Introduction to Intelligent Systems: Homework 2

Alvin Lin - Section 1

August 2017 - December 2017

Problem 1

For each of the following, gives a PEAS description of the task and given solver of the tasks. There may be several reasonable answers, but the key is that all four parts of your answer go together.

(a) Robots play soccer. The task environment is (usually) fully observable via sensors on the robot (though from personal experience they're usually notoriously unreliable). By the nature of the game, the environment is multi agent, stochastic, sequential, dynamic, and continuous. The rules of the game are set, so the robots operate in a known environment.

Agent Type	Performance Measure	Environment	Actuators	Sensors
Soccer robot	Goals scored, ball possession time, acquisition speed	Soccer field, other robots, soccer ball	Motors for steering, movement, and kicking	Infrared (usually for the ball), motor encoders, direction sensors, ultrasonic distance sensors

(b) Netflix/Amazon on-line recommendation system. The task environment is partially observable, so it is necessary to keep state information about a user's previous choices. The recommendation system operates as a single agent. Because it interacts with a human as a feedback mechanism, the environment is uncertain, episodic, static, and discrete. Human preferences can vary according to other factors, so the environment is unknown.

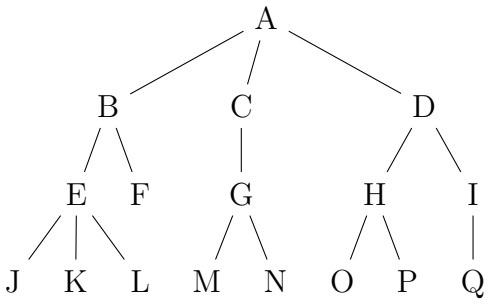
Agent Type	Performance Measure	Environment	Actuators	Sensors
Recommendation System	User engagement, retention, and click-through	The collection of possible shows a user can watch	Interface where suggestions are displayed	User clicks, mouse movements, and engagement

(c) Expert system for medical diagnosis. The task environment is partially observable since it is not feasible to have all the data on a person at any given time. The system is single agent and interacts with a stochastic, sequential, dynamic, continuous, and unknown environment.

Agent Type	Performance Measure	Environment	Actuators	Sensors
Recommendation System	Diagnosis accuracy and patient resolution accuracy	The symptoms, medical history, and predispositions of a patient	Interface where diagnoses and suggestions are displayed	Database of patient data, x-rays machines, MRI scans and other tests performed on the patient

Problem 2

Show the execution of the following search techniques on the tree shown below. A is the root node and M is the goal state. For BFS, and DFS, keep track of the open and closed arrays at each step of the search. If a node has more than one child, add them to the open list in the left-to-right order as they as shown in the tree (not alphabetical search). For IDS, give a list of the states that are examined for each limit (starting with limit 0), in the order that the states are examined.



(a) Breadth-First Search

open	closed
A	
B,C,D	A
C,D,E,F	B,A
D,E,F,G	C,B,A
E,F,G,H,I	D,C,B,A
F,G,H,I,J,K,L	E,D,C,B,A
G,H,I,J,K,L	F,E,D,C,B,A
M is a child of G	G,F,E,D,C,B,A

(b) Depth-First Search

open	closed
A	
B,C,D	A
E,F,C,D	B,A
J,K,L,F,C,D	E,B,A
K,L,F,C,D	J,E,B,A
L,F,C,D	K,J,E,B,A
F,C,D	L,K,J,E,B,A
C,D	F,L,K,J,E,B,A
G,D	C,F,L,K,J,E,B,A
M is a child of G	G,C,F,L,K,J,E,B,A

(c) Iterative Deepening

limit	searched
0	A
1	A,B,C,D
2	A,B,E,F,C,G,D,H,I
3	A,B,E,J,K,L,F,C,G,M

Problem 3

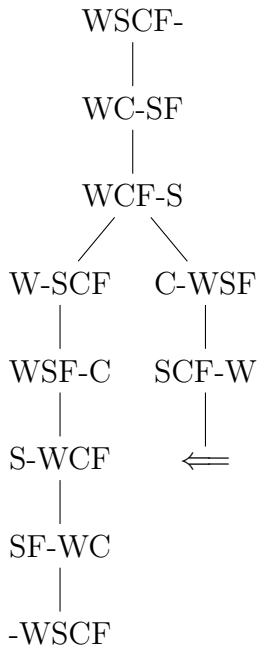
There is a famous problem similar to the missionaries and cannibals from the notes: A farmer has to get a fox, a chicken, and a sack of corn across a river. The farmer has a rowboat that can only carry one thing at a time (in addition to the farmer). If the fox and the chicken are left together, the fox will eat the chicken. If the chicken and the corn are left together, the chicken will eat the corn. How does the farmer get everyone across the river?

(a) Give a complete problem formulation for this problem. Choose a formulation that is precise enough to be implemented. Formulate a scheme for the problem similar to the one from the notes, including an initial state, goal state, and solution sequence.

To represent the problem, I will use a version more familiar to me involving a wolf instead of a fox, sheep instead of a chicken, and cabbage instead of corn (this allows me to abbreviate the wolf as W, the sheep as S, the cabbage as C, and the farmer as F, to avoid naming conflicts). The two sides of the river will be separated with a horizontal dash (-). The start state is represented with WSCF-, and the end state is represented with -WSCF. Since the wolf should not be left alone with the sheep, and the sheep should not be left alone with the cabbage, the following states are examples of forbidden states: WS-CF, SC-WF, CF-WS. Solution:

step	state	instruction
0	`WSCF-`	start state
1	`WC-SF`	farmer brings sheep
2	`WCF-S`	farmer returns alone
3	`W-SCF`	farmer brings cabbage
4	`WSF-C`	farmer returns with sheep
5	`S-WCF`	farmer brings wolf
6	`SF-WC`	farmer returns alone
7	`-WSCF`	farmer brings sheep

(b) Show the entire search tree for the farmer problem. Remove illegal states, and duplicate states.



