CS 188 Spring 2019 Midterm Review Search Solutions

Q1. Search

Each True/False question is worth 1 points. Leaving a question blank is worth 0 points. **Answering incorrectly is worth −1 points.**

(a) Consider a graph search problem where for every action, the cost is at least ϵ, with $\epsilon > 0$. Assume the used heuristic is consistent.

(i) [*true* or *false*] Depth-first graph search is guaranteed to return an optimal solution. False. Depth first search has no guarantees of optimality. Further, it measures paths in length and not cost.

(ii) [*true* or *false*] Breadth-first graph search is guaranteed to return an optimal solution. False. Breadth first search has no guarantees of optimality unless the actions all have the same cost, which is not the case here.

(iii) [*true* or *false*] Uniform-cost graph search is guaranteed to return an optimal solution. True. UCS expands paths in order of least total cost so that the optimal solution is found.

(iv) [*true* or *false*] Greedy graph search is guaranteed to return an optimal solution. False. Greedy search makes no guarantees of optimality. It relies solely on the heuristic and not the true cost.

(v) [*true* or *false*] A* graph search is guaranteed to return an optimal solution. True, since the heuristic is consistent in this case.

(vi) [*true* or *false*] A* graph search is guaranteed to expand no more nodes than depth-first graph search. False. Depth-first graph search could, for example, go directly to a sub-optimal solution.

(b) Let $h_1(s)$ be an admissible A* heuristic. Let $h_2(s) = 2h_1(s)$. Then:

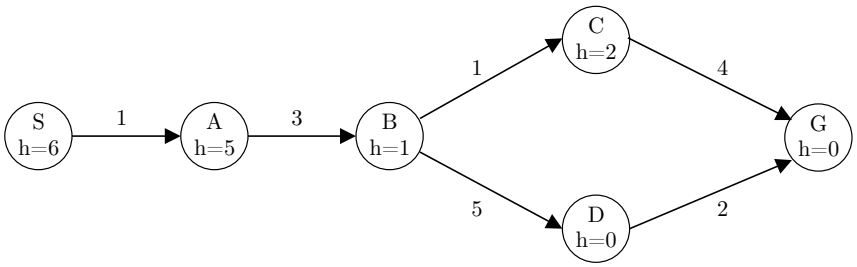
(i) [*true* or *false*] The solution found by A* tree search with h_2 is guaranteed to be an optimal solution. False. h_2 is not guaranteed to be admissible since only one side of the admissibility inequality is doubled.

(ii) [*true* or *false*] The solution found by A* tree search with h_2 is guaranteed to have a cost at most twice as much as the optimal path. True. In A* tree search we always have that as long as the optimal path to the goal has not been found, a prefix of this optimal path has to be on the fringe. Hence, if a non-optimal solution is found, then at time of popping the non-optimal path from the fringe, a path that is a prefix of the optimal path to the goal is sitting on the fringe. The cost $\bar{g}$ of a non-optimal solution when popped is its f-cost. The prefix of the optimal path to the goal has an f-cost of $g + h_0 = g + 2h_1 \leq 2(g + h_1) \leq 2C^*$, with C^* the optimal cost to the goal. Hence we have that $\bar{g} \leq 2C^*$ and the found path is at most twice as long as the optimal path.

(iii) [*true* or *false*] The solution found by A* graph search with h_2 is guaranteed to be an optimal solution. False. h_2 is not guaranteed to be admissible and graph search further requires consistency for optimality.

(c) The heuristic values for the graph below are not correct. For which single state (S, A, B, C, D, or G) could you change the heuristic value to make everything admissible and consistent? What range of values are possible to make this correction?

State: B Range: [2,3]


S
h=6
1
A
h=5
3
B
h=1
1
C
h=2
4
5
D
h=0
2
G
h=0

Q2. Formulation: Holiday Shopping

You are programming a holiday shopping robot that will drive from store to store in order to buy all the gifts on your shopping list. You have a set of N gifts $G = \{g_1, g_2, \ldots g_N\}$ that must be purchased. There are M stores, $S = \{s_1, s_2, \ldots s_M\}$ each of which stocks a known inventory of items: we write $g_k \in s_i$ if store s_i stocks gift g_k. Shops may cover more than one gift on your list and will never be out of the items they stock. Your home is the store s_1, which stocks no items.

The actions you will consider are travel-and-buy actions in which the robot travels from its current location s_i to another store s_j in the fastest possible way and buys whatever items remaining on the shopping list that are sold at s_j. The time to travel-and-buy from s_i to s_j is $t(s_i, s_j)$. You may assume all travel-and-buy actions represent shortest paths, so there is no faster way to get between s_i and s_j via some other store. The robot begins at your home with no gifts purchased. You want it to buy all the items in as short a time as possible and return home.

