CS 188 Fall 2014 Introduction to Artificial Intelligence Midterm Solutions

INSTRUCTIONS

- You have 80 minutes.
- The exam is closed book, closed notes except a one-page crib sheet.
- Please use non-programmable calculators only. Laptops, phones, etc., may not be used. If you have one in your bag please switch it off.
- Mark your answers ON THE EXAM ITSELF. If you are not sure of your answer you may wish to provide a *brief* explanation. All short answer sections can be successfully answered in a few sentences at most.
- Questions are not sequenced in order of difficulty. Make sure to look ahead if stuck on a particular question.

Last Name	
First Name	
SID	
Email	
First and last name of person on left	
First and last name of person on right	
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
/10	/30	/ 20	/ 20	/20	/100

1. (10 points) Agents

Here is pseudocode for three agent programs A, B, C:

```
function A(percept)
return f_A()
```

```
function B(percept)
return f_B(percept)
```

```
function C(percept)
persistent: percepts, initially []
percepts ← push(percept,percepts)
return f_C(percepts)
```

In each of these agents, the function f is some arbitrary, possibly randomized, function of its inputs with no internal state of its own; the agent program runs on computer with unbounded memory but finite clock speed. We'll assume also that the environment and its performance measure are computable.

(a) (3 pt) Suppose the environment is fully observable, deterministic, discrete, single-agent, and static. For which agents, if any, is it the case that, for *every* such environment, there is *some* way to choose f such that the agent is perfectly rational?

A **B** **C**

For a fully observable environment, only the current percept is required for an optimal decision. Because the environment is static, computation is not an issue. Note that Agent A cannot make optimal decisions because it always makes the *same* decision (or samples a decision from the same probability distribution), having no internal state.

(b) (3 pt) Suppose the environment is *partially* observable, deterministic, discrete, single-agent, and static. For which agents, if any, is it the case that, for *every* such environment, there is *some* way to choose f such that the agent is perfectly rational?

A **B** **C**

Agent B, the reflex agent, cannot always function optimally in a partially observable environment.

(c) (3 pt) Suppose the environment is partially observable, *stochastic*, discrete, single-agent, and *dynamic*. For which agents, if any, is it the case that, for *every* such environment, there is *some* way to choose f such that the agent is perfectly rational?

A **B** **C**

None of the agents can be optimal for an arbitrary dynamic environment, because we can make the environment complex enough to rended optimal decisions infeasible for any finite-speed machine.

(d) (1 pt) True/False: There is *some* environment such that *every* agent is perfectly rational in that environment.

There are many possible examples; perhaps the easiest is the environment whose performance measure is always 0; also environments with only one available action per state; etc.

2. (30 points) 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) (5 pt) 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) (2 pt) How large is the state space?
$(MN)^k \cdot 2^{MN}$

(c) (3 pt) What is the maximum branching factor for this problem?

(A) 4^k (B) 8^k (C) $4^k 2^{MN}$ (D) $4^k 2^4$
Each of k Pacbabies has a choice of 4 actions.

(d) (6 pt) 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.

(e) (2 pt) Give one pair of heuristics h_i, h_j from part (d) such that their *maximum* — $h(n) = max(h_i(n), h_j(n))$ — is an admissible heuristic.
Any pair from h_A, h_B, and h_E: the max of two admissible heuristics is admissible.

(f) **(2 pt)** Is there a pair of heuristics h_i, h_j from part (d) such that their *convex combination* — $h(n) = \alpha h_i(n) + (1 - \alpha)h_j(n)$ — is an admissible heuristic for any value of α between 0 and 1? Briefly explain your answer.
Any pair from h_A, h_B, and h_E: the convex combination of two admissible heuristics is dominated by the max, which is admissible.

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.

(g) **(4 pt)** 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.

(h) **(2 pt)** How many possible environmental configurations are there in the initial belief state, before the Pacbabies receive any wetness percepts?
2^{MN}