Problem 4

For each of the following assertions, say whether it is true or false and support your answer with examples or counterexamples where appropriate.

(a) An agent that senses only partial information about the state cannot be perfectly rational. False, rational agents can operate in a partially observable environment. Autonomous cars can be rational agents by only acting on the information they can obtain from their sensors.

(b) There exist task environments in which no pure reflex agent can behave rationally. True, reflex agents act based on rules and the current state of the world. They would not be able to perform rationally for environments where state memory is required, such as Netflix recommendation.

(c) There exists a task environment in which every agent is rational. True, in the simplest possible state where only one action (or inaction) is possible, every agent would be rational.

(d) The input to an agent program is the same as the input to the agent function. Sometimes false, it depends whether or not the input function needs to perform preprocessing on the input or combine it with state information.

(e) Every agent function is implementable by some program/machine combination. False, it is possible for some environment to require infinite memory if each state requires memory and the rational agent is run forever. This cannot be overcome by any heuristic.

(f) Every agent is rational in an unobservable environment. False, in an observable environment, it is impossible for some agents to maximize performance without any data from which to make a decision.

(g) A perfectly rational poker-playing agent never loses. False, someone has to lose if a perfectly rational agent plays another perfectly rational agent. There are situations in poker where a loss will result no matter what decision is taken.

Problem 5

Write pseudocode agent programs for the following agents:

(a) goal-based agent

```
def action(agent, environment):
  let state = agent.sense(environment)
  for action in agent.availableActions:
    let resultingState = agent.do(action, state)
    if resultingState == agent.goal:
      return action
```

(b) utility-based agent

```
def action(agent, environment):
  let state = agent.sense(environment)
  let maxUtility = -Infinity
  let maxUtilityState = None
  for action in agent.availableActions:
    let resultingState = agent.do(action, state)
    let resultingStateUtility = agent.getUtility(resultingState)
    if resultingStateUtility > maxUtility:
```

```
        maxUtility = resultingStateUtility
        maxUtilityState = resultingState
    return maxUtilityState
```

Problem 6

Give a complete problem formation for each of the following. Choose a formulation that is precise enough to be implemented (this includes: initial state, goal test, actions, path cost, and potentially a solution if a problem is specific enough to be solved).

1. Using only four colors, you have to color a map in such a way that no two adjacent regions have the same color.

   The initial state can be created by picking a random color for a random section. It does not matter what color is selected for the initial state because if a solution exists, then the starting section can be of any color and a solution with exist for it. After that, a search tree can be created where the branches are possible colors for the adjacent sections. Each branch of the tree would represent a possible action from that point. In our search tree, we can prune out all invalid branches where adjacent colors are the same. In a map with n sections, this search tree with have a height of n with a maximum branching factor on any level equal to the maximum number of adjacent sections to any given section on the map. We can use a depth first search to search through this tree (though it will take some time and is not very efficient). A goal test would check if all regions are colored such that no two adjacent regions are the same colors.

2. A 3-foot-tall monkey is in a room where some bananas are suspended from the 8-foot feiling. The monkey would like to get the bananas. The room contains two stackable, movable, climbable 3-foot-high-crates.

   The initial state would be the problem as described. Actions that the monkey can do are: move itself, move the crates, stack the crates, climb the creates, and acquire the bananas (given the prerequisite state that the monkey is at a height where it can reach the bananas). A goal test would be checking if the monkey is currently at a height where it can reach the bananas, while the current height of the monkey relative to the ground could be used as a metric of how close it is to the goal. The number of actions taken by the monkey could be used as a path cost. If the monkey stacks the two crates and climbs on top of them, then it has reached the goal state.

3. You have two jugs, measuring 5 gallons and 3 gallons, and a water faucet. You can fill the jugs up or empty them out (completely) from one to another or onto the ground. You need to end up with exactly 4 gallons in the larger jug.

   The initial state would have both jugs empty. Possible actions are: fill the large jug from the faucet, fill the small jug from the faucet, empty the large jug onto the ground, empty the small jug onto the ground, pour the large jug into the small jug, or pour the small jug into the large jug. A goal test would check if the large jug is currently holding 4 gallons, while path cost could be determined by the number of actions taken. The problem could be solved as follows:

Small Jug	Large Jug	Action
0/3	0/5	Initial State
0/3	5/5	Fill the large jug from the faucet
3/3	2/5	Pour the large jug into the small jug
0/3	2/5	Empty the small jug
2/3	0/5	Pour the large jug into the small jug
2/3	5/5	Fill the large jug from the faucet
3/3	4/5	Pour the large jug into the small jug

This solution can also be determined by using a search tree.

4. You are able to pick from coins in three denominations: 3 cent, 7 cent, and 12 cent. What is the largest amount that you *cannot* represent with these coins?

   (Frobenius coin problem - NP hard). An initial state for this problem would be having no coins. Each step, you can pick one of the three denominations to add to your current amount as an action. For a goal test, one would check if the current amount is equal to the desired amount. Path cost would be determined by the number of steps taken where a coin is chosen during each step. This problem must be brute forced, so a search tree could be created from the initial state to simulate every possible combination of 1 coin, 2 coins, 3 coins, and so forth, with the n^{th} level of the tree representing any combination of n coins. This tree would be searched downward for the largest integer not included in any of its branches that was unable to be produced by a further branch.

If you have any questions, comments, or concerns, please contact me at alvin@omgimanerd.tech