Fall 2018
COMS 4701 Artificial Intelligence

Midterm Exam

INSTRUCTIONS

- You have 75 minutes.
- Do not turn the page until the official start time.
- The exam is closed book, closed notes except a one-page, double-sided crib sheet.
- Write your UNI on the top of **each** page where indicated.
- Write all your answers in the space provided for each question. We will not look at any work outside the provided exam pages. You may use and detach the scratch sheet at the end of the exam for scratch work.
- Questions are not sequenced in order of difficulty. Make sure to look ahead if stuck on a particular question.
- If you finish early, you may turn in your exam if you can minimize disturbances to other students. Please wait in your seat if there are fewer than 10 minutes left.

Last Name	
First Name	
UNI	
All the work on this exam is my own. **(please sign)**	

For staff use only

Q. 1	Q. 2	Q. 3	Q. 4	Q. 5	Total
/18	/18	/24	/ 22	/ 18	/100

THIS PAGE INTENTIONALLY LEFT BLANK

1. **(18 points) Search Problems**

(a) **(6 pt)** You are planning a dinner party, and you have a shopping robot to help you. The robot has a list of K stores to visit. Call these X_i, where $i \in \{1, 2, \ldots, K\}$. The robot also has an internal map of distances between all pairs of stores D_{ij}. We want the robot to start at the house X_0, visit all K stores, and come back to the house X_0 while minimizing the total distance traveled.

i. **(2 pt)** Circle all properties that apply to this problem; no explanations are needed.

Fully observable **Deterministic** **Sequential** **Dynamic** **Discrete** **Multi-agent**

ii. **(2 pt)** Provide an efficient state space formulation for the problem.

Robot's current location X_t, K Booleans indicating whether each store has been visited

iii. **(1 pt)** What are the start state and goal test (or terminal state) for your formulation?

Start state: X_0, K False Booleans. Goal test: $X_t == X_0$ && All K Booleans == True

iv. **(1 pt)** Circle the approach below that is *best* suited for solving this problem.

Informed (direct) search **CSP** **Minimax/expectimax trees** **MDP** **RL**

(b) **(6 pt)** After the party, all g guests G_i have to get home. Unfortunately that is easier said than done because it's late and public transit is not an option. Luckily, there are c designated drivers that can take different guests in their cars C_i. Of course there are some limitations—certain guests dislike each other and cannot share cars, while some drivers and guests live too far apart to feasibly drive them back. Each of these preferences can be summarized as a single statement P_i.

i. **(2 pt)** Circle all properties that apply to this problem; no explanations are needed.

Fully observable **Deterministic** **Sequential** **Dynamic** **Discrete** **Multi-agent**

ii. **(2 pt)** Provide an efficient state space formulation for the problem.

List of all current assignments to each guest G_i.

iii. **(1 pt)** What are the start state and goal test (or terminal state) for your formulation?

Start state: All guest variables *unassigned*. Goal test: Are all guests g_i assigned to some car C_j, while all constraints P_k are satisfied?

iv. **(1 pt)** Circle the approach below that is *best* suited for solving this problem.

Informed (direct) search **CSP** **Minimax/expectimax trees** **MDP** **RL**

(c) **(6 pt)** It's the morning after and your house is a mess. Luckily you have a cleaning robot at hand. Unfortunately, there is so much trash everywhere that the robot can't rely on its previous knowledge of the room layouts, so it will just have to visit each room and determine how to clean as it goes. The robot is also now uncertain of the efficacy of its actions; a single sweep of a room may not sufficiently clean it. The robot does have a "cleanliness" sensor that rates each room's cleanliness as an integer percentage between 0% and 100%. Suppose we place the robot in room R_0, which is 0% clean, and instruct it to visit and clean each room R_i until all K of them are 100% clean.

i. **(2 pt)** Circle all properties that apply to this problem; no explanations are needed.

Fully observable **Deterministic** **Sequential** **Dynamic** **Discrete** **Multi-agent**

ii. **(2 pt)** Provide an efficient state space formulation for the problem.

Robot's current location R_t, K integers indicating cleanliness percentage of each room

iii. **(1 pt)** What are the start state and goal test (or terminal state) for your formulation?

Start state: R_0, all room cleanliness levels = 0 (unknown cleanliness for rooms other than R_0 also acceptable since problem is not fully observable). Terminal states: All room cleanliness levels at 100 (robot's location does not matter).

iv. **(1 pt)** Circle the approach below that is *best* suited for solving this problem.

