Assignment 2

Search and Genetic Algorithms

Deadline: March 24, 2020
Score: 100 points

Assignment Instructions:

Teams: Assignments should be completed by groups of up to 4 students. No additional credit will be given for students working individually. You are strongly encouraged to form a team. Please inform the TAs as soon as possible about the members of your team so they can update the scoring spreadsheet.

Submission Rules: Submit your reports electronically as a PDF document through Sakai (sakai.rutgers.edu). For programming questions, you need to also submit a compressed file via Sakai, which contains your results and/or code as requested by the assignment. For your reports, do not submit Word documents, raw text, or hardcopies etc. Make sure to generate and submit a PDF instead. Each team of students should submit only a single copy of their solutions and indicate all team members on their submission. Failure to follow these rules will result in a lower grade in the assignment.

Program Demonstrations: There will be no program demonstrations for this assignment.

Late Submissions: No late submissions are allowed. You will be awarded 0 points for late assignments!

Precision: Try to be precise in your description and thorough in your evaluation. Have in mind that you are trying to convince a very skeptical reader (and computer scientists are the worst kind...) that your answers are correct.

Collusion, Plagiarism, etc.: Each team must prepare its solutions independently from other teams, i.e., without using common code, notes or worksheets with other students or trying to solve problems in collaboration with other teams. You must indicate any external sources you have used in the preparation of your solution. Do not plagiarize online sources and in general make sure you do not violate any of the academic standards of the course, the department or the university (the standards are available as part of the course syllabus). Failure to follow these rules may result in failure in the course.

Assignment Description:

Problem 1: [35 points] Two players, the MAX and the MIN are playing a game where there are only two possible actions, “left” and “right”. The search tree for the game is shown in Figure 1. The leaf/terminals nodes, which are all at depth 4, contain the evaluation function at the corresponding state of the game. The MAX player is trying to maximize this evaluation function, while the MIN player is trying to minimize this evaluation function.

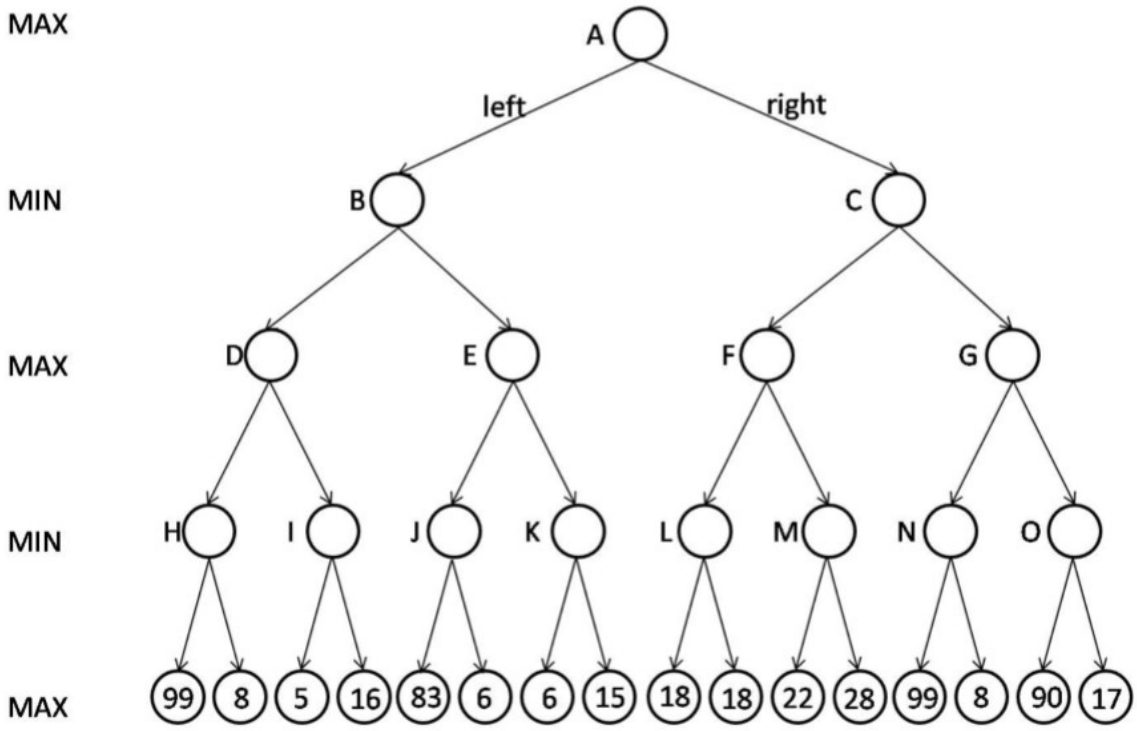

Figure 1: The adversarial search tree.

a) Consider the operation of the Minimax algorithm over the above search tree and indicate the value of each intermediate node.

b) Consider the operation of the Minimax algorithm with alpha-beta pruning over the above search tree. Indicate which nodes will not be consider by this algorithm (you can reuse the figure and mark them visibly or mention which intermediate and terminal nodes are not visited in text). Furthermore, indicate the alpha and beta values on each intermediate node that is visited.

c) What action will the MAX player choose at the root state according to the exhaustive Minimax algorithm? What action will the MAX player choose at the root state according to the Minimax algorithm employing alpha-beta pruning? In general, is the best move computed by the two versions of the algorithm guaranteed to be the same or not?

d) Consider a version of the algorithm that heuristically evaluates the quality of nodes at depth 2 so as to guide the ordering of nodes considered during the alpha-beta pruning.

