

# 16.410 Principles of Automated Reasoning and Decision Making

**Problem Set #10**

**Due: Session 25**

## Learning, Control and Adversaries

### Objectives

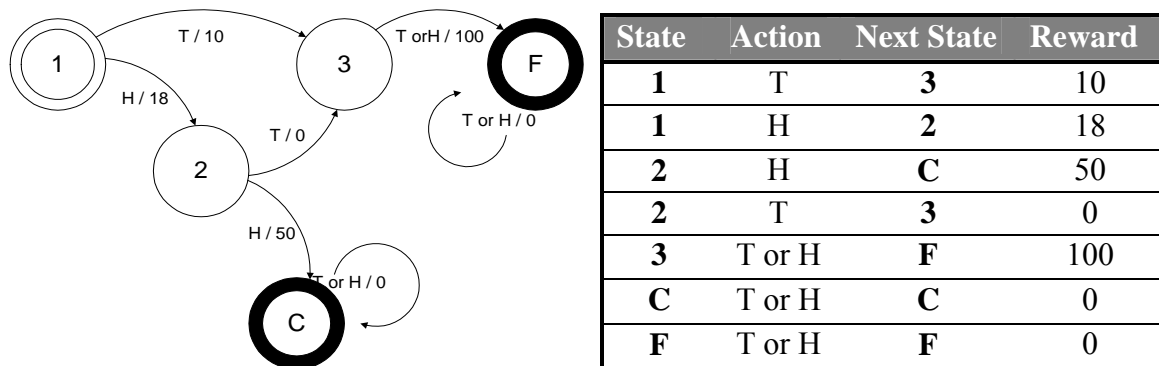
In this problem set you will develop your understanding of how agents act to maximize their utility, while interacting within a changing environment. The methods you will apply include decision tree learning for classification, Markov decision processes and reinforcement learning, and adversarial, game tree search.

### Readings

The material in this problem set corresponds primarily to Lectures 7, 23 and 24. Please review the corresponding lecture notes and any assigned readings, specified in the notes. Note that Lecture 7 covers game-tree search, Lecture 23 covers decision-tree learning, and Lecture 24 covers reinforcement learning and control, based on Markov Decision Processes.

### Problem 1 –MDPs: Tortoise and Hare

The following question is taken from last year’s final. We all know, as the story goes, that the Tortoise beat the Hare to the finish line. The Tortoise was slow, but extremely focused on the finish line, while the Hare was fast, but easily distracted. Although the Tortoise crossed the finish first, who really gained the greatest reward, the Tortoise or the Hare? It’s a matter of perspective. To resolve this age old question, we frame the race as an MDP, solve for the optimal policy, and use this policy to determine once and for all whose path is best, the Tortoise or the Hare.



We model the race with the above MDP. The race starts at 1, and finishes at F. 2 and 3

denote intermediate check points along the race course, while C denotes a Cabbage patch, which is very enticing to the Hare. Actions are T and H. T denotes actions focused towards the finish line, while H denotes an action that grabs the greatest immediate reward. The tortoise's sequence <T, T> is the shortest path to the finish line. The hare's sequence <H,H> is the direct path to the cabbage patch, with rewards along the way. <H, T, T> represents a mixed strategy, balancing immediate and long term reward.

### Part A. Value Function and Policy for Tortoise Discount

Use value iteration to determine the infinite horizon, discounted lifetime reward for states 1-3. For this part the Tortoise provides us with a discount factor, which is  $\gamma = 0.9$ . **Fill in the missing entries in the table below.** On the left,  $V^i$  denotes the value function at iteration i. **Stop** iterating as soon as  $V^i$  **converges**. Next **fill in** the actions for the **optimal policy on the right**.

State	$V^0$	$V^1$	$V^2$	$V^3$	$V^4$	$V^5$	$\pi^*$	State	Action
1	0							1	
2	0							2	
3	0							3	
F	0								
C	0								

Derive your answers below.

