CS 495 & 540: Problem Set 2

Section: MW 10-11:50 am

Total: 100pts Due: 03/02/2016

Instructions:
1. I leave plenty of space on each page for your computation. If you need more sheet, please attach your work right behind the corresponding problem. If your answer is incorrect but you show the computation process, then partial credits will be given. Please staple your solution and use the space wisely.

2. This assignment contains two parts. Both are due on the March 2. You are allowed to work on the homework in a group of size up to two. No late assignment is accepted. Identical solutions (same wording, paragraph, code), turned in by different groups (persons), will be considered cheating.

3. Full credit will be given only to the correct solution which is described clearly. Convoluted and obtuse descriptions might receive low marks, even when they are correct. Also, aim for concise solutions, as it will save you time spent on write-ups, and also help you conceptualize the key idea of the problem.

First Name:

Last Name:

Group ID:

Score: /

Problem 1 Adversarial Search: 10pts

Prove the following assertion: For every game tree, the utility obtained by MAX using minimax decisions against a suboptimal MIN will be never be lower than the utility obtained playing against an optimal MIN. Can you come up with a game tree in which MAX can do still better using a suboptimal strategy against a suboptimal MIN?

Problem 2 Adversarial Search: 20pts

Consider the two-player game described in Figure 5.17 in the textbook.
(a) Draw complete game tree, using the following conventions:

- Write each state as (S_A, S_B), where S_A and S_B denote the token locations.
- Put each terminal state in a square box and write its game value in a circle
- Put *loopstates*(states that already appear on the path to the root) in double square boxes. Since their value is unclear, annotate each with a "?"' in a circle.

(b) Mark each node with its backed-up minimax value (also in a circle). Explain how you handled the "?" values and why.

(c) Explain why the standard minimax algorithm would fail on this game tree and briefly sketch how you would fix it, drawing on your answer to (b). Does your modified algorithm give optimal decisions for all games with loops?

(d) This 4-square game can be generalized to n squares for any $n > 2$. Prove that A wins if n is even and loses if n is odd.

Problem 3 Adversarial Search: tic-tac-toe: 20pts

Consider the family of generalized tic-tac-toe games, defined as follows. Each particular game is specified by a set S of squares and a collection W of winning positions. Each winning position is a subset of S For example, in standard tic-tac-toe, S is a set of 9 squares and W is a collection of 8 subsets of W: the three rows, the three columns, and the two diagonals. In other respects, the game is identical to standard tic-tac-toe. Starting from an empty board, players alternate placing their marks on an empty square. A player who marks every square in a winning position wins the game. It is a tie if all squares are marked and neither player has won.

a. Let $N = |S|$, the number of squares. Give an upper bound on the number of nodes in the complete game tree for generalized tic-tac-toe as a function of N.

b. Give a lower bound on the size of the game tree for the worst case, where $W = \{\}$.

c. Propose a plausible evaluation function that can be used for any instance of generalized tic-tac-toe. The function may depend on S and W.

d. Assume that it is possible to generate a new board and check whether it is a winning position in $100N$ machine instructions and assume a 2 gigahertz processor. Ignore memory limitations. Using your estimate in (a), roughly how large a game tree can be completely solved by alphabeta in a second of CPU time? a minute? an hour?

Problem 4 ExpectiMax: 20 pts

This question considers pruning in games with chance nodes. Figure 5.19 shows the complete game tree for a trivial game. Assume that the leaf nodes are to be evaluated in left- to-right order, and that before a leaf node is evaluated, we know nothing about its valuethe range of possible values is $-\infty$ to ∞.
a. Copy the figure, mark the value of all the internal nodes, and indicate the best move at the root with an arrow.

b. Given the values of the first six leaves, do we need to evaluate the seventh and eighth leaves? Given the values of the first seven leaves, do we need to evaluate the eighth leaf? Explain your answers.

c. Suppose the leaf node values are known to lie between 2 and 2 inclusive. After the first two leaves are evaluated, what is the value range for the left-hand chance node?

d. Circle all the leaves that need not be evaluated under the assumption in (c).

Problem 5 CSP: Coloring: 5+3+2pts

Consider the Figure 6.1 in the textbook.
(a) How many solutions are there for the map-coloring problem, suppose 3 colors are allowed?

(b) How many solutions if four colors are allowed?

(c) How many solutions if two colors are allowed?

Problem 6 CSP: Most Constrained Variable 10pts

Explain why it is a good heuristic to choose the variable that is most constrained but the value that is least constraining in a CSP search.

Problem 7 CSP: K-Consistency 10pts

Use the AC-3 algorithm to show that arc consistency can detect the inconsistency of the partial assignment $\{WA = green, V = red\}$ for the problem shown in Figure 6.1.