Informed (direct) search **CSP** **Minimax/expectimax trees** **MDP** **RL**

2. (18 points) Cryptarithmetic

We are going to consider a simplified version of the cryptarithmetic problem:

$$\begin{array}{cccc} & T & W & O \\ + & T & W & O \\ \hline F & O & U & R \end{array}$$

The solution to this problem is to generally assign values between 0 and 9 to each of the letters. For this problem instance, we need to satisfy the following requirements.

- All letters must have different values.
- $O < 5$ and $W < 5$.
- The values must satisfy the addition operation of two identical three-digit numbers above.

For example, a valid assignment would be $O = 2$, $R = 4$, $W = 3$, $U = 6$, $T = 1$, and $F = 0$.

(a) (4 pt) List all binary constraints that are a result of the addition structure. Do not list inequality constraints.

$2O = R$, $2W = U$, $2T \% 10 = O$, $F = 0$ if $T < 5$, $F = 1$ if $T \geq 5$.

(b) (4 pt) Draw the *complete* constraint graph for this problem, accounting for both inequality and addition constraints. Do not include arrows in your arcs.

This is simply a completely connected graph of six nodes, one for each variable. (Graph with constraint nodes as discussed in lecture is also acceptable.)

(c) (6 pt) Prior to assigning any variables, we can use both unary constraints and arc consistency due to binary constraints to reduce the domains of the problem. In the table below, cross out all domain values that are eliminated.

O	0	1	2	3	4	5	6	7	8	9
R	0	1	2	3	4	5	6	7	8	9
W	0	1	2	3	4	5	6	7	8	9
U	0	1	2	3	4	5	6	7	8	9
T	0	1	2	3	4	5	6	7	8	9
F	0	1	2	3	4	5	6	7	8	9

A systematic way to do this is as follows. First apply all unary constraints to reduce O, W, and F. 0 should also be eliminated from all domains except for F due to inequality constraints. The addition constraints require that R, U, and O are even. At this point O should only have 2 and 4 left.

Propagating the constraints $2O = R$ and $2T = O$, we are left with 4 and 8 for R and 1, 2, 6, and 7 for T.

(d) (2 pt) What is the size of this problem's state space after the single domain reduction and arc consistency step above?

$2^3 \times 4^3 = 512$

(e) (2 pt) If we choose to assign the first variable using the minimum remaining values (MRV) heuristic, which variable(s) would get chosen? List all such variables if there is a tie.

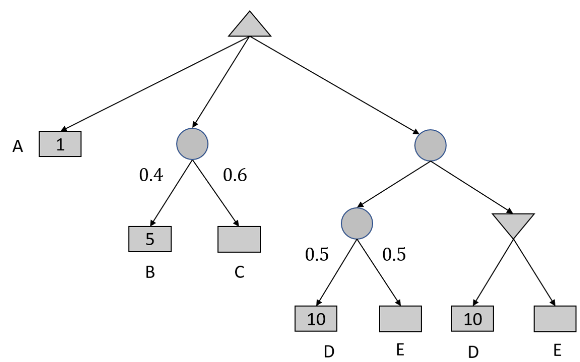
F, O, R

3. (24 points) Difficult Choices

You are playing a strange game where you have to choose to open one of several boxes to get some monetary reward inside. But they may also contain negative value, which means that you will lose money if you choose it. Your options are as follows:

- You can choose box A, which has a reward of \$1.
- You can choose to flip an unfair coin to obtain either box B or C. You know that you can get box B, which has a value of \$5, with 40% chance, but you cannot see inside box C.
- You let the host choose for you. There is a 50% chance that the host will choose randomly between boxes D and E, and a 50% chance that the host will give you the minimum between boxes D and E. The host knows the values of all boxes. You know that box D has a reward of \$10 but you know nothing about E.

(a) (8 pt) Draw the game tree for your options from your perspective of maximizing the reward. Be sure to use the appropriate shapes for max, min, and chance nodes. Label all leaf nodes with the letter of the corresponding box, and fill in any known rewards inside the leaf nodes. Indicate any probabilities on branches associated with chance nodes.



(b) (4 pt) Give the range of rewards that both C and E have to simultaneously satisfy for you to choose A in maximizing your expected rewards, or write *none* if you will never choose A.