## Part B. Value Function and Policy for Hare Discount

The Hare complains that the model is unrealistic, “one should always live in the moment” he says. Instead he gives you a new discount factor of  $\gamma = 0.1$ . To give the Hare the benefit of the doubt, recompute  $V$  and  $\pi^*$  with this new  $\gamma$ . Fill in the following table, according to the same directions as Part A.

State	$V^0$	$V^1$	$V^2$	$V^3$	$V^4$	$V^5$	$\pi^*$	State	Action
1	0							1	
2	0							2	
3	0							3	
F	0								
C	0								

Derive your answers below.

### Part C. A $\gamma$ That Splits Hares

In order to get the Hare to head towards the finish line, we will need to perform the difficult task of training the Hare to discount the future less. What is the minimum value of the Hare's discount factor, so that the Hare heads to the finish line? That is, compute the greatest lower bound on  $\gamma$  such that the Hare does not head to the cabbage patch.

Answer:  $\underline{\quad} \leq \gamma$

Prove that your bound is correct:

## Problem 2 –Decision Tree Learning

You are using a rover on Mars to determine whether or not rocks you encounter are Martian in origin or are extramartian meteorites. You have collected eight samples, and used a human expert to determine whether or not the eight sample rocks are meteorites. You now want the rover to learn to classify meteorites by itself using the labelled examples.

Example	IsHeavy	IsShiny	IsSpotted	IsSmooth	IsMeteorite
A	0	0	0	0	0
B	1	1	1	1	0
C	1	1	1	1	0
D	1	0	0	1	1
E	0	1	1	0	1
F	0	0	1	1	1
G	1	1	0	1	1
H	1	1	0	0	1
U	1	1	1	1	?
V	0	1	0	1	?
W	1	1	0	0	?

You know whether or not rocks A through H are meteorites, but you do not know about U through W.

### Part A. Creating a Decision Tree

Build an ID-3 decision tree to classify rocks.

### Part B. Classification using a Decision Tree

Classify rocks U, V and W as meteorites or not using this decision tree.

## **Problem 3 – Exploration Within Reinforcement Learning**

### Part A. Random Exploration Strategy

Give an advantage and a disadvantage of exploring randomly.

### Part B. Greedy Exploration Strategy

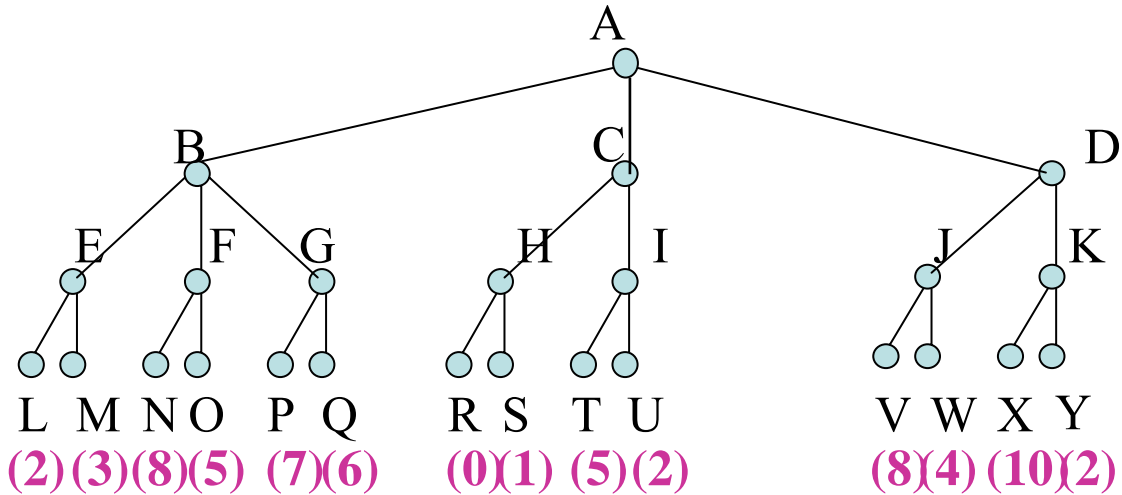
Give an advantage and a disadvantage of choosing your actions greedily, that is, choosing the action for which the current Q-table entry of your current state and chosen action is at a maximum.

