上 海 交 通 大 学 试 卷

(2019 2020 1 Academic Year/Fall Semester)

Class No.	Student ID No.
Name in English/Pinyin:	Name in Hanzi, if applicable:

Ve492 Introduction to Artificial Intelligence

Midterm Exam

October 28, 2018, 10:00am-11:40am

The exam paper has **17** pages in total.

Exam rules and information:

- The three main sections of this exam are independent.
- This is a closed book exam. You are allowed two sheets of A4 paper, which should be stapled, with your own handwritten notes. Photocopies are not allowed.
- The last two pages of the exam papers can be used as scratch papers. If you need more scratch papers, let the proctors know.
- No electronic devices are allowed. These include laptops, cell phones and smart watches.

You are to abide by the University of Michigan-Shanghai Jiao Tong University Joint Institute (UM-SJTU JI) honor code. Please sign below to signify that you have kept the honor code pledge.

THE UM-SJTU JI HONOR CODE

I accept the letter and spirit of the honor code:
I have neither given nor received unauthorized aid on this examination, nor have I concealed any violations of the Honor Code by myself or others.

Signature: ____________________

Please enter grades here:

Exercice No. 题号	Points 得分	Grader’s Signature 流水批阅人签名
1.1		
1.2		
1.3		
1.4		
2		
3.1		
3.2		
4		
5		
Total		

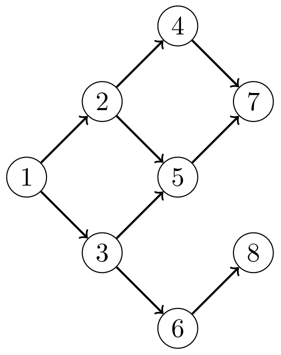


Figure 1: An 8-Node Graph.

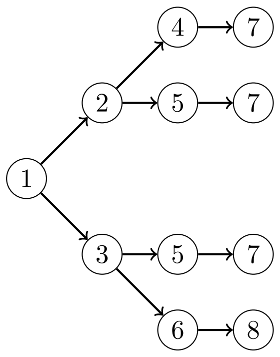
1 Search

1.1 Non-valued graph

Consider the graph depicted in Figure 1 interpreted as a state-space graph where the initial state is 1 and the goal state is 8.

1. (2 points) Draw the corresponding search tree.

Solution:



2. (2 points) List the states in the order they are visited by Breadth-First Search (BFS) by assuming that:

   - a state with a smaller index is visited first when there is a choice,
   - BFS does not keep a memory of visited states.

Solution: 1, 2, 3, 4, 5, 5, 6, 7, 7, 7, 8

1.2 Valued graph

Consider the graph of Figure 1 interpreted as a valued state-space graph where we assume that an edge is valued by the index of its source state, e.g., the cost of edge (i, j) is i. The initial state and the goal state are again 1 and 8 respectively.

1. (2 points) List the states in the order they are visited by Uniform-Cost Search (UCS) by assuming that:

   - a state with a smaller index is visited first when there is a choice,
   - UCS does not keep a memory of visited states.

   Solution: 1, 2, 3, 4, 5, 5, 6, 7, 7, 7, 8

2. (8 points) Assume that the optimal cost of a state that cannot reach the goal state is infinite.

   (a) (4 points) Define $h_1(i) = i$ if i has any successor and $h_1(i) = 0$ otherwise.

   i. (2 points) Is h_1 an admissible heuristic? Why?

   Solution: yes, because $h_1(i) = c(i, i') \leq c(i, i') + h^*(i')$

   ii. (2 points) Is h_1 a consistent heuristic? Why?

   Solution: yes, because $h_1(i) = c(i, i') \leq c(i, i') + h_1(i')$

   (b) (2 points) Define $h_2(i) =$ the length of the shortest path (in terms of number of edges) from i to the goal state. It is infinite if there is no such path. Does h_1 dominate h_2 or the other way around? Why?

