# CSU33061 – Artificial Intelligence Exam

*Daniel Whelan – 19335045*
*Exam No. – 71755*

Q1.

(a) True, The Search provided is a depth first search (DFS).

(b) False, If the graph contains a cycle of nodes (i.e. A -> B -> C -> A) where none of these nodes are the goal node, and has no additional paths out of the cycle (i.e. B -> D) then the graph will continue to loop indefinitely.

(c) True, a Breadth-First Search of a graph containing a finite number of arcs must terminate as each node is only examined once by BFS by nature.

(d) (i)
In all circumstances, it does follow that there exists a path from n to the goal node that is less than or equal the cost of any other path. For instance if there are three nodes, S A and G. The cost S -> A is 1 the cost A -> G is 1 and the cost S -> G is 5. Then there is a minimum cost path from S -> G through A with a cost of 2. If all the costs were equal to 1 then direct S -> G would also be a minimum costing path. If there was a fourth node B such that; S -> B = 1, B -> G = 1, S -> A = 1 and A -> G = 1, then there are two minimum cost paths from S -> G that are equal (cost = 2). Therefore, as above, it follows that with costs between hops, there exists a path from n to Goal that costs less than or equal to any other path.

(ii)
It does follow for h(n) <= n, A* will find the minimal costing path. If it is guaranteed that h(n) will always be less than n then A* will always find the shortest path from n to Goal. This is due to the fact that A* calculates the cost of the path by adding the cost of the path so far to the estimated cost to the goal (h(n)). If this h(n) value is less than or equal to the actual cost from n to the Goal then it will always find the shortest path as its calculated cost will be less than or equal to this actual cost. In essence, the cost of the path calculated is “optimistic”.

(e) (i)

Yes A* is admissible under the given arc cost and heuristic function. This is because the heuristic function calls the inbuilt prolog length/2 function that returns the length of a list. As the list being checked is the Node List that contains the remaining nodes to be searched and the cost of hopping from node to node is 1 (1+L/(L+1) always is 1), Then the h(n) that is returned is equal to the actual cost of travelling through all of the nodes. As h(n) = cost(n), A* is admissible.

(ii)

a, d, c, d, c, d, c (infinitely)

(iii)

(Assuming that reverse of KB implies that [a, d] now equals [d, a])
Firstly if still using the starting point q under the above logic, the algorithm would fail as there no longer exists a starting point q in the knowledge base.
(Assuming that reverse of KB implies changing the order of the KB so [c, d] is at the start).
If we were to reverse the of the KB as above then we would find that the A* algorithm would terminate with starting node q. This is due to the fact that q:- b, c is being evaluated before q :- a is, which prevents the algorithm from entering the infinite loop entered above (ii). Q would search node b first and find that node b is a fact, once this is discovered the algorithm would terminate having reached node b which then reaches the goal of an empty list (q -> b -> []).

Q2

(a) Raising the discount factor of an MDP will in turn raise the Value of the discounted future rewards to the token. It may also change the optimal policy of the MDP as a result of increasing the value of these discounted futures.

(b) (i)

True, as in an MDP actions taken in one state do not affect the rewards for actions taken in others, therefore we can maximise the policy state by state.

(ii)

True, as the max(Q(s, a)) is calculated by getting the maximum return available from the given (s, a) pair in any policy, therefore if the optimal policy is chosen then the maximum return from each (s, a) pairing is in this policy.

(iii)

It does follow that $r(s_1,a,s) >= r(s_2,a,s)$ as it would be unlikely that the policy would choose $s_1$ over $s_2$ if it had a smaller reward, regardless of the probability of one occurring over the other. It is likely that it is equal to or greater than $s_2$.

(iv)

It does not follow that with given conditions $r(s_1,a,s) > r(s_2,a,s)$ as this is not the only factor that is taken into account when calculating $V^{\pi}(s_i)$. It may follow that the $p(s_1,a,s)$ is greater than the $p(s_2,a,s)$, and hence the reward of $s_2$ has been made smaller as a result of it.

(v)

Yes it is possible that $\pi(s) = a'$ as firstly as time goes on the value of performing an action decreases due to the discount factor. So eventually after long periods of time the rewards and values they create will all begin to converge towards zero and at this time it is possible for the action a' to be more worthwhile than performing a. In an example given in class, it was seen that due to the discount factor and even with a large difference in the rewards of a and a', it became more worthwhile to perform a' as a was beginning to decline in value in the later stages due to a more evenly divided probability of a occurring (e.g. 0.6 and 0.4) compared to the large distance between the probability of a' occurring

(0.95 and 0.05). This affected the amount that was being discounted as the (s, a) pair that had a 95% chance of occurring had a high value as there had been a greater reward for a than on a' and a had occurred each time. In essence, it is possible for $\pi(s) = a'$.

(c)

- Both MDP and Q Learning make use of a discount factor in order to discount the values of rewards in the future.
- Both MDP and Q Learning have a defined set of actions (A) that can be performed at each state.
- Both contain a defined series of states (S) that can be explored.
- Both rely on maximising the return of the action that is performed in a given state.

(d)

True, as Q-Learning occurs it is important to decay the learning rate by a constant amount. This is due to the fact at the beginning of training, we are in an unknown space and hence it is important to make larger jumps to reach the end goal, but as training continues it is important that we begin to make smaller and smaller steps and by decaying the learning rate we can begin to exploit this knowledge gained by exploring early on due to a large amount of learning occurring. In essence, early on more learning about the space needs to occur than later when there is a more understandable knowledge of the space.

(e)

Yes this is a justified statement to make. Once every action has been tried at every state, and it is known that no additional actions can be applied at any state, the optimal policy has been formed and the learner knows what is the best route to take through each state in order to maximise the return that it gets by performing actions at all states. It would not make sense to attempt to learn more about the environment when all actions have been exhausted at every possible point in the defined environment. It is known that performing a at s will not lead to a new state and hence there is no need to continue learning about the return of each action in every state. In essence, if all actions have been exhausted at all states, then the policy is known and the maximum return has been received from the environment.