CS 188
Spring 2020

Exam Prep 1 Solutions

Q1. Search problems

It is training day for Pacbabies, also known as Hungry Running Maze Games day. Each of k Pacbabies starts in its own assigned start location s_i in a large maze of size MxN and must return to its own Pacdad who is waiting patiently but proudly at g_i; along the way, the Pacbabies must, between them, eat all the dots in the maze.

At each step, all k Pacbabies move one unit to any open adjacent square. The only legal actions are Up, Down, Left, or Right. It is illegal for a Pacbaby to wait in a square, attempt to move into a wall, or attempt to occupy the same square as another Pacbaby. To set a record, the Pacbabies must find an optimal collective solution.

(a) Define a minimal state space representation for this problem.

The state space is defined by the current locations of k Pacbabies and, for each square, a Boolean variable indicating the presence of food.

(b) How large is the state space?

$(MN)^k \cdot 2^{MN}$

(c) What is the maximum branching factor for this problem?

[✓ 4^k] ○ $4^k 2^{MN}$
○ 8^k ○ $4^k 2^4$

Each of k Pacbabies has a choice of 4 actions.

(d) Let $MH(p,q)$ be the Manhattan distance between positions p and q and F be the set of all positions of remaining food pellets and p_i be the current position of Pacbaby i. Which of the following are admissible heuristics?

[✓ h_A: $\frac{\sum_{i=1}^{k} MH(p_i,g_i)}{k}$]
[✓ h_B: $\max_{1\le i\le k} MH(p_i,g_i)$]
☐ h_C: $\max_{1\le i\le k}[\max_{f\in F} MH(p_i,f)]$
☐ h_D: $\max_{1\le i\le k}[\min_{f\in F} MH(p_i,f)]$
[✓ h_E: $\min_{1\le i\le k}[\min_{f\in F} MH(p_i,f)]$]
☐ h_F: $\min_{f\in F}[\max_{1\le i\le k} MH(p_i,f)]$

h_A is admissible because the total Pacbaby–Pacdad distance can be reduced by at most k at each time step.
h_B is admissible because it will take at least this many teps for the furthest Pacbaby to reach its Pacdad.
h_C is inadmissible because it looks at the distance from each Pacbaby to its most distant food square; but of course the optimal solution might another Pacbaby going to that square; same problem for h_D.
h_E is admissible because some Pacbaby will have to travel at least this far to eat one piece of food (but it's not very accurate).
h_F is inadmissible because it connects each food square to the most distant Pacbaby, which may not be the one who eats it.

A different heuristic, $h_G = \max_{f \in F}[\min_{1 \le i \le k} MH(p_i, f)]$, *would* be admisible: it connects each food square to its closest Pacbaby and then considers the most difficult square for any Pacbaby to reach.

Now suppose that some of the squares are flooded with water. In the flooded squares, it takes two timesteps to travel through the square, rather than one. However, the Pacbabies don't know which squares are flooded and which aren't, until they enter them. After a Pacbaby enters a flooded square, its howls of despair instantly inform all the other Pacbabies of this fact.

(e) Define a minimal space of belief states for this problem.

The physical states about which the agent is uncertain are configurations of MN wetness bits, of which there are 2^{MN}. In general, the space of belief states would be all possible subsets of the physical states, i.e., $2^{2^{MN}}$ subsets of the 2^{MN} configurations. However, percepts in this world give either no information about a location or perfect information, so the reachable belief states are those 3^{MN} belief states in which each square is wet, dry, or unknown. Either answer is OK.

(f) How many possible environmental configurations are there in the initial belief state, before the Pacbabies receive any wetness percepts?

2^{MN}

(g) Given the current belief state, how many different belief states can be reached in a single step?

○ 4^k	○ $4^k 2^{MN}$
[✓ 8^k]	○ $4^k 2^4$

After each of 4^k joint movements of Pacbabies, there are 2^k possible joint percepts, each leading to a distinct belief state.

Q2. Search

(a) Rubik's Search

Note: You do not need to know what a Rubik's cube is in order to solve this problem.

A Rubik's cube has about 4.3×10^{19} possible configurations, but any configuration can be solved in 20 moves or less. We pose the problem of solving a Rubik's cube as a search problem, where the states are the possible configurations, and there is an edge between two states if we can get from one state to another in a single move. Thus, we have 4.3×10^{19} states. Each edge has cost 1. Note that the state space graph does contain cycles. Since we can make 27 moves from each state, the branching factor is 27. Since any configuration can be solved in 20 moves or less, we have $h^*(n) \leq 20$.

For each of the following searches, estimate the approximate number of states expanded. Mark the option that is closest to the number of states expanded by the search. Assume that the shortest solution for our start state takes exactly 20 moves. Note that 27^{20} is much larger than 4.3×10^{19}.