   Solution: $h_1(3) \geq h_2(3)$ but $h_1(1) \leq h_2(1)$

   (c) (2 points) Provide an admissible heuristic h_4 that depends on h_1 and h_2 such that h_4 dominates $h_3(i) = \max(h_1(i), h_2(i))$ and is strictly greater in at least one state.

   Solution: $h_4(i) = \begin{cases} 0 & \text{if } h_2(i) = 0 \\ h_1(i) + h_2(i) - 1 & \text{otherwise} \end{cases}$

1.3 Problem Formalization

Formulate the Missionaries and Cannibals problem as a state-state graph. This problem is described as follows on Wikipedia:

Three missionaries and three cannibals must cross a river using a boat which can carry at most two people, under the constraint that, for both banks, if there are missionaries present on the bank, they cannot be outnumbered by cannibals (if they were, the cannibals would eat the missionaries). The boat cannot cross the river by itself with no people on board.

1. (4 points) Provide the definition of its state-space graph (i.e., states, successors of states, end states, initial state and goal state). You can provide successors as a function with conditions on its inputs/outputs. Make sure that your state does not contain redundant information.

Solution: A state can be represented as a triplet (n, m, b) which respectively indicates the numbers of missionaries, cannibals and boat at the initial river bank.

$$\text{successors of } (n,m,1) = \begin{cases} \left.\begin{array}{ll} (n-2,m,0) & \text{if } n-2 \geq 0 \\ (n,m-2,0) & \text{if } m-2 \geq 0 \\ (n-1,m-1,0) & \text{if } \begin{cases} n-1 \geq 0 \\ m-1 \geq 0 \end{cases} \end{array}\right\} & \text{moving 2 pers.} \\ \left.\begin{array}{ll} (n-1,m,0) & \text{if } n-1 \geq 0 \\ (n,m-1,0) & \text{if } m-1 \geq 0 \end{array}\right\} & \text{moving 1 pers.} \end{cases}$$

$$\text{successors of } (n,m,0) = \begin{cases} \left.\begin{array}{ll} (n+2,m,1) & \text{if } n+2 \leq 3 \\ (n,m+2,1) & \text{if } m+2 \leq 3 \\ (n+1,m+1,0) & \text{if } \begin{cases} n+1 \leq 3 \\ m+1 \leq 3 \end{cases} \end{array}\right\} & \text{moving 2 pers.} \\ \left.\begin{array}{ll} (n+1,m,0) & \text{if } n+1 \leq 3 \\ (n,m+1,0) & \text{if } m+1 \leq 3 \end{array}\right\} & \text{moving 1 pers.} \end{cases}$$

Initial state: $(3, 3, 1)$

Goal state: $(0, 0, 0)$

2. (2 points) Define a non-null admissible heuristic function for this problem.

Solution: Here is one possible heuristic function. By relaxing the constraint that the number of missionaries should be higher than that of cannibals at any banks, we obtain the following heuristic function:

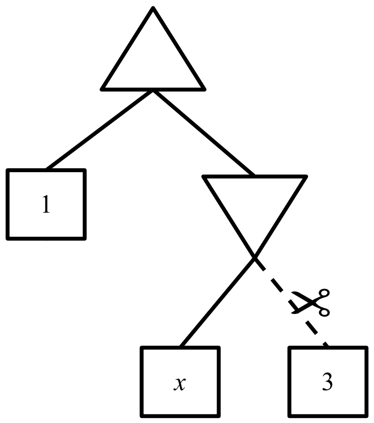
$$h(n,m,1) = \begin{cases} 1 & \text{if } n+m \leq 2 \\ 2*(n+m-2)+1 & \text{otherwise} \end{cases}$$

because when $n + m \geq 3$, we need two moves to transport one person to the other bank, and

