Nadya Postolaki

1)[15 points] Answer the following questions briefly but precisely:

Can an agent that keeps no history of its percept sequence be rational? Please explain.

-For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure , given the evidence provided by the percept sequence and whatever built in knowledge the agent has.

- An agent's choice of action at any given instant can depend on the entire percept sequence observed to date, but not on anything it hasn't.

- Above definitions found in the textbook.

- An agent doesn't necessarily need to depend solely on the entire percept sequence observed to date, but it CAN depend on it, this is because it depends on what circumstances the agent is under. Take for example a simple calculator; it doesn't need to remember what the previous outputs it had provided if it just takes a function and some variable(s) inputs provided by the environment (person using the calculator) to compute the function. It doesn't care what two plus two equaled last time, but will compute it correctly everytime because all it needed was the function and two variables to be presented and any built in information in order for it to compute. The calculator (agent) is therefore rational without using percept squence (or lack thereof) because it produced the correct solution to the probelem at hand (4).

In a circumstance where an agent would need its percept sequence, for example, a vacuum needing to check whether it should move on to a new space once the current space is clean, at that point the agent would not be rational if the vacuum needs to also be energy efficient and each movement costs energy and so to prevent from going over spaces that already had been cleaned, the vacuum would have to remember where it was and the states of those locations.

So an agents' rationality doesn't always depend on its percept sequence- it depends on the circumstances it is in and what purpose the agent serves and what problems it is to face (performance measure).

Why performance measures for an agent should be designed according to the effects in the environment instead of the behaviors of the agent?

If success is defined in terms of an agent's opinion of its own performance, an agent could achieve perfect rationality simply by deluding itself that its performance was perfect. (Found on page 37). For example, human agents are often found believing they did not really want something after not getting it.

Why it is important to know if an agent's environment is fully observable or if it is partially observable?

For an agent to be able to make rational decisions (and therefor be a rational agent), it needs the ability to observe its environment and gather as much data as possible before coming to a conclusion as it bases its decisions on the environment around it. If an environment is fully observable, the agent has all information necessary to start acting in order to reach a solution whereas a partially observable environment requires for the agent to take extra action to observe as much as it can (example being looking both ways before crossing the street) to then use that information to make a judgement call towards the solution. Even then accidents can occur in a partially obervable environment, because sometimes even when an agent observes everything it needed to make a rational decision, there are cases where other problems arise that have nothing to do with the rationality of the decision the agent had made and may cause the agent to fail its task. The less observable an environment, the higher the risk of failure of a task, but it doesn't always mean that the agent is no longer rational.

2)[25 points] You are given a simple scheduling problem. You have a group of M people and a set of N tasks, each with a duration. Assume that N > M. You want to assign tasks to people so that the time when the last task is completed is minimized.
Formulate the problem using a state-space search representation. Describe (precisely):

how you would represent the states (i.e., what information needs to be included in any state to make sure the state describes all the information you need to apply the actions);

The states would be represented as a 'free' person (any one person who has not currently been assigned a task nor is actively working on completing a task) and a 'busy' person (a person who has been assigned a task and/or is actively working on completing a task).
The tasks are represeted as 'waiting' (not yet assigned to anyone and not being worked on), assigned (someone now is responsible for the task and has or will begin working on it), and completed.

what is the initial state;
Initial state of all N tasks is that they are not completed and not assigned to anybody.

Initial state of all M people is that they have not yet been assigned at least one task and are not working on a task, therefore they are 'free' to begin a task.

what goal test you would use;

Multiple goal tests can be used as benchmarks. We test all M people throughout the process of begining, during, and after completing the tasks (as we do not know how long each of the tasks take and they could all take different amounts of time) whether they are 'free', as was crudely defined above in the initial state.

```
loop:
If person is free
    if there is at least one task still not completed
        assign one task to the one free person
    else if there are no tasks
        check if all people are 'free'
            if all are 'free'
                then goal state reached
                break loop
Else person is NOT free
    move onto next person
```

what are the actions;

Actions for the people are to perform task, accept next task, and do nothing. When person is free, they 'do nothing' when there is no task available for them to do and thus they wait until goal state was reached. Accept next task/perform task are exactly what it sounds like it is. As soon as a person is free, their next course of action is to accept a new task and at that point they can work to 'perform' that task in order to complete it

what cost function you would use;

Since all N tasks could technically be performed by a single person, we will measure the cost function by the amount of time it takes to complete the tasks. Since it would take significantly longer for one person to complete all N tasks, we utilize all other available agents (M-1) as well to aid in completing the work in a timely manner, with each person accepting a new task as soon as they become free. This significantly reduces the cost of time by performing multiple tasks at the same time, similar to pipelining in code.

is the state-space a tree or a graph?

