ASSIGNMENT 4: ELEVATOR CONTROL

Goal: The goal of this assignment is to find a solution to a realistic problem using ideas from sequential decision making under uncertainty (Markov Decision Processes).

Scenario: Imagine a building not unlike the Lecture Hall Complex. You are responsible for the elevator control algorithm so that overall wait time and your electricity consumption is optimized.

The setting is a typical setting in which

1. There are N floors (numbered 1 to N), and K elevators (1,…,K).
    a. Simplifying assumption 1: Each elevator can carry infinite people.
2. At each floor (except 1 and N) there are two push buttons, one to call the lift to go up (BU_f) and one to call the lift to go down (BD_f); here f is the floor number. BU stands for button up, and BD stands for button down. At floor 1 there is only BU_1 button and at floor N there is only BD_N button.
3. Whenever a person needs to use a lift, he/she presses the appropriate push button (if it is not already pressed).
    a. Simplifying assumption 2: In any unit time t_u, no more than 1 person enters the system (i.e., arrives at a floor and presses the push button). Whether the person arrives or not, is governed by a single parameter p.
4. Whenever an elevator stops at a floor, it indicates which direction it is going by a blinking light. The lights are LU_f and LD_f, for up light and down light for each floor. Of course, floor 1 only has LU_1, and floor N only has LD_N.
5. Whenever an elevator stops at a floor and opens, in one time unit whoever had to exit at this floor exits. And anyone, who had to enter at this floor, enters. All people who exit the floor, exit the system and don't incur further wait costs.
6. Inside each elevator e, there are N push buttons BF_e, where F is the floor number. For example, elevator 1 has buttons $B1_1$, $B2_1$,…, BN_1. Whenever a person enters e, he/she presses the floor number they wish to go to.
7. The lift can go in any direction irrespective of the buttons BFe pressed.
    a. Simplifying assumption 3: if a floor f is on the elevator's way and the button Bf_e has been pressed then the elevator is bound to stop there.

8. At any time unit t_u,
    a. An elevator can move 1 floor up or down (AU_e, AD_e – action up for elevator e, action down for elevator e)
    b. An elevator can choose to open itself in its current floor in the direction up and down. (AOU_e , AOD_e)
    c. An elevator can stay still. (AS_e)
9. Arrival model. As mentioned in assumption 2 above, after every time point, exactly 1 person arrives with probability p, and no person arrives with probability 1-p. Where the person arrives is governed by a high probability q for floor 1 and equal probabilities (1-q)/(N-1) for other floors. Where they wish to go is decided by equal probability 1/(N-1) for people arriving at floor 1. For other floors, they wish to go to floor 1 with probability r, and the rest of the floors equally with probability (1-r)/(N-2).
10. Corner case: if an elevator opens at a time point and a new person arrives at the same floor going in the same direction, even then we assume that the elevator barely misses the new person.
11. Costs
    a. Each waiting person incurs a cost of 2 per time point that they are waiting, whether at the floor or in the elevator. They stop paying as soon as they exit the system.
        i. Notice that the elevators don't know the *exact* number of people waiting at the floor or moving in the lift. So, you will need some approximate model of cost here for your model.
    b. Each action AU and AD costs 1 in electricity charges. Actions AS and AO don't cost any electricity cost.

Problem Statement: In this assignment your task is to control the elevators. As a first step we suggest you model the problem as an MDP. First, you carefully think of the MDP inputs that you will need.

- What will you keep in the state space
- What will be your actions
- What will be your transition function
- What will the cost of taking an action in a state

It is possible that the state space you define is HUGE. This means that you could run out of memory just in storing the MDP. Or Value Iteration may take very long. That is a real problem in AI. Here, you will have to be creative. Some ideas of how to deal with large state spaces are:

1. Find a compressed data structure to represent a state.
2. Think about whether the transition function should be cached or computed on the fly. (Because the transition function is $O(S^2A)$, this takes maximum amount of storage)
3. Prove that some actions will never be optimal in a state, and remove them from analysis.
4. Prove that some states are symmetric and hence will have similar values and similar actions. Reduce the MDP model to deal with 1 representative state per symmetry group.
5. Consider modeling the problem as hierarchies of MDPs in which you divide the task somehow into multiple abstract MDPs. First you decide some top level action and then you refine it further in the bottom layers. It will require some thought to model the problem this way.

If you are able to compute the optimal policy using Value Iteration or Policy Iteration (or its variants) in less than a half hour, call it success!

If, however, you are unable to compute the optimal policy for the whole state space, don't be disheartened -- you may have to perform interleaved planning and execution or approximate in other ways. Some ideas:

1. Use UCT algorithm instead of value iteration to learn just the next action.
2. Use function approximation ideas to approximate the value function for the whole state space.
3. Define some reasonable base policy that provides the exact action in many states. This reduces the MDP and you solve for the rest of the state space.

