EECS 592: AI Foundations Homework 1

Assigned date: August 31, 2022
Due date: September 23, 2022

Instructions:

- Please submit all the materials including written answers and code into one .zip file to Canvas before the due date. Please name it as HW1_<uniqname>.zip, where <uniqname> should be replaced by your unique name.
- For the written answers, please include them in one PDF file, named as HW1_<uniqname>.pdf. You can write it in any format (LaTeX, Word, written), but it should be ultimately transformed into one PDF.
- For the questions that involve programming, create a folder for the corresponding questions, e.g., Q5/, and put all scripts under this directory. In each folder, also include a README.txt which gives details and explanations about the corresponding functions that you are asked to implement, the input parameters, and the output formats.
- Failure to comply with the above requirements will lead to deduction of up to 5 points per aspect.

1 Rational Agent

1.1 True of False [8 Points]

Decide True or False for the following statements, and provide a brief justification using 1-2 sentences.

(a) Simple reflex agents can act rationally in some scenarios.

(b) A rational chess-playing agent never loses.

(c) A perfectly rational poker-playing agent never loses.

(d) An agent that is considered to be rational in one task environment may be irrational if the performance measure changes.

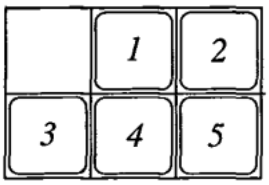


Fig. 1: Five puzzle.

(e) Once we write a agent program that properly implements the agent function for the agent, it can work as expected in the environment.

(f) A rational agent can be omniscient if the task environment is deterministic.

(g) Agent function can change as the agent gains experience from the environment.

(h) The learning ability of rational agents ensures that it would only get better.

1.2 Task Environment (PEAS) [5 Points]

Use the PEAS description to specify the task environment of **buying a wireless keyboard on the Internet**. And determine whether the environments are (1) fully or partial observable, (2) single or multi-agent, (3) episodic or sequential (4) static or dynamic, and (5) discrete or continuous.

2 Problem Formulation [20 Points]

Tab. 1: Price of pieces

length	1	2	3	4	5	6
price	3	5	6	8	9	10

Formulate the following problems using (1) state representation, (2) initial state, (3) goal state, (4) the total number of reachable state, (5) action representation, (6) action costs, (7) transition model, (8) the range of branching factor. The formulations are expected to be the ones with the minimal number of states (in the cases where different formulations are possible).

Notices: To find and report the answers of (4) or (8), you should show your derivations.

(a) [10 points] Cutting a rod. Given a rod of length 6 inches and prices of all pieces of size smaller than 6 (shown in Table 1) The cost of each cut is 2. Determine the maximum net profit by cutting up the rod and selling the pieces.

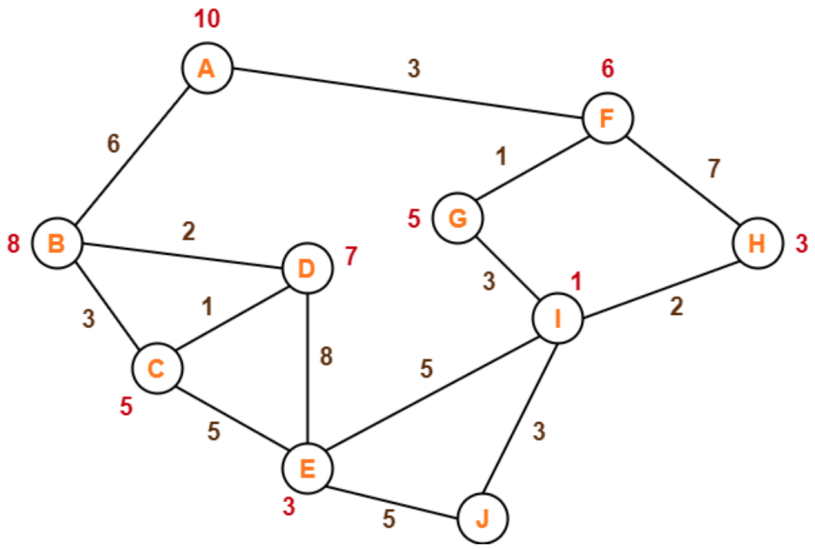


Fig. 2: Start node: A; Final node: J

(b) [10 points] Five puzzle. It is 2×3 version of the well-known sliding-tile puzzles. There are five numbered square tiles, and one empty position, called the "blank". Any tile horizontally or vertically adjacent to the blank can be slid into the blank position. The goal is to rearrange the tiles from some random initial configuration into a particular goal configuration shown in Figure 1.

3 Search Procedure [17 Points]

Consider the graph shown in Figure 2. Our goal is to start from node A to reach node J. The edge costs and heuristic values of points are labeled correspondingly in the figure.

Your task is to provide the search procedures of the following algorithms. Specifically, you are asked to provide in each iteration the sets of reached points, the frontier, (ordered if implemented by a priority queue) the node that get expanded in this iteration, and the successors.

(a) Breadth-first search, the version in Figure 3.9 of the textbook.

(b) Depth-first search, implemented as best-first-search in Figure 3.7 of the textbook, with evaluation f being the negative of depth.

(c) A^*. Also decide is h admissiable?

