COMP 3211: 2013-2014 Fall Semester Midterm Exam

Instructions

1. The exam time is 2 hours and 20 minutes, 12:00-14:20.
2. There are total 6 questions.
3. Write your answers in the exam paper only.

Student Name:

Student Number:

Marks

Q1:

Q2:

Q3:

Q4:

Q5:

Q6:

Total:

Problem 1 (22 pts) An artificial ant lives in a two-dimensional grid world and is able to follow a continuous pheromone trail (one cell wide) of marked cells. The ant occupies a single cell and faces either up, left, down, or right. It is capable of three actions, namely, move one cell ahead (m), turn to the left while remaining in the same cell (l), turn to the right while remaining in the same cell (r). The ant can sense whether or not there is a pheromone trace in the cell immediately ahead of it (in the direction it is facing). Specify a production system for controlling the ant such that it follows the trail. Assume that it starts in a cell where it can sense a pheromone trace. (Recall that a production system consists of an ordered set of condition-action rules; the action executed is the one that corresponds to the first satisfied condition. A rule might have more than one action on its right-hand side.) Make sure your ant doesn't turn around and retrace its steps backward!

(1.1) Show that if the ant does not have any memory, then this cannot be done.

(1.2) Now solve the problem by adding a state bit (x) to the ant's memory system: x is true iff the state bit is on. Assume that x is false initially. The agent can sense whether or not x is true, and it has two actions on x: *on* - make x true, and off - make x false.

Suggested Solution:
(1.1) show two states that are equivalent but need to take two different actions.
(1.2) We use the following notation: x1 is true iff there is a pheromone trace in the cell directly ahead of the ant. Here is the production system:

$$x1 \rightarrow m, off$$
$$x \rightarrow l$$
$$1 \rightarrow r, on$$

Marking Scheme:
(1.1) **8 pts**

- Simply said "two different actions are required for the same input configuration" or did not mention otherwise it would retrace its steps. -2
- No examples. -4

(1.2) **14 pts**

- Incorrect rule. -4/each

Problem 2 (18 pts)

(2.1) Prove that if a Boolean function $f(x_1, ..., x_n)$ can be implemented by a TLU, then its complement $\bar{f}(x_1, ..., x_n)$ can be implemented by a TLU as well. Here the complement $\bar{f}$ is: $\bar{f}(x_1, ..., x_n)$ returns true if and only if $f(x_1, ..., x_n)$ returns false, for any $x_1, ..., x_n$.

(2.2) Can the following Boolean function be implemented by a TLU? If so, give a TLU for it, if not, explain why not.

$$(\overline{x_1} + x_2)(x_1 + \overline{x_2}).$$

Suggested Solution:
(2.1) The simplest way to prove this is by linear separation. If $f(x_1, ..., x_n)$ can be implemented by a TLU, then it is linearly separable, meaning there is a $(n-1)$th

dimensional plane to separate its domain into two regions: those where f is true and those where f is false. Now the same plane also separates the domain for $\bar{f}$. Thus $\bar{f}$ is also linearly separable, so can be implemented by a TLU.

Alternatively, this can be proved by definition. Suppose f is implemented by a TLU with weight vector $\vec{w}$ and threshold θ. First of all, observe that we can modify θ if necessary so that for any input $\vec{x}$, $f(\vec{x}) = 1$ iff $\Sigma\vec{w}\vec{x} > \theta$. Now it is easy to see that the TLU with weight vector $-\vec{w}$ and threshold $-\theta$ will implement $\bar{f}$.

(2.2) It cannot be implemented by a TLU as its complement is the well-known exclusive-or.

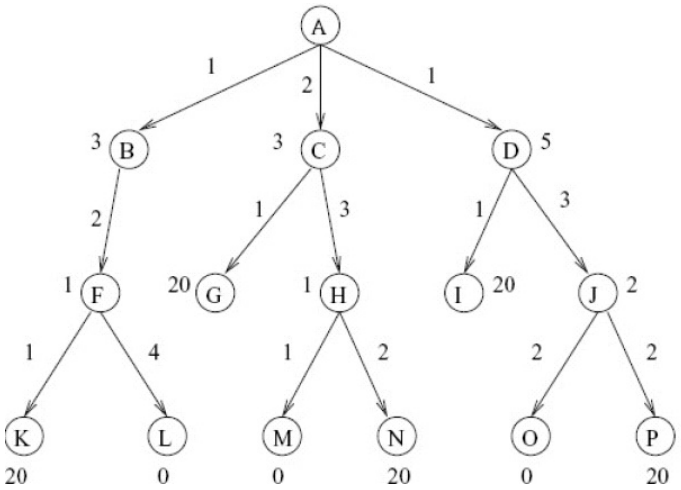
Marking Scheme:
(2.1) **10 pts**

- Simply used the opposite numbers of the weights and θ, need to handle the special case where $\sum w_i x_i = \theta$. -2

(2.2) **8 pts**

- No or wrong explanation. -6

Problem 3 (18 pts) Consider the following search tree. The number next to a node is the value of the heuristic function on the state of the node, and the number next to an arc is the cost of the corresponding operator. If the number next to a node is 0, that means that the node is a goal node, and if the number is 20, then it is a dead-end.