(a) What is one possible state space for this planning problem? One solution is a state space where each state is a pair (s, u) where s is the current location and u is the set of unpurchased gifts on your list (so $g \in u$ indicates that gift g has not yet been purchased).

(b) How large is the state space in terms of the quantities defined above? $M \times 2^N$. You are in one of M places (simple index from 1 to M), and have not purchased some subset of N items (binary vector of size N).

(c) For each of the following heuristics, which apply to states (s, u), circle whether it is admissible, consistent, neither, or both. Assume that the minimum of an empty set is zero.

([neither] / admissible / consistent / both) The shortest time from the current location to any other store: $\min_{s' \neq s} t(s, s')$

(neither / admissible / consistent / [both]) The time to get home from the current location: $t(s, s_1)$

(neither / admissible / consistent / [both]) The shortest time to get to any store selling any unpurchased gift: $\min_{g \in u}(\min_{s':g \in s'} t(s, s'))$

(neither / [admissible] / consistent / both) The shortest time to get home from any store selling any unpurchased gift: $\min_{g \in u}(\min_{s':g \in s'} t(s', s_1))$

([neither] / admissible / consistent / both) The total time to get each unpurchased gift individually: $\sum_{g \in u}(\min_{s':g \in s'} t(s, s'))$

([neither] / admissible / consistent / both) The number of unpurchased gifts times the shortest store-to-store time: $|u|(\min_{s_i, s_j \neq s_i} t(s_i, s_j))$

Remember, a consistent heuristic doesn't decrease from state to state by more than it actually costs to get from state to state. And of course, a heuristic is admissible if it is consistent. If you're confused, remember: the problem defines the minimum of an empty set as 0.

1. This heuristic does not return 0 in the goal state (s_1, $\emptyset$), since it gives the minimum distance to any store *other than the current one.*
2. We'll always need to get home from any state; the distance to home from home is 0; and this heuristic does not decrease by more than it costs to get from state to state.
3. We'll always need to get that last unpurchased item, and taking the min distance store guarantees that we underestimate how much distance we actually have to travel. It is consistent because the heuristic never diminishes by more than what is travelled.
4. We'll always need to get home from getting the last unpurchased item, and taking the min underestimates the actual requirement. What makes this heuristic inconsistent is that when we visit the last store to pick up the last unfinished item, the value of the heuristic goes to 0. Let's say the graph looks like this: s_3 `--1-->` s_2 `----5----->` s_1, with s_2 containing the last item. From s_3, the

heuristic is 5, but from s_2, the heuristic is now 0, meaning that traveling from s_3 to s_2 decreases the heuristic by 5 but the actual cost is only 1.

5. This can overestimate the actual amount of work required.
6. Same.

You have waited until very late to do your shopping, so you decide to send an swarm of R robot minions to shop in parallel. Each robot moves at the same speed, so the same store-to-store times apply. The problem is now to have all robots start at home, end at home, and for each item to have been bought by at least one robot (you don't have to worry about whether duplicates get bought). Hint: consider that robots may not all arrive at stores in sync.

(d) Give a minimal state space for this search problem (be formal and precise!) We need the location of each robot at each time. At a given time, a robot can either be at one of M stores, or in any of $(T-1)M$ transition locations, where T is the maximum travel distance between two stores. Thus, the location of each robot takes $(MT)^R$. We also need the set of items purchased (2^N). Therefore, the size of each state is: $(MT)^R \times 2^N$.

One final task remains: you still must find your younger brother a stuffed Woozle, the hot new children's toy. Unfortunately, no store is guaranteed to stock one. Instead, each store s_i has an initial probability p_i of still having a Woozle available. Moreover, that probability drops exponentially as other buyers scoop them up, so after t time has passed, s_i's probability has dropped to $\beta^t p_i$. You cannot simply try a store repeatedly; once it is out of stock, that store will stay out of stock. Worse, you only have a single robot that can handle this kind of uncertainty! Phrase the problem as a single-agent MDP for planning a search policy for just this one gift (no shopping lists). You receive a single reward of +1 upon successfully buying a Woozle, at which point the MDP ends (don't worry about getting home); all other rewards are zeros. You may assume a discount of 1.

(e) Give a minimal state space for this MDP (be formal and precise!) Which stores have been checked: 2^M
Whether Woozle has been bought: 2
Current time: T.

We may also want to keep track of the current location (M), but since there is no reward for traveling, we don't have to model that aspect of the problem.