Notes:

- For tie-breaking, assume the algorithms process the nodes in an alphabet order with regard to their state label.
- You can represent a node by its state label for simplification. For example, the first three iterations of BFS should be reported as:

reached	frontier	to expand	successor
A	A	A	BF
ABF	BF	B	ACD
ABCDF	CDF	F	AGH

4 Heuristics [10 Points]

4.1 [6 Points]

Many games and web-based maps use A^* search algorithm to find the shortest path very efficiently. And A^*'s behavior can be controlled by the heuristics.

(a) In some situations, we would rather have a "good" path than a "perfect" path. How would you adjust the $h(n)$ in this case? Explain.

(b) A^*'s ability to vary its behavior based on the heuristic and cost functions can be exploited to make your game faster. Assume that there are two types of terrain, Flat and Mountain in your game, and the movement costs are 1 for flat land and 4 for mountains. How can we speed up A^* by setting values? How can we redesign the cost function to make the tradeoff between speed and accuracy dynamic? Please give a brief explanation.

4.2 [4 Points]

Consider a best-first search in which the evaluation function is

$$f(n) = 8g(n) + w(h(n) - g(n))$$

What kind of search does this perform for $w = 0$ and $w = 8$?

For what w values can the search find the optimal path, assuming that h is admissible?

5 Local Search [20 Points]

In this problem we are going to investigate the local search methods on the 8-queen problems. Suppose each queen is assigned to a particular column and constrained to move along that column alone. Each state is represented by 8 digits, with the c-th digit representing the row number of the queen in column c. The columns are counted left to right starting at 0 for the leftmost column and the rows are counted top to bottom, with the value zero for the topmost row. We give a utility function h that computes the number of conflict pairs in **utils.py**.

5.1 [8 Points]

Implement (steepest ascent) hill climbing. At each step of the search you are allowed to move a single queen to another (different) square in the same column. Break ties by choosing the move in the column farthest to the left. Given a tie within that column, choose the move closest to the top of the board. Given initial state [1, 0, 0, 3, 4, 5, 1, 6], run your algorithm.
Answer:

(a) What is the final state after the algorithm terminates?

(b) What is the cost of the initial state?

(c) What is the cost of the final state?

(d) What is the number of steps taken (in the environment) to reach that final state?

5.2 [6 Points]

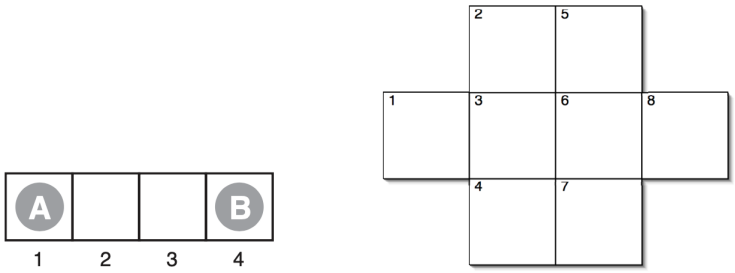
Using the steepest-ascent algorithm from above, implement the random-restart steepest ascent hill climbing algorithm. For each (re)start, you should choose a new, random initial state and begin steepest-ascent with this new initial state. This process should continue in a loop until a solution (no attacking queens) is found, or until some limit on the number of allowed (re)starts is reached. In your implementation, set the limit on allowed (re)starts to 100. Run the random-restart algorithm 1000 times and report **(a).** the empirical estimate of the expected number of restarts required to find a solution and **(b).** the empirical estimate of the expected total number of steps across restarts (sum across restarts) required to find a solution.

5.3 [6 Points]

Implement Genetic algorithm. For the algorithm, please use (1) population size 4, (2) each time select a state as a parent with probability in proportion to the number of pairs that **do not conflict**, (3) chose a random cross-over point uniformly distributed among all 7 possibilities, and (4) a mutation rate 0.15. The initial population are all generated from a uniform distribution. And let the algorithm terminate at iteration 100. During one run of the algorithm, you have 4 states in your population with corresponding 4 h values in each time step. And for each run of the algorithm, you should record a minimum h value encountered across all 400 states in the trajectory. Run the algorithm for 1000 times, and report the empirical average of this minimum h across all 1000 runs.

6 Adversarial Search [10 Points]

Consider a two-player game: The starting position of this game is shown in Fig. 3i. Player A moves first. The two players take turns moving, and each



(i) Starting position of the game. (ii) boxes

Fig. 3

player must move his token to an open adjacent space in either direction. If the opponent occupies an adjacent space, then a player may jump over the opponent to the next open space if any. (For example, if A is on 3 and B is on 2, then A may move back to 1.) The game ends when one player reaches the opposite end of the board. If player A reaches space 4 first, then the value of the game to A is +1; if player B reaches space 1 first, then the value of the game to A is −1.

(a) Draw the complete game tree, using the following conventions:

1. Write each state as $(\mathcal{S}_\mathcal{A}, \mathcal{S}_\mathcal{B})$, where $\mathcal{S}_\mathcal{A}$ and $\mathcal{S}_\mathcal{B}$ denote the token locations.

2. Put each terminal state in a square box and write its game value in a circle.

3. Put loop states (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) Explain why the standard minimax algorithm would fail on this game tree.

7 Constraint Search [10 Points]

Fill cells with the numbers 1 to 8 such that the labels of any pair of adjacent cells (i.e., horizontal, vertical, or diagonal) differ by at least 2. Please refer to Fig. 3ii.

(a) Write the constraints in a relational form (i.e., relation and domain) and draw the constraint graph.

(b) Is the network arc-consistent ? If not, compute the arc-consistent network.

(c) Is the network consistent ? If yes, give a solution.