### Part C. Mixed Exploration Strategy

A possible alternative for exploration that is a compromise between the random and greedy strategies is to choose your actions stochastically, but biased according to the current Q-values. Is this a good exploration strategy? Why or why not?

## Problem 4 – Game Tree Search

Consider the following game tree, in which the static scores (in parentheses at the leaf nodes) are all from the first player's point of view. Assume that the first player is the maximizing player.



### Part A. First Move

What move should the first player choose? Why?

### Part B. Alpha-Beta Pruning

What nodes would not need to be examined using the alpha-beta algorithm—assuming that nodes are examined in left-to-right order?

## Problem 5 A Ghost from The Past

During the mid-term we found that most of the class could use some additional practice at worst case complexity analysis. The following two questions are intended to provide this practice, in preparation for the final.

### Part A Complexity of Backtrack with Forward Checking

In this part we re-examine the complexity question from the mid-term.

**Derive the worst case run-time** of the **Backtrack Search with Forward Checking** algorithm, taught earlier in the course. Assume that there are  $v$  variables, each with a domain size of  $d$ , and there are  $e$  binary constraints. **Show and justify each step in your derivation.** Remember that you need to consider both the number of search nodes expanded, and the combined cost of performing forward checking on each of these nodes.

Begin by assuming that checking consistency of a pair of assignments against a single constraint is constant time, that is  $O(1)$ .

To help your analysis, work through the following steps:

### **Part A.1 Cost of the Forward Check Step**

Recall that a node in the search tree corresponds to a partial assignment to the variables in the CSP. What is the worst case run-time for performing arc consistency on a single node?

### **Part A.2 Number of Nodes Checked**

In the worst case, what is the maximum number of nodes in the search tree, for which the forward checking step is performed?

### **Part A.3 Cost of Backtrack Search with Forward Checking**

Taking in consideration your results from part A.1 and A.2, what is the worst case runtime complexity of Backtrack Search with Forward. Checking?

### **Part A.4 Improved Estimate For Backtrack Search with Forward Checking**

Consider any path through the search tree, from the root to one of its leaves. Notice that the combined set of forward checking tests performed on every variable along the path will perform forward checking on each arc (binary constraint) at most once.

Derive a complexity bound for performing forward checking along any such path.

Use this observation to improve the complexity bound that you computed in Part A.3.

### **Part A.5 A Better Bound on Testing Consistency**

Earlier we assumed that testing consistency of a constraint against a set of variables is constant time. This in general is not the case. Suppose we have a constraint over  $r$  variables, each with a maximum domain size of  $d$ . Furthermore, assume that the constraint is encoded as a set of tuples, each of which satisfies the constraint. What is the worst case run-time complexity for testing consistency of an assignment against that constraint?

Note that a straight forward algorithm to perform the test would check the assignment against every tuple in the constraint. This is, however, not particularly efficient. Think about how to index and search the set of tuples so that you do not need to examine all tuples in your test.

## **Part B Complexity of the Baum-Welch Algorithm**

Perform a similar analysis for the Baum-Welch algorithm. In particular, compute the worst time complexity for a single iteration of the Baum-Welch algorithm. Once again, it is important to break down the problem in steps. Here are some questions that you

should consider in your analysis. What parameters do you want to use to measure complexity of the problem (e.g., number of states, length of the training sequence, ...)? Recall that Baum-Welch consists of an Expectation step followed by a Maximization step. The expectation step involves applying the forward algorithm and the backward algorithm. What is the computational complexity of each algorithm? Maximization involves computing the new transition probabilities for each transition in the HMM. What is the complexity of this step? Finally, what is your combined result?

### **Problem 6: Time (5 pts)**

Please let us know the amount of time it took you to complete this problem set.