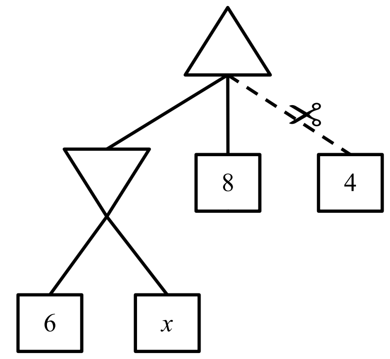
$$h(n,m,0) = \begin{cases} 0 & \text{if } n+m = 0 \\ h(n+1,m,1) = h(n,m+1,1) = 2*(n+m) & \text{otherwise} \end{cases}$$

1.4 Alpha-beta Pruning

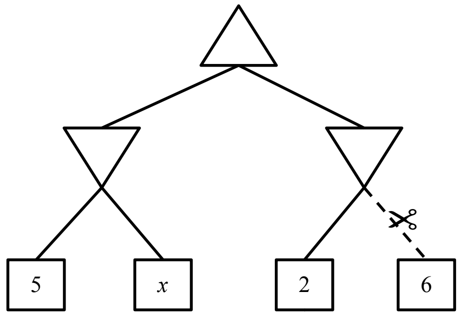
1. (10 points) For each of the game-trees shown below, state for which values of x the dashed branch with the scissors will be pruned. For instance, if pruning will happen for values of x larger than or equal to 4, write "$x \geq 4$". If pruning will happen for values of x smaller than or equal to 2, write "$x \leq 4$". If the pruning will not happen for any value of x write "none". If pruning will happen for all values of x write "all". We are assuming that nodes are evaluated left to right and ties are broken in favor of the latter nodes.



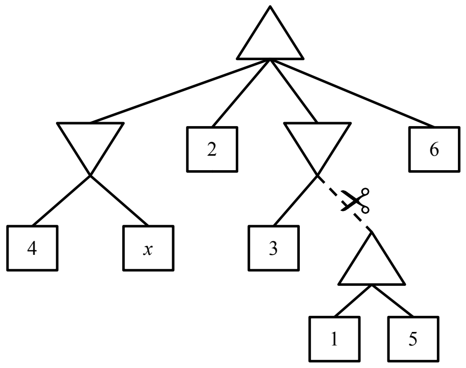
(a) Tree 1. Answer: $\underline{x \leq 1.}$



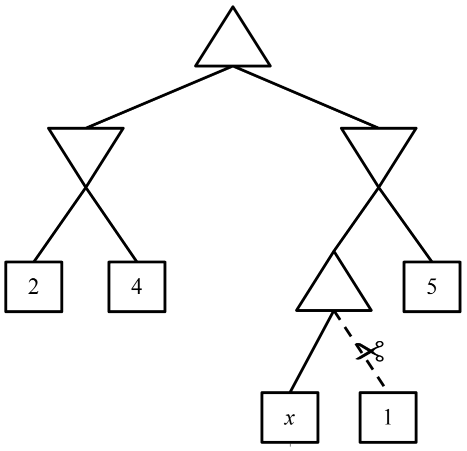
(a) Tree 2. Answer: <u>None</u>



(b) Tree 3. Answer: $\underline{x \geq 2}$



(c) Tree 4. Answer: $\underline{x \geq 3,}$



(d) Tree 5. Answer: <u>None</u>

2 Game Theory

Morra is a hand game that dates back thousands of years to ancient Roman and Greek times. We will study a simplified version of this game. Each player simultaneously reveals their hand, extending one or two fingers, and calls out a number (2, 3 or 4). If one player guessed correctly the total number of extended fingers and the other was wrong, the latter pays that number in dollars to the former. In all other cases, nobody pays anything.

We denote (k, s) the pure strategy that consists in extending k fingers and guessing s as the total number.

1. (4 points) Is it a zero-sum game? Express this game in normal form.

Solution:

Yes, therefore we can only give the payoffs for one of the player. Here's table for the row player.