Box A has a reward of 1, which has to be greater than the expected rewards of the other two options. We thus have two inequalities.

$$1 > 0.4(5) + 0.6C$$
$$1 > 0.5[0.5(10) + 0.5E] + 0.5\min(10, E)$$

We immediately see that $\mathbf{C < -1.67}$. The second equation will differ depending on whether E is less or greater than 10, but note that the equation cannot be satisfied for $E \geq 10$. So we must have $E < 10$ and satisfy

$$1 > 2.5 + 0.25E + 0.5E$$

This gives us $\mathbf{E < -2}$.

(c) (6 pt) Suppose we now know that C contains a reward of \$10. Give the range of rewards that E would have to take such that you would pass your decision to the host to maximize your expected reward, or write *none* if you would never do that.

If C = 10, then choosing the second option to flip a coin between B and C gives an expected reward of $0.4(5) + 0.6(10) = 8$. So we require the third option to be greater than 8 in order to choose it.

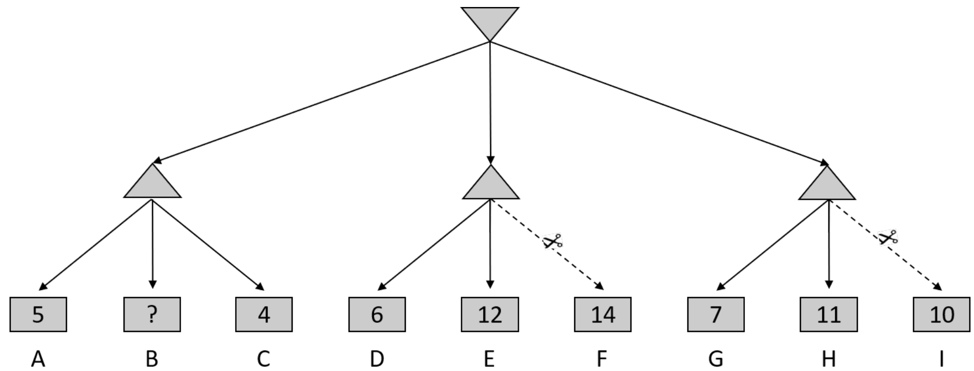
$$0.5[0.5(10) + 0.5E] + 0.5\min(10, E) > 8$$

Suppose E is smaller than 10. Then the equation simplifies to

$$2.5 + 0.25E + 0.5E > 8$$

giving us $\mathbf{E > \frac{5.5}{0.75}}$ or $\mathbf{E > 7.33}$. Note that this also includes the range $E > 10$, so we are done.

You're now playing a second, more straightforward game against the host where all randomness is removed. However, your goal is to *minimize* the reward you receive (for some reason). The game tree is shown below.



(d) (3 pt) If you were to traverse the tree left to right, what range of values can the leaf node B take to ensure that the indicated branches (with the scissors) get pruned?

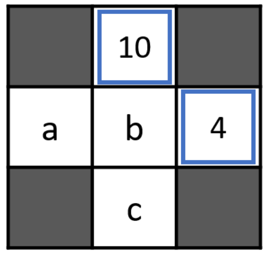
$7 < B \leq 11$

(e) (3 pt) Suppose the value of B is 8 and that you can traverse the successors of a given node in an order of your choosing. Give one ordering of the leaf nodes visited in which you are able to prune away the maximum number of branches possible. Do not include the pruned leaf nodes in your ordering.

Any ordering of A, B, C; one of (E,F) and one of (H,I), in either order.

4. (22 points) Mini-Gridworld

We are interested in the mini-gridworld shown below. The top and right squares are terminal states with the utilities shown. An agent can attempt to go in one of the four cardinal directions from any of the states **a**, **b**, and **c**. If the agent would land in a shaded space, the agent stays in its original space. All transitions incur a reward R. Assume all actions are deterministic for the first three questions below.



(a) (2 pt) True False States **a** and **c** will be equal at every iteration when performing value iteration.

(b) (2 pt) True False State **b** has a higher optimal value than states **a** and **c** for all R.

(c) (4 pt) What are the optimal state values for **a**, **b**, and **c** if $\gamma = 0.8$ and $R = -1$?