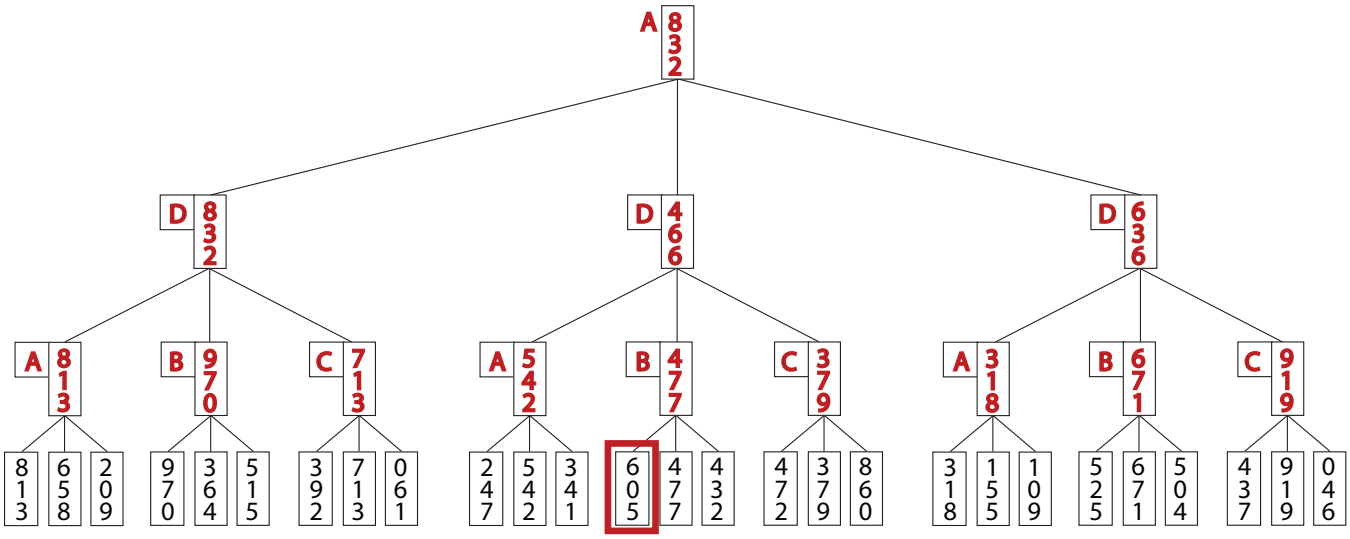
(i) **(4 pt)** Given the current belief state, how many different belief states can be reached in a single step?

(A) 4^k (B) 8^k (C) $4^k 2^{MN}$ (D) $4^k 2^4$
After each of 4^k joint movements of Pacbabies, there are 2^k possible joint percepts, each leading to a distinct belief state.

3. **(20 points) Adversarial search** Consider a game with three players (A, B, and C) in which, *before every move*, a fair 3-sided die is rolled to determine which player gets to make a move. (The sides of the die are marked A, B, and C.) The first die has been rolled, and it is A's turn to start the game. In every nonterminal state, the player whose turn it is has a choice of three moves. In a terminal state s, each player receives their own payoff from a payoff tuple $[U_A(s), U_B(s), U_C(s)]$; the aim of each player is to maximize the expected payoff he or she receives.

   (a) (8 pt) The skeleton of the game tree below corresponds to two turns of the game, after which the game ends and the values shown are attained.

   First, label each nonterminal node of the tree with A, B, C, or D. A, B, or C correspond to the respective player making a move, and D corresponds to a die roll. For any die roll node, label the outgoing branches A, B, C in that order. Then, propagate the values up the tree, assuming each player plays optimally and the die is fair.



   (b) (6 pt) Suppose you know in advance that each value must be in the range 0-9, as shown. Using this knowledge, circle the *first* leaf node that need not be evaluated, if the tree is explored left-to-right. [Hint: you may find it helpful to keep track of bounds on the values of the internal nodes as the search progresses.]

   The whole of the leftmost subtree of the root must be evaluated. In the middle subtree, the A branch must be fully evaluated because it might contain a 9. Once it has been evaluated to 5, the D node above it has a maximum value for A of (5+9+9)/3 = 23/3, i.e., less than 8; so the B branch need not be explored.

   (c) (6 pt) Now consider a version of the above game in which the moves are moves in an 8-puzzle, and a player receives +1 for making the final move that solves the puzzle, –1 if another player does so. If the same state is repeated 3 times, the game ends with everyone receiving 0. In the initial state it is A's turn to move and the puzzle is two steps from being solved. To keep things simple, remember that in an 8-puzzle every move takes you either one step closer or one step further away from the goal, so each player has essentially two choices and (to make this choice) we can use number-of-steps-from-goal as the state variable. Should A move towards the goal or away from it? Justify your answer *quantitatively*.

   If A moves towards the goal, whoever goes next will win; A has a 1/3 chance of going next, so A's expected payoff is $(1/3)(+1) + (2/3)(-1) = -1/3$. If it moves away, B and C must also (by symmetry) move away in the same circumstance, so no one will ever win. Because the state space is finite, some position will repeat and the payoff is 0, which is indeed better; so A will move away.

4. (20 points) Constraint satisfaction

Suppose you have a search problem defined by more or less the usual stuff:

- a set of states S;
- an initial state s_0;
- a set of actions A including the $NoOp$ action that has no effect;
- a transition model $Result(s, a)$;
- a set of goal states G.

Unfortunately, you have no search algorithms! All you have is a CSP solver.

(a) (10 pt) Given some time horizon T, explain how to formulate a CSP such that (1) the CSP has a solution exactly when the problem has a solution of length T steps; (2) the solution to the original problem can be "read off" from the variables assigned in CSP solution. Your formulation must give the variables, their domains, and all applicable constraints expressed as precisely as possible. You should have at least one variable per time step, and the constraints should constrain the initial state, the final state, and consecutive states along the way.

As stated, a CSP problem formulation consists of variables, domains, and constraints. The straightforward solution is to have variables $S_0, \ldots, S_T$ for the state at each time step, with the domain of each being S; and variables $A_0, \ldots, A_{T-1}$ for the action at each time step, with the domain of each being A; the constraints are

- $S_0 = s_0$;
- $S_T \in G$;
- For t from 0 to $T-1$, $S_{t+1} = Result(S_t, A_t)$.