These are the constraints on the parameters (for the main part of the assignment):

N will be between 4 and 5.

K will be exactly 2.

q and r will be high-ish numbers, say between 0.25 and 0.75. Basically, more people will enter/exit the 1^{st} floor.

p can have any value.

$t_{\underline{u}}$ will be 1-5 seconds, likely closer to 1 sec.

Bonus part:

Just to understand this problem further we will optionally test you with a building where N=50 or more, and K=3 or more. This is only for extra credit (up to 4 marks). Not everyone has to attempt this.

Input/output format:

You start the controller with the parameters N K p q r t_u. For example ./run.sh 5 2 0.8 0.5 0.5 1 says that there are five floors, 2 elevators, 0.8 probabilty that someone arrives in the next time step, and 0.5 probability each for someone arriving at the first floor and wanting to go to the first floor. Each time unit is 1 secs of wall clock time.

After this we will wait for the controller to print

0

This basically says that the controller has done any upfront computation it wanted to do and is now ready to control the elevators. There is a maximum time limit of 30 minutes for this. If the controller doesn't print 0 after 30 minutes, then the controller will be considered timed out. Once we get a zero, we start the game assuming both elevators are at floor 1. After each unit time we input one of {0, BU1, BU2, BD2, … , BDN}. Within t_u, you need to output actions, one for elevator. Your actions will be one of {AU1, AD1, AO1, AS1}, one of {AU2, AD2, AO2, AS2} and so on. A possible interaction sequence is

python sim.py run.sh 5 2 0.8 0.5 0.5 2

0 //within 30 mins controller says its ready

0 //no one enters the system

AS1 AS2 //controller decides to not move any elevator

BU4 //someone (say P1) wants to go up from floor 4. Only possibility for target is floor 5

AU1 AS2 //controller decides to move lift 1 up

BD2 //a new person (P2) arrives at floor 2. Wants to go down. If lift1 wishes it can open now since its at floor 2

AU1 AU2 //controller decides to move both lifts up… after this lift 1 will be at 3 and lift 2 at 2

BU3 //P3 arrives at floor 3. Wants to go up. Lifts do not know where.

AOU1 AOD2 //both lifts open. One at level 3 going up and one at level 2 going down. P2 and P3 enter because the directions match.

0 B51 B12 // no new person enters. Button 5 is pushed for lift 1 and button 1 for lift 2. (P3 wants to go to 5)

AU1 AD2 //lift 1 goes up, lift 2 goes down

0 //no new person enters

AOU1 AOU2 //lift1 opens going up, lift 2 opens going up. P2 exits the system; P1 enters lift 1 going up

0 //no new person enters

AU1 AS2 //lift 1 goes up, lift 2 stays

0 //no new person enters

AOD1 AS2 //lift 1 opens at floor 5. Both P1 and P3 exit

0 //no new person enters

AS1 AS2 //lifts don't move

…

It took 7 time units for P1, 4 time units for P2 and 5 for P3. So total wait cost is 32. Total electricity cost is 4 for elevator 1, and 2 for elevator 2. So total cost is 32+6=38.

Code

Your code must compile and run on our VMs. They run amd64 Linux version ubuntu 12.04. You are already provided information on the compilers on those VMs. These configurations are similar to GCL machines like 'todi' (have a RAM of 16 GB). Please supply a compile.sh script and a run.sh script. We will run your code as follows:

./compile.sh

python sim.py run.sh N K p q r t_u

(sim.py will call run.sh with the parameters N K p q r t_u, analogous to Assignment 3)

All subsequent interactions with the controller will be over stdout.

What is being provided?

We provide a way to interact with the simulator where the system plays a random valid move. And writes the execution details to the file simulation.txt so that you can compute the cost entered so far and the simulation state. You can find the simulator here: https://github.com/suragnair/lift-sim

What to submit?

1. Submit your code in a .zip file named in the format **<EntryNo>.zip.** If there are two members in your team it should be called <EntryNo1>_<EntryNo2>.zip. Make sure that when we run "unzip yourfile.zip" the following files are produced in the present working directory:

   compile.sh

   run.sh

   writeup.txt

   You will be penalized for any submissions that do not conform to this requirement.

2. The writeup.txt should have two lines as follows

   First line should be just a number between 1 and 3. Number 1 means C++. Number 2 means Java and Number 3 means Python.

   Second line should mention names of all students you discussed/collaborated with (see guidelines on collaboration vs. cheating on the course home page). If you never discussed the assignment with anyone else say None.

   After these first two lines you are welcome to write something about your code, though this is not necessary.

Code verification before submission: Your submission will be auto-graded. This means that it is absolutely essential to make sure that your code follