Homework 2 - Stochastic Task

Introduction

In this exercise, you continue running the taxi business, but this time your clients exhibit non-deterministic behavior.

Task

The environment is similar to the one in Exercise 1: The same grid world, with the same passengers and taxis. Key differences include:

- The goals of the passengers are not predetermined but rather chosen from a set of possible goals and may change throughout the run. To put it another way, as the run progresses, passengers can change their goals from a set of possible choices.
- The agent is not required to output a complete plan but rather to output an action based on a state.
- The execution has a limited number of turns.
- The goal is to collect as many points as possible (more on that below).

The Stochastic Passengers

Each passenger has some initial position, a goal location, a list of possible goal positions, and a number (0-1) that describes the probability the passenger will change his goal each step. For example, let “Ron” be a passenger with the following parameters:

```
passenger = {
  “name”: “Ron”
  “location”: (0,0), “destination”: (5,5),
  “possible_goals” = ((5,5), (4,5), (3,3)),
  “prob_change_goal”:0.05
}
```

Here, Ron is currently in (0,0), his current goal is (5,5), but there is a 5% chance Ron draws a new goal location from the “possible_goals” list. At every turn, first, the taxis execute their actions and only then do the passengers reconsider their goals. That is, assuming the taxi has Ron and the taxi is in location (X, Y), which is the current desired location of Ron, the taxi can drop Ron and gain points. Then, Ron can reconsider his goal and again be handled by the taxis to earn more points.

Actions

The actions' syntax and rules remain as they were in Exercise 1, with two new actions:

1. **Reset**: It resets the places of passengers and taxis to the initial state with their original fuel levels. The action does not reset the number of turns or the points accumulated. It is advisable to use this action when all the passengers are at their goal location so you can earn more points, but this may not be the only use. The syntax is simple – "`reset`".
2. **Terminate**: This action terminates the execution before the turns run out. This may be of use if resetting yields negative expected results. The syntax is simple – "`terminate`".
3. The rest of the actions ("move", "pick up", "drop off", "refuel", "wait") stay the same.
4. Important notes:
    a. It is now illegal to pick up a passenger that is located in his goal location.

Points

In this exercise, your goal is to achieve maximum points. Points are given for the following:

- Successful delivery of a passenger: 100 points.
- Resetting the environment: -50 points.
- Refuel: -10

The input

The input is presented as follows:

```
{
    "optimal": True,
    "map": [['P', 'P', 'I', 'P'],
            ['P', 'P', 'P', 'P'],
            ['P', 'I', 'G', 'P'],
            ['P', 'P', 'P', 'P'],],
    "taxis": {'taxi 1': {"location": (3, 3),
                         "fuel": 15,
                         "capacity": 2}},
    "passengers": {'Dana': {"location": (0, 0),
                            "destination": (2, 3),
                            "possible_goals": ((5, 5), (4, 5), (3, 3)),
                            "prob_change_goal": 0.05}},
    "turns_to_go": 100
}
```

Everything is the same as it was in Exercise 1 beside:

- "`optimal`" indicates that the agent must act optimally.
- The extra parameters that describe the passengers.
- "`turns_to_go`" are the number of turns the execution lasts.

The Task

Code-wise, your task is to implement two agents (TaxiAgent and OptimalTaxiAgent). Each agent shall have (at least) two functions:

- `__init__(self, initial)` – A constructor.
- `act(self, state)` - A function that returns an action given a state.

Some important notes:

- As you could have guessed, the OptimalTaxiAgent is required to solve the problem optimally. The algorithm you learned in class could be useful here.
- The TaxiAgent doesn't have to be optimal, yet we expect it to gain strictly positive scores.
- As compared to the OptimalTaxiAgent, the TaxiAgent will need to handle larger problems.
- Timeouts for initiating an agent and choosing an action are 300 seconds and 0.1 seconds, respectively.

Code handout

Code that you receive has 3 files:

1. ex2.py - The only file you should modify, which implements your agent.
2. check.py - The file that implements the environment, the file that you should run.
3. utils.py - The file that contains some utility functions. You may use the contents of this file as you see fit.

Note: We do not provide any means to check whether the solution your code provided is correct, so it is your responsibility to validate your solutions.

Submission and grading

You are to submit only the file named ex2.py as a python file (no zip, rar, etc.). We will run check.py with our inputs and your ex2.py. The check is fully automated, so it is important to be careful with the names of functions and classes.

The grades will be assigned as follows:

- 60% - If your TaxisAgent finishes solving the test inputs with a strictly positive score.
- 20% - If your OptimalTaxisAgent solves all given problems optimally.
- 20% - Grading on the relative performance of the algorithm, based on points.
- There is a possibility to earn bonus points by reporting bugs in the checking code. The bonus will be distributed to the first reporter.
- The submission is due on 20.12.2022, at 23:59.
- Submission in pairs/singles only.
- Write your ID numbers as strings in the appropriate field ('ids' in ex2.py). If you submit alone, leave only one string.

- The name of the submitted file should be “ex2.py”. Do not change it.

Important notes

- You are free to add your functions and classes as needed. Keep them in the ex2.py file.
- We encourage you to double-check the syntax of the actions as the check is automated. More inputs will be released a week after the release of the exercise.