   In particular, assume that the heuristic value at the depth 2 nodes are: h(D) = 9, h(E) = 7, h(F) = 24 and h(G) = 16, and Minimax backs-up these values at nodes B and C as well.

   Now, the MAX player executes the alpha-beta pruning algorithm down to

depth 4. This time, however, it uses the heuristic values at nodes B through G to reorder the nodes of the depth-4 game tree at depths 1 and 2. The objective is to maximize the number of nodes that will not be examined by the alpha-beta pruning process.

How will the alpha-beta pruning algorithm reorder the nodes, given the information generated up to depth 2? Show the corresponding new tree considered by the alpha-beta pruning algorithm using the labels of the nodes from the above figure. In addition, inside each leaf node give the value of the evaluation function.

How many nodes will not be examined by the alpha-beta pruning algorithm in this case?

e) Consider now a variation of the game, where the opponent is known to play randomly, i.e., instead of trying to minimize the evaluation function, it selects with 50% the left action and with 50% the right action. Provide the value of each node on the original tree in this case. What will be the choice of the MAX player in this case at the root state?

Can you apply alpha-beta pruning in this case? If yes, show how. If not, explain why.

Problem 2: [25 points] Consider the game Sudoku, where we try to fill a 9×9 grid of squares with numbers subject to some constraints: every row must contain all of the digits 1,...,9, every column must contain all of the digits 1,...,9, and each of the 9 different 3×3 boxes must also contain all of the digits 1,...,9. In addition, some of the boxes are filled with numbers already, indicating that the solution to the problem must contain those assignments. Here is a sample board:

	1		4	2				5
		2		7	1		3	9
							4	
2		7	1					6
				4				
6					7	4		3
	7							
1	2		7	3		5		
3				8	2		7	

Figure 2: Sample Sudoku board

Each game is guaranteed to have a single solution. That is, there is only one assignment to the empty squares which satisfies all the constraints. For the purposes of this homework, let's use $n_{i,j}$ to refer to the number in row i, column j of the grid. Also, assume that M of the numbers have been specified in the starting problem, where M = 29 for the problem shown above.

a) This is an instance of a Constraint Satisfaction Problem. What is the set of variables, and what is the domain of possible values for each? How do the constrains look like?

b) One way to approach the problem, is through an incremental formulation approach and apply backtracking search. Formalize this problem using an incremental formulation. What are the start state, successor function, goal test, and path cost function?

   Which heuristic for backtracking search would you expect to work better for this problem, the degree heuristic, or the minimum remaining values heuristic and why?

   What is the branching factor, solution depth, and maximum depth of the search space? What is the size of the state space?

c) What, is the difference between "easy" and "hard" Sudoku problems? [Hint: There are heuristics which for easy problems will allow to quickly walk right to the solution with almost no backtracking.]

d) Another technique that might work well in solving the Sudoku game is local search. Please design a local search algorithm that is likely to solve Sudoku quickly, and write it in pseudocode. You may want to look at the WalkSAT algorithm for inspiration. Do you think it will work better or worse than the best incremental search algorithm on easy problems? On hard problems? Why?

Problem 3: [20 points] Consider the following sequence of statements, which relate to Batman's perception of Superman as a potential threat to humanity and his decision to fight against him.

For Superman to be defeated, it has to be that he is facing an opponent alone and his opponent is carrying Kryptonite. Acquiring Kryptonite, however, means that Batman has to coordinate with Lex Luthor and acquire it from him. If, however, Batman coordinates with Lex Luthor, this upsets Wonder Woman, who will intervene and fight on the side of Superman.

a) Convert the above statements into a knowledge base using the symbols of propositional logic.

b) Transform your knowledge base into 3-CNF.

c) Using your knowledge base, prove that Batman cannot defeat Superman through an application of the resolution inference rule (this is the required methodology for the proof).

Problem 4: [20 points] Disjunctive logical sentences are of the form: $(l_1 \vee l_2 \vee \cdots l_n)$, where l_i is a literal or clauses, and sentences that have the form $(\alpha \Rightarrow \beta)$ (i.e., implication sentences).

a) Can you show the logical equivalence of $(\neg P_1 \vee \cdots \vee \neg P_m \vee Q)$ and $(P_1 \wedge \cdots \wedge P_m) \Rightarrow Q$?

b) Show that regardless of the number of positive literals in a clause, such expressions can be written in the form $(P_1 \wedge \cdots \wedge P_m) \Rightarrow (Q_1 \vee \cdots \vee Q_n)$, where the Ps and Qs are proposition symbols.

c) Write down the full resolution rule using implications. As a reminder the full resolution rule looks as follows

- Given clause $(l_1 \vee \ldots \vee l_k)$
- and clause $(m_1 \vee \ldots \vee m_n)$
- we can infer that:

$$(l_1 \vee \ldots l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n)$$

where l_i and m_j are complementary literals.