The state-space can be either, but is best represented as a tree. By splitting the tasks evenly (or as evenly as possible based on the time it takes for each task to be completed) among all M people, there is really only one direction to go when it comes to completing the tasks. The sole goal is to complete the tasks, and we add a restriction which is time, so the best way to complete the work is by pipelining all the tasks among the people.

[20 points] You are given an instance of the traveling salesperson problem (TSP). A salesperson has to visit a group of cities, visiting each only once and getting back to the starting city. Assume each city is directly connected to each other city. The objective is to minimize the total distance traveled.

Describe a state-space representation for the problem, specifying the initial state, goal state(s), and actions.

initial state: The sales person can start at any of the cities with all other cities yet unvisited.

goal state(s): Sales person having visited a city, and the sales person having visited all cities. Restricted by time, so it is prefered to travel the shortest distance combined between all cities.

actions: Sales person heads to next unvisited city (can be any that have not yet been visited), sales person heads back to a visited city (can be any that has already been visited), sales person stays in the same location.

state-space representation: represented by a graph/map. The sales person has the ability to travel anywhere to any city and can back track to any city he had previously visited. Whether backtracking saves total distance traveled, we do not know since it depends where the cities are located, how far apart they are, and where the starting location of the traveling sales person is.

For a problem with N cities, how many states are there in the search space? Explain briefly your answer.

There is always at least 1 visited city and N-1 unvisited cities because the sales person begins at one of the N cities, then as more cities are visited (we only count the first visit as a single visit, not a second time if the sales person backtracks to an already visited city) the number of unvisited cities decreases by the same amount the visited cities increase. The sales person can begin at any one of the N cities, so there are N many initial states, and (N-1)! many different ways to visit the cities (different states), therefore there are N * (N-1)! different states (or N!).

[20 points] This question is on properties of search algorithms. Consider the following algorithms:

depth-first,

breadth-first,

uniform cost,

depth-limited depth first,

iterative deepening depth first,

bidirectional.

You need to explain the reasons for your answers, not simply write the name of one algorithm.

if you want to limit the memory requirements, which algorithm(s) would you choose? why?

Depth-first because it always expands the deepest node in the current frontier of the search tree. It doesn't spend much memory as it blindly goes through every node as deep as it goes until it finds what it needs (Backtracking as necessary).

if you want to limit the time required the find a solution, which algorithm(s) would you choose? why?

Bidirectional would be best due to the two searches happening at the same time breadth first which would reduce the time needed to find the solution by 2.

if you want to find the minimum cost solution, which algorithm(s) would you choose? why?

Uniform cost- this allows for an approximate cost solution to be made at the start and decisions can be made based on this approximation.

if you want to solution with the minimum number of steps, but you do not want to use a lot of memory, which algorithm(s) would you choose? why?

Iterative deepening because it is a complete search and constantly sets new levels to continue searching in case the solution is not found in the current level. This is like depth first but with limited depth and sets a new depth whenever it can't find the solution.

#2 and 3 Programming part:

```
def test_ReflexVacuumAgent():
   # create an object of the ReflexVacuumAgent
   agent = ReflexVacuumAgent()
   # create an object of TrivialVacuumEnvironment
   environment = TrivialVacuumEnvironment()
   # add agent to the environment
   environment.add_thing(agent)
   # run the environment
   environment.run()
   # check final status of the environment
   assert environment.status == {(1, 0): 'Clean', (0, 0): 'Clean'}
   return (environment.status)
```

```
python3 -m doctest -v reflexvacuumagent.py
2 items had no tests:
   reflexvacuumagent
   reflexvacuumagent.test_ReflexVacuumAgent
0 tests in 2 items.
0 passed and 0 failed.
Test passed.
```

```
def test_RandomVacuumAgent():
   # create an object of the RandomVacuumAgent
   agent = RandomVacuumAgent()
   # create an object of TrivialVacuumEnvironment
   environment = TrivialVacuumEnvironment()
   # add agent to the environment
```

```
    environment.add_thing(agent)
    # run the environment
    environment.run()
    # check final status of the environment
    assert environment.status == {(1, 0): 'Clean', (0, 0): 'Clean'}
    return (environment.status)

python3 -m doctest -v randomvacuumagent.py
2 items had no tests:
    randomvacuumagent
    randomvacuumagent.test_RandomVacuumAgent
0 tests in 2 items.
0 passed and 0 failed.
Test passed.
```