(3.1) Give the sequence of the nodes expanded by A^* by tree, starting from the root A and terminating at a goal node. Notice that whenever there is a tie, we prefer nodes at deeper levels, and on the same level, left to right.

(3.2) Can you adjust the heuristic function so that the goal node O is returned? Notice that your heuristic function still needs to be admissible, and you don't need to do this question if O is already the terminating goal node in your solution to (3.1).

Suggested Solution:
(3.1) A,B,F,C,H,M.

(3.2) Many possible solutions. An easy one is to change D's value to 1 and J's value to 1. Some people changed them to 0, which strictly speaking is not allowed as 0 means a solution node.

Marking Scheme:
(3.1) **10 pts**

- Gave the goal path as answer while the sequence of expansion was shown. -2
- Wrong sequence (e.g. expanded D and J). -10

(3.2) **8 pts**

- Inadmissible heuristic function or other goal node (e.g. M) was returned. -8

Problem 4 (16 pts) We said that for A^* to always return an optimal solution, it should only check if a node is a goal state when that node is chosen for expansion, not when a node is first generated. Make up an example to illustrate this, i.e. using your example to show that had we check if a node is a goal state when it is first generated, A^* would have returned a sub-optimal solution. You can use either A^* by tree or A^* by graph.

Suggested Solution:
Consider a search problem with 4 states: A,B,C,and D. A is the initial state, both B and D are goal states. Operator O1 maps A to B with cost 5, O2 maps A to C with cost 1, and O3 maps C to D with cost 1. Suppose h(A)=h(B)=h(C)=h(D)=0. Then A* expands nodes in the following order:

A $\rightarrow$ {B,C}
C $\rightarrow$ {D}
D $\rightarrow$ goal.

Thus D is returned. Had we done goal checks immediately when a node is generated, B would have been returned. But clearly B is sub-optiaml.

Marking Scheme:
Inadmissible heuristic function was used. -5

Problem 5 (16 pts) Consider a competition between Coke and Pepsi. Either of them can decide to raise its price by 10%, lower its price by 10%, or do nothing. If one raises its price by 10% while the other does nothing, then the one which is raising its price will lose 10% of its customer to the other company. However, if one lowers its price by 10% while the other does nothing, then the one which is lowering its price will gain 10% of the other company's customers. If both companies raise or lower their price by 10%, then there would be no change in their customers. If one raises its price by 10% while the other lowers its price by 10%, then the one which is lowering its price will gain 20% of the other's customers.

(5.1) Formalize this situation as a game between Coke and Pepsi. For simplicity, assume that their goal is to have as many customers as possible.

(5.2) Find the Nash equilibria of your game, if there is any.

(5.3) Is your game a zero-sum game?

Suggested Solution:
(5.1)

	Do Nothing	Raise	Lower
Do Nothing	0, 0	1,-1	-1,1
Raise	-1, 1	0, 0	-2,2
Lower	1, -1	2, -2	0,0

(5.2) One Nash equilibrium: (Lower,Lower).
(5.3) It is a zero-sum game.

Marking Scheme:
(5.1) **8 pts**

- Partially correct payoff table. -4
- Used the profit instead of the number of customers to calculate the utility functions (payoffs). -4

(5.2) **4 pts**

- Wrong Nash equilibrium/equilibria. -4

Problem 6 (10 pts) We know that for a zero-sum game, all Nash equilibria must have same payoff: if (x, y) and (x', y') are Nash equilibria of a zero-sum game, then $u_1(x, y) = u_1(x', y')$ and $u_2(x, y) = u_2(x', y')$. We say that a two person game is a constant-sum game if there exists a constant c such that for all possible profile (x, y), $u_1(x, y) + u_2(x, y) = c$. Is it also true that for a constant-sum game, all Nash equilibria must have same payoff? If your answer is yes, give a proof of this. You can use any known results in your proof. If your answer is no, then give a counter-example.

Suggested Solution:
It is true. For a constant-sum game (A, B, u_1, u_2) is equivalent to a zero-sum game: suppose the constant is c. Now define a new game (A, B, u_1', u_2') such that $u_1'(x, y) = u_1(x, y) - c$ and $u_2'(x, y) = u_2(x, y)$. It is clear that the new game is zero-sum, and that for any profile (x, y), it is a Nash equilibrium of the original game iff it is a Nash equilibrium of the new game.

Marking Scheme:
Answered "false". -10
Answered "true" but with no or wrong proof. -8
Partially correct proof. -2 $\sim$ -6