	(1, 2)	(1, 3)	(2, 3)	(2, 4)
(1, 2)	0	2	-3	0
(1, 3)	-2	0	0	3
(2, 3)	3	0	0	-4
(2, 4)	0	-3	4	0

2. (6 points) Assume the row player plays a mixed strategy where the probabilities for strategies $(1, 2), (1, 3), (2, 3)$, and $(3, 4)$ are respectively denoted α, β, γ, and δ. Write a system of inequalities that expresses that this mixed strategy is a Nash equilibrium.

Solution:

$$\begin{aligned}
&-2\beta + 3\gamma \geq 0\\
&2\alpha - 3\delta \geq 0\\
&-3\alpha + 4\delta \geq 0\\
&3\beta - 4\gamma \geq 0\\
&\alpha + \beta + \gamma + \delta = 1\\
&\alpha \geq 0; \beta \geq 0; \gamma \geq 0; \delta \geq 0
\end{aligned}$$

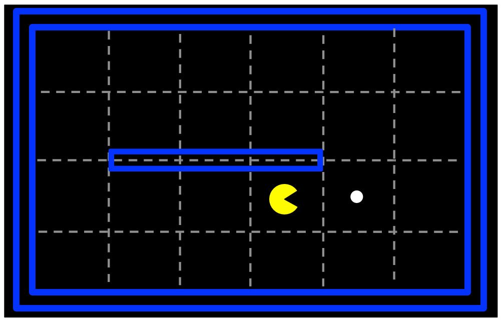
3 Markov Decision Process

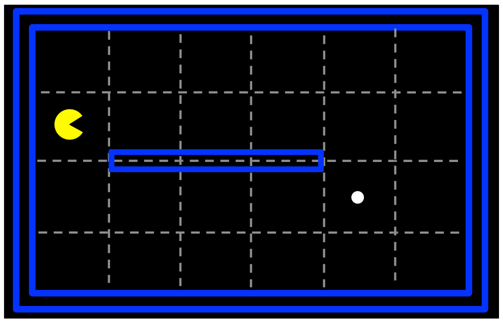
3.1 Solving Search Problems with MDPs

The following parts consider a Pacman agent in a deterministic environment. A goal state is reached when there are no remaining food pellets on the board. Pacman's available actions are $\{N, S, E, W\}$, but Pacman **can not** move into a wall. Whenever Pacman eats a food pellet he receives a reward of +1.

Assume that pacman eats a food pellet as soon as he occupies the location of the food pellet—i.e., the reward is received for the transition into the square with the food pellet.

Consider the particular Pacman board states shown below. Throughout this problem assume that $V_0(s) = 0$ for all states, s. Let the discount factor, $\gamma = 1$.


State A


State B

1. (2 points) What is the optimal value of state A, $V^*(A)$?

Solution: 1

2. (2 points) What is the optimal value of state B, $V^*(B)$?

Solution: 1
The reason the answers are the same for both (b) and (a) is that there is no penalty for existing. With a discount factor of 1, eating the food at any future step is just as valuable as eating it on the next step. An optimal policy will definitely find the food, so the optimal value of any state is always 1.

3. (2 points) At what iteration, k, will $V_k(B)$ first be non-zero?

Solution: 5
The value function at iteration k is equivalent to the maximum reward possible within k steps of the state in question, B. Since the food pellet is exactly 5 steps away from Pacman in state B, $V_5(B) = 1$ and $V_{K<5}(B) = 0$.

4. (2 points) How do the optimal q-state values of moving W and E from state A compare? (*choose one*)

   ○ $Q^*(A, W) > Q^*(A, E)$ ○ $Q^*(A, W) < Q^*(A, E)$ [✓ $Q^*(A, W) = Q^*(A, E)$]

Solution: Once again, since $\gamma = 1$, the optimal value of every state is the same, since the optimal policy will eventually eat the food.

5. (4 points) If we use this MDP formulation, is the policy found guaranteed to produce the shortest path from pacman's starting position to the food pellet? If not, how could you modify the MDP formulation to guarantee that the optimal policy found will produce the shortest path from pacman's starting position to the food pellet?

