:
  $$A_3 = 3 + \frac{4}{3} = \frac{13}{3}.$$
  (f) PQ: $\{\{A_4, 3, 2, [B_2, B_4]\}, \{B_3, 3, 2, [B_2, B_4]\}, \{D_1, 3, 2, [B_2, D_1]\}, \{C_2, 3, 2, [B_2, C_2]\}, \{A_3, \frac{13}{3}, 3, [B_2, D_2, D_3]$
- Step 6:
  (a) Pop A_4 lowest priority
  (b) Not answer, continue
  (c) Neighbor: $\{D_4\}$
  (d) cost: 3
  (e) priority: $3 + 0 = 3$
  (f) PQ: $\{D_4, 3, 3, [B_2, B_4, A_4]\}\{B_3, 3, 2, [B_2, B_4]\}, \{D_1, 3, 2, [B_2, D_1]\}, \{C_2, 3, 2, [B_2, C_2]\}, \{A_3, \frac{13}{3}, 3, [B_2, D_2, D$
- Step 7:
- pop D_4
- is answer, we are done, return path!

3 Problem 3

1. The objective function would be the satisfaction of the consumer. The expresso machine could have a function that allows the user to rate the coffee that they just had on a scale from 1-10. Over some training set of data, the machine would then record the inputs (coffee weight, grind size, water pressure, brew time) and the output (user satisfaction), then an approach could be taken to fit the parameters to the datapoints, trying to predict what would give higher user scores. A **local search** algorithm could be used such as gradient ascent/hill climbing which would explore on the surface to find a local maximum.

2. In case that the user has specific tastes such as they prefer high water pressure with heavy beans, low grind, and low brew and rate it consistently high, but they in fact also have a different palate for low water pressure, small bean,s high grind, and low brew, which they also rate high and maybe even rate higher. In the first case, that would be a local maximum and the second case would be a global maximum. We would use variants such as random restarts to randomize the configurations and with enough iterations, converge to the global maximum. As the user continues to use the coffee machine, it's unlikely that they will continue to optimze their tastes, more likely they will already have an idea of what they like. So we could use simulated annealing to decrease the likelihood of accepting randomized states the longer they have the machine. Finally, we could vary step size, getting smaller as we approach a maximum and larger as the as we go away. This will be helpful when the user has a very precise, down to the milligram, taste bud so that inputs as they approach that maximum have to take smaller steps so as not to continuously overshoot and oscillate below the maximum.

3. We know the formula for simulated annealing is $e^{\frac{E(curr)-E(neigh)}{T\cdot d^t}}$ where t is the timestep, d is the decay constant, and T is the intial temperature. We want T, d such that

$$f(t) = e^{\frac{E(curr)-E(neigh)}{T\cdot d^0}} \geq .995.$$
$$f(t) = e^{\frac{E(curr)-E(neigh)}{T\cdot d^{35}}} \leq .005.$$

Let E the objective function defined as desired score − user score $= 10 - h(\vec{x})$ and we want to minimize this to 0. Assume that $E(curr) - E(neigh) = -.5$, i.e. the neighbor has a higher score by .5. Then,

$$e^{\frac{-.5}{T}} \geq .995$$
$$T \geq 99.74\ldots.$$

and,

$$e^{\frac{-.5}{T\cdot d^{35}}} \leq .005$$
$$e^{\frac{-.5}{99.74\ldots\cdot d^{35}}} \leq .005$$
$$d = .99997\ldots.$$

If we were to accept any worse configuration, then $E(curr) - E(neigh) < 0$. This give us,

$$e^{\frac{\Delta E}{T}} \geq .995$$
$$\Delta E \geq \ln(.995)T$$
$$T \geq \frac{\Delta E}{\ln(.995)}.$$

and,

$$e^{\frac{\Delta E}{T\cdot d^{35}}} \leq .005$$
$$\Delta E \leq T \cdot d^{35} \ln(.005)$$
$$\frac{\Delta E}{T \cdot \ln(.005)} \leq d^{35}.$$

As for our objective function, the highest value possible is 10 because the user rates on a scale of $0 - 10$, so the desired score (10) - user score (0) = 10, and the lowest possible score is 0, when the coffee is rated by the user to be 10/10. So $0 \geq \Delta E \geq -10$. This means in our case, we can narrows the bounds:

$$T \geq \frac{\Delta E}{\ln(.995)}$$
$$T \geq \frac{-10}{\ln(.995)}$$
$$T \geq 1,994.995\ldots.$$

and,

$$\frac{\Delta E}{T \cdot \ln(.005)} \leq d^{35}$$
$$(\frac{-10}{1994.995\ldots\cdot \ln(.005)})^{\frac{1}{35}} \leq d$$
$$.96179 \leq d.$$

4 Academic Integrity

I have read and understood the academic integrity policy as outlined in the course syllabus for CS4100. By pasting this acknowledgment in my submission, I declare that all my work presented here is my own and any conceptual discussions with my classmates have been fully disclosed. I declare that generative AI was not used to answer any questions in this assignment. Any use of generative AI to improve writing clarity alone is accompanied by an appendix with my original unedited answers.