Optimal action from state b is to go up; optimal actions from states a and c are both to go to state b.

$$V^*(b) = -1 + 0.8(10) = 7$$

$$V^*(a) = V^*(c) = -1 + 0.8V^*(b) = 4.6$$

(d) (8 pt) Now suppose that the agent's actions are no longer deterministic. The agent succeeds in actually moving in the direction it intends to with probability P. Otherwise, it slips to either the left or the right of its intended direction, each with probability $\frac{1}{2}(1-P)$ (recall that if the transition were to place the agent in a grid with a wall, it stays in its current grid). If $\gamma = 1$ and $R = 0$, for what value of P would going up and going right from state **b** be equally optimal?

Since we want going up and going right to be equally good, the corresponding Q-values should be equal.

$$Q(b, \text{up}) = P(0 + 1(10)) + \frac{1}{2}(1-P)(4) + \frac{1}{2}(1-P)V^*(a)$$
$$Q(b, \text{right}) = P(0 + 1(4)) + \frac{1}{2}(1-P)(10) + \frac{1}{2}(1-P)V^*(c)$$

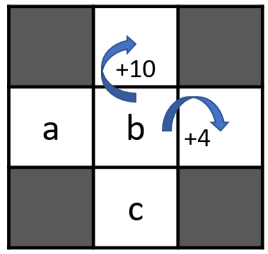
We equate the equations above and obtain

$$10P + 2(1-P) = 4P + 5(1-P)$$

where we eliminated the $V^*(a)$ and $V^*(c)$ terms since they are equal (by symmetry and from part (a)). We thus find that

$$P = \frac{1}{3}$$

(e) (6 pt) Mini-gridworld is modified so that the top and right states are no longer terminal. These states can be entered and exited repeatedly (just like **a**, **b**, and **c**), but an agent can obtain the rewards of either +10 or +4 *each time* it enters those two states from state b. All other living rewards R are 0.



If $\gamma = 0.8$ and all actions are again deterministic, what is the optimal value of state **b**?

Optimal action from state b is to go up (call this state x) and receive +10. Optimal action from state x is to go back to b (and receive +0). We have the following two Bellman equations:

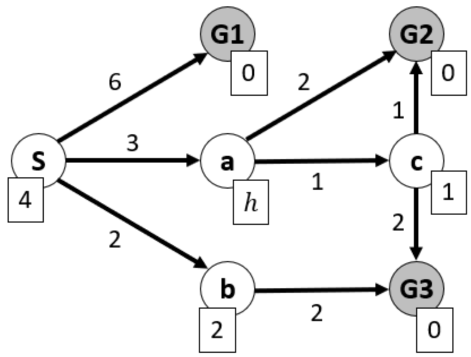
$$V^*(b) = 1.0(10 + \gamma V^*(x))$$
$$V^*(x) = 1.0(0 + \gamma V^*(b))$$

Plugging in $\gamma = 0.8$ and simplifying we have that

$$V^*(b) = \frac{10}{0.36}$$

5. (18 points) Search Redux

Consider the search tree shown below. The start state is S; goal states are shaded in gray. Costs between states are shown along edges, while heuristic values estimating the cost to a goal from each state are shown in the rectangles.



(a) (2 pt) Give an **optimal** solution that can be returned by depth-first search, or write *none* if it is impossible for DFS to return an optimal solution.

S - b - G3

(b) (2 pt) Give an **optimal** solution that can be returned by breadth-first search, or write *none* if it is impossible for BFS to return an optimal solution.

none

(c) (2 pt) Give a solution that can be returned by both depth-first and breadth-first search, or write *none* if DFS and BFS do not share any common solutions.

S - G1

(d) (3 pt) Give the range of non-negative values for the missing heuristic h that would allow greedy search to return the **optimal** solution, or write *none* if it is impossible for greedy to be optimal.

none

(e) (3 pt) Give the range of non-negative values for the missing heuristic h that would make the heuristic function consistent, or write *none* if it is impossible to do so.

$1 \leq h \leq 2$

(f) (3 pt) Give the range of non-negative values for the missing heuristic h that would make the heuristic function inadmissible but still allow A* to return an **optimal** solution, or write *none* if it is impossible to do so.

$h > 2$

(g) (3 pt) Give the range of non-negative values for the missing heuristic h that would allow A* to expand the same nodes in the same order as uniform-cost search, or write *none* if it is impossible to do so.

none (or 1 in a specific tie-breaking case)

THIS PAGE INTENTIONALLY LEFT BLANK

SCRATCH PAPER - DETACH BEFORE HANDING IN