Solution: No. The Q-values for going $West$ and $East$ from state A are equal so there is no preference given to the shortest path to the goal state. Adding a negative living reward (example: -1 for every time step) will help differentiate between two paths of different lengths. Setting $\gamma < 1$ will make rewards seen in the future worth less than those seen right now, incentivizing Pacman to arrive at the goal as early as possible.

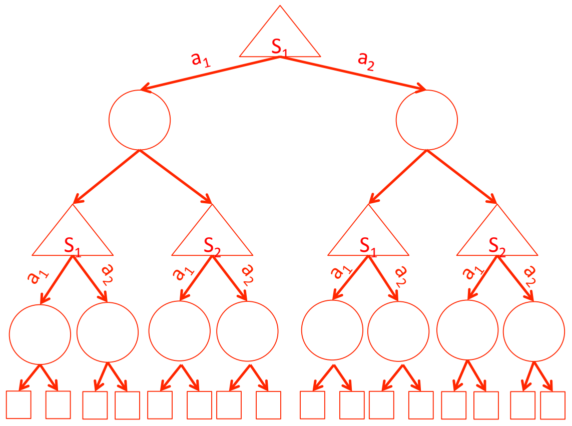
3.2 X Values

Instead of the Bellman update equation, consider an alternative update equation, which learns the X value function. The update equation, assuming a discount factor $\gamma = 1$, is shown below:

$$X_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s, a, s') \left[R(s, a, s') + \max_{a'} \sum_{s''} T(s', a', s'') \left[R(s', a', s'') + X_k(s'') \right] \right]$$

1. (6 points) Assuming we have an MDP with two states, S_1, S_2, and two actions, a_1, a_2, draw the expectimax tree rooted at S_1 that corresponds to the alternative update equation.

Solution:



The leaf nodes above will be the values of the previous iteration of the alternate update equation. Namely, if the value of the tree is $X_{k+1}(S_1)$, then the leaf nodes from left to right correspond to $X_k(S_1)$, $X_k(S_2)$, $X_k(S_1)$, $X_k(S_2)$, etc.

2. (4 points) Write the mathematical relationship between the X_k-values learned using the alternative update equation and the V_k-values learned using a Bellman update equation, or write *None* if there is no relationship.

Solution: $X_k(s) = V_{2k}(s), \forall s$
The thing to demonstrate here is that X is doing two-step lookahead relative to V.

Why?

$$X_0(s) = V_0(s)$$

Run an iteration to update X. This is the same as updating V for two iterations. Hence,

$$X_1(s) = V_2(s)$$

Run another iteration to update X. This is the same as updating V for two iterations. Hence,

$$X_2(s) = V_4(s)$$

.

...

Hence,

$$X_k(s) = V_{2k}(s)$$

.

4 Constraint Satisfaction Problem

A group of 7 students (Alice, Bob, Charlotte, David, Erin, Faye, and George) needs to complete a research project which can be divided in 6 tasks:

- task 1. Problem formulation
- task 2. Literature review
- task 3. Formulation of a novel method
- task 4. Programming the method
- task 5. Running experiments
- task 6. Result analysis

In order to be more efficient, they decide to allocate one task to each student. Because of different expertises, affinities, and enmities, the allocation should satisfy the following constraints:

(i) Bob (B) will not work on a task together with David (D).

(ii) Erin (E) must work on either Problem formulation, Literature review, or Formulation of a novel method.

(iii) Faye (F) can only contribute to Problem formulation, Formulation of a novel method, or Running experiments.

(iv) George (G) must work on a task that's before Faye (F)'s task.

(v) Erin (E) must work on a task that's before Bob (B)'s task.

(vi) Alice (A) can only work on Running experiments.

(vii) David (D) must work on a task that's after George (G)'s task.

(viii) If Alice (A) is to work with someone, it cannot be with George (G).