Some students pointed out, correctly, that a solution that achieves the goal early can then add NoOp actions and return a solution with the goal true at T as well; could also add constraints disallowing goal at earlier steps and simply remove those constraints for the next part.

It's also possible to have just the S_t variables, but then the action sequence cannot be directly read off the solution and the transition constraint becomes more complex (you have to say *for some* $a \in A$, $S_{t+1} = Result(S_t, a)$).

There are still other formulations possible. E.g., (1) Variables can be each state with domain A, in which case the CSP finds a *policy* mapping states to actions; but the initial state cannot be represented and the goal constraint has to include *all* variables as well as encoding the transition model so that policies do achieve the goal from any state. (2) Variables can be any state with domain [0,...,T,-1], indicating the time at which the agent will be in that state (-1 for never); the transition constraint is on all pairs of variables, saying they can have consecutive time values iff one is the result of some action applied to the other. And so on!

(b) (2 pt) Explain how to modify your CSP formulation so that the CSP has a solution when the problem has a solution of length $\leq T$ steps, rather than exactly T steps.

This constraint can be written $(S_0 \in G) \vee (S_1 \in G) \vee \cdots \vee (S_T \in G)$; or note as above that $S_T \in G$ does allow earlier solutions via NoOp completion.

(c) (8 pt) Suppose you also know the step cost function $c(s, a, s')$ and you have a local-search CSP solver. What heuristic can you give the solver, and what local-search algorithm should it run, such that a globally optimal solution is returned (if one exists) of length $\leq T$?

The heuristic should be the total path cost (i.e., $\sum_{t=0}^{T-1} c(S_t, A_t, S_{t+1})$) plus some cost for violating any of the constraints. We guarantee that the global optimum is a solution by setting the cost of a violated constraint higher than the maximum possible path cost (T times the maximum step cost). With such a heuristic, we can use simulated annealing to find an optimal solution; hill-climbing with random restart would also work. Plain hill-climbing and local beam search are not gauranteed to find a global optimum.

5. (20 points) Propositional logic

(a) (6 pt) Consider a vocabulary with only four symbols, A, B, C, and D. For each of the following sentences, how many possible worlds make it true?

i. $(A \wedge B) \vee (C \wedge D)$
7 (4 for $A \wedge B$, 4 for $C \wedge D$, minus 1 for the model that satisfies both).

ii. $\neg(A \wedge B \wedge C \wedge D)$
15 — it's the negation of a sentence with 1 model.

iii. $B \Rightarrow (A \wedge B)$
12 — it's true when B is false (8) and when B is true and A is true (4).

(b) (8 pt) A certain procedure to convert a sentence to CNF contains four steps (1-4 below); each step is based on a logical equivalence. Circle ALL of the valid equivalences for each step.

i. Step 1: drop biconditionals

[a] $(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$
b) $(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \vee (\beta \Rightarrow \alpha))$
c) $(\alpha \Leftrightarrow \beta) \equiv (\alpha \wedge \beta)$

ii. Step 2: drop implications

a) $(\alpha \Rightarrow \beta) \equiv (\alpha \vee \neg\beta)$
[b] $(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta)$
c) $(\alpha \Rightarrow \beta) \equiv (\neg\alpha \wedge \beta)$

iii. Step 3: move not inwards

[a] $\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta)$
b) $\neg(\alpha \vee \beta) \equiv (\neg\alpha \vee \neg\beta)$
[c] $\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$

iv. Step 4: move "or" inwards and "and" outwards

a) $(\alpha \vee (\beta \wedge \gamma)) \equiv (\alpha \vee \beta \vee \gamma)$
[b] $(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$
c) $(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$

(c) **(4 pt)** A group of Stanford students write a Convert-to-CNF-ish procedure. In their implementation, they simply apply the *first* equivalence (the one labeled "a)") from each of the four steps in part (b). Show the transformed sentence generated by Convert-to-CNF-ish at each stage, when applied to the input sentence $A \Leftrightarrow (C \vee D)$.

$A \Leftrightarrow (C \vee D)$
$(A \Rightarrow (C \vee D)) \wedge ((C \vee D) \Rightarrow A)$
$(A \vee \neg(C \vee D)) \wedge ((C \vee D) \vee \neg A)$
$(A \vee (\neg C \wedge \neg D)) \wedge ((C \vee D) \vee \neg A)$
$(A \vee \neg C \vee \neg D)) \wedge (C \vee D \vee \neg A)$

(d) **(2 pt)** Is the final output of the Convert-to-CNF-ish equivalent to the input sentence in part (c)? If not, give a possible world where the input and output sentences have different values.

No.
A counterexample is any model where the two sentences have different truth values. The first clause in the final sentence says $(C \wedge D) \Rightarrow A$ rather than $(C \vee D) \Rightarrow A$. So countereamples are $\{A = false, C = true, D = false\}$ and $\{A = false, C = false, D = true\}$.