(i) DFS Tree Search

Best Case:	[✓ 20]	4.3×10^{19}	27^{20}	∞ (never finishes)
Worst Case:	20	4.3×10^{19}	27^{20}	[✓ ∞ (never finishes)]

(ii) DFS graph search

Best Case:	[✓ 20]	4.3×10^{19}	27^{20}	∞ (never finishes)
Worst Case:	20	[✓ 4.3×10^{19}]	27^{20}	∞ (never finishes)

(iii) BFS tree search

Best Case:	20	4.3×10^{19}	[✓ 27^{20}]	∞ (never finishes)
Worst Case:	20	4.3×10^{19}	[✓ 27^{20}]	∞ (never finishes)

(iv) BFS graph search

Best Case:	20	[✓ 4.3×10^{19}]	27^{20}	∞ (never finishes)
Worst Case:	20	[✓ 4.3×10^{19}]	27^{20}	∞ (never finishes)

(v) A* tree search with a perfect heuristic, $h^*(n)$, Best Case

[✓ 20] 4.3×10^{19} 27^{20} ∞ (never finishes)

(vi) A* tree search with a bad heuristic, $h(n) = 20 - h^*(n)$, Worst Case

20 4.3×10^{19} [✓ 27^{20}] ∞ (never finishes)

(vii) A* graph search with a perfect heuristic, $h^*(n)$, Best Case

[✓ 20] 4.3×10^{19} 27^{20} ∞ (never finishes)

(viii) A* graph search with a bad heuristic, $h(n) = 20 - h^*(n)$, Worst Case

20 [✓ 4.3×10^{19}] 27^{20} ∞ (never finishes)

(b) Limited A^* Graph Search

Consider a variant of A^* graph search called Limited A^* graph search. It is exactly like the normal algorithm, but instead of keeping all of the fringe, at the end of each iteration of the outer loop, the fringe is reduced to just a certain amount of the best paths. I.e. after all children have been inserted, the fringe is cut down to the a certain length. The pseudo-code for normal A^* graph search is reproduced below, the only modification being an argument W for the limit.

```
 1: function A* GRAPH SEARCH(problem, W)
 2:     fringe ← an empty priority queue
 3:     fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
 4:     closed ← an empty set
 5:     ADD INITIAL-STATE[problem] to closed
 6:     loop
 7:         if fringe is empty then
 8:             return failure
 9:         end if
10:         node ← REMOVE-FRONT(fringe)
11:         if GOAL-TEST(problem, STATE[node]) then
12:             return node
13:         end if
14:         if STATE[node] not in closed then
15:             ADD STATE[node] to closed
16:             for successor in GETSUCCESSORS(problem, STATE[node]) do
17:                 fringe ← INSERT(MAKE-SUCCESSOR-NODE(successor, node), fringe)
18:             end for
19:         end if
20:         fringe = fringe[0:W]
21:     end loop
22: end function
```

(i) For a positive W, limited A^* graph search is complete.

True [✓ False]

(ii) For a positive W, limited A^* graph search is optimal.

True [✓ False]

(iii) Provide the smallest value of W such that this algorithm is equivalent to normal A^* graph search (i.e. the addition of line 20 makes no difference to the execution of the algorithm).

W = Size of the State Space

Q3. Pacman's Life

Suppose a maze has height M and width N and there are F food pellets at the beginning. Pacman can move North, South, East or West in the maze.

(a) In this subquestion, the position of Pacman is known, and he wants to pick up all F food pellets in the maze. However, Pacman can move North at most two times overall.

What is the size of a minimal state space for this problem? Give your answer as a product of terms that reference problem quantities such as (but not limited to) M, N, F, etc. Below each term, state the information it encodes. For example, you might write $4 \times MN$ and write number of directions underneath the first term and Pacman's position under the second.
$MN \times 2^F \times 3$. Pacman's position, a boolean vector representing whether a certain food pellet has been eaten, and the number of times Pacman has moved North (which could be 0, 1 or 2).

(b) In this subquestion, Pacman is lost in the maze, and does not know his location. However, Pacman still wants to visit every single square (he does not care about collecting the food pellets any more). Pacman's task is to find a sequence of actions which guarantees that he will visit every single square.

What is the size of a minimal state space for this problem? As in part(a), give your answer as a product of terms along with the information encoded by each term. You will receive partial credit for a complete but non-minimal state space.
$2^{((MN)^2)}$. For every starting location, we need a boolean for every position (MN) to keep track of all the visited locations. In other words, we need MN sets of MN booleans for a total of $(MN)^2$ booleans. Hence, the state space is $2^{((MN)^2)}$.