(ix) George (G) cannot work on task 6.

(x) Charlotte (C) cannot work on tasks 4, 5, or 6

(xi) Bob (B) cannot work on task 5.

(xii) Bob (B) must work on a task before Charlotte (C)'s task.

Note that for the problem to be feasible, a task should be allocated to at least one student. Therefore, given the number of tasks and students, at least one task will be shared by two students (and maybe more if the problem is infeasible).

1. (2 points) We model this problem as a constraint satisfaction problem (CSP). Our variables correspond to each of the students (A, B, C, D, E, F, and G), and the domains are the tasks (1, 2, 3, 4, 5, and 6). After applying the unary constraints, what are the resulting domains of each variable? Cross off the eliminated tasks. (The second grid with variables and domains is provided as a back-up in case you mess up on the first one.)

A	1	2	3	4	5	6
B	1	2	3	4	5	6
C	1	2	3	4	5	6
D	1	2	3	4	5	6
E	1	2	3	4	5	6
F	1	2	3	4	5	6
G	1	2	3	4	5	6

A	1	2	3	4	5	6
B	1	2	3	4	5	6
C	1	2	3	4	5	6
D	1	2	3	4	5	6
E	1	2	3	4	5	6
F	1	2	3	4	5	6
G	1	2	3	4	5	6

Solution:

A					5	
B	1	2	3	4		6
C	1	2	3			
D	1	2	3	4	5	6
E	1	2	3			
F	1		3		5	
G	1	2	3	4	5	

2. (2 points) If we apply the Minimum Remaining Value (MRV) heuristic, which variable should be assigned first?

Solution: Alice – because she has the least values left in his domain.

3. (3 points) Normally we would now proceed with the variable you found in (b), but to decouple this question from the previous one (and prevent potential errors from propagating), let's proceed with assigning Faye first. For value ordering we use the Least Constraining Value (LCV) heuristic, where we use *Forward Checking* to compute the number of remaining values in other variables domains. What ordering of values is prescribed by the LCV heuristic? Include your work—i.e., include the resulting filtered domains that are different for the different values.

Solution: Faye's value will be assigned as 5, 3, 1, in that order.

Why these variables? They are the only feasible variables for Faye. Why this order? This is the increasing order of the number of constraints on each variable.

The only binary constraint involving Faye is "George (G) must work on a task that's before Faye (F)'s task." So, only George's domain is affected by forward checking on these assignments, and it will change from {1, 2, 3, 4, 5} to {1, 2, 3, 4}, {1, 2}, and { } for the assignments 5, 3, 1, respectively.

4. (11 points) Realizing this is a tree-structured CSP, we decide not to run backtracking search, and instead use the efficient two-pass algorithm to solve tree-structured CSPs. We will run this two-pass algorithm after applying the unary constraints from question 1. Below is the linearized version of the tree-structured CSP graph for you to work with.

(a) (7 points) **First Pass: Domain Pruning.** Pass from *right to left* to perform Domain Pruning. Circle the values that remain in each domain below each node in the figure below.

$$\begin{tikzcd}
E \arrow[r] & B \arrow[r] \arrow[rr, bend left] & C & D \arrow[r] & G \arrow[r] \arrow[rr, bend left] & A & F
\end{tikzcd}$$

1	1	1	1	1	1	1
2	2	2	2	2	2	2
3	3	3	3	3	3	3
4	4	4	4	4	4	4
5	5	5	5	5	5	5
6	6	6	6	6	6	6

Solution: Remaining values in each domain after the domain pruning right-to-left pass:
Erin: 1
Bob: 1,2
Charlotte: 1,2,3
David: 2,3,4,5,6
George: 1,2,3,4
Alice: 5
Faye: 1,3,5

(b) (4 points) **Second Pass: Find Solution.** Pass from *left to right*, assigning values for the solution. If there is more than one possible assignment, choose the highest value.

Solution: Assigned Values after the left-to-right pass:
Erin: 1
Bob: 2
Charlotte: 3
David: 6
George: 4
Alice: 5
Faye: 5

5 General Questions

For each question, state if you agree or not and justify your answer.

1. (2 points) *Uniform-cost search* can be seen as a special case of the A* algorithm.

   Solution: Yes, UCS corresponds to A* when the heuristics function is taken as the constant function zero.

2. (2 points) Let $G = (V, E, c)$ be a valued state-space graph and define $\forall e \in E, c'(e) = c(e) - \min_{e' \in E} c(e')$ (therefore $\forall e \in E, c'(e) \geq 0$). An optimal solution (i.e., shortest path) in $G = (V, E, c')$ is also optimal for $G = (V, E, c)$.

   Solution: No, if there are two paths with a different number of edges, their costs will be changed differently when we subtract $\min_{e' \in E} c(e')$ to the cost of each edge.

3. (2 points) Let $G = (V, E, c)$ be a valued state-space graph and define $\forall e \in E, c''(e) = c(e)/\max_{e' \in E} c(e')$ (therefore $\forall e \in E, c''(e) \leq 1$). An optimal solution (i.e., shortest path) in $G = (V, E, c'')$ is also optimal for $G = (V, E, c)$.

   Solution: Yes if $\max_{e' \in E} c(e') > 0$, because scaling by a positive multiplicative factor does not change how two paths are ranked by their costs. Otherwise, no, because the order of comparison of two paths will be reversed.

4. (6 points) Consider an adversarial game tree where the root node is a maximizer, and the minimax value of the game is v_M. Now, also consider an otherwise identical tree where every minimizer node is replaced with a chance node (with an arbitrary but known probability distribution). The expectimax value of the modified game tree is v_E.

   (a) (2 points) v_M is guaranteed to be less than or equal to v_E.

   Solution: True. The maximizer is guaranteed to be able to achieve v_M if the minimizer acts optimally. He can potentially do better if the minimizer acts suboptimally (e.g. by acting randomly). The expectation at chances nodes can be no less than the minimum of the nodes' successors.

   (b) (2 points) Using the optimal *minimax* policy in the game corresponding to the modified (chance) game tree is guaranteed to result in a payoff of at least v_M.

Solution: True. The minimax policy is always guaranteed to achieve payoff of at least v_M, no matter what the minimizer does.

(c) (2 points) Using the optimal *minimax* policy in the game corresponding to the modified (chance) game tree is guaranteed to result in a payoff of at least v_E.

Solution: False. In order to achieve v_E in the modified (chance) game, the maximizer may need to change his or her strategy. The minimax strategy may avoid actions where the minimum value is less than the expectation. Moreover, even if the maximizer followed the expectimax strategy, he or she is only guaranteed v_E *in expectation*.

5. (2 points) In game theory, a correlated equilibrium is necessarily Pareto optimal.

Solution: No, see game of chicken.

6. (4 points) Value iteration is usually by initializing $V_0(s) = 0$ for all states s. In fact, this algorithm could be run with $V_0(s)$ initialized to any value and it would still converge to the optimal value function.

Solution: Yes. Let V_t be the sequence generated by value iteration with non-null V_0. Let V'_t be the sequence generated by value iteration with null V'_0. We know that V'_0 converge to the optimal value function. As $||V_t - V'_t||_\infty \leq \gamma^{t-1} \max_s |V_0(s)|$ converges to zero, V_t converges as well to the optimal value function.

7. (2 points) Consider a CSP with three variables: A, B, and C. Each of the three variables can take on one of two values: either 1 or 2. There are three constraints: $A \neq B$, $B \neq C$, and $A \neq C$. In the table below, enforcing *arc-consistency* would not cross off any values.

A	1	2
B	1	2
C	1	2

Solution: There is no solution to this CSP, but arc-consistency does not detect this. There is a consistent assignment in the head for each value of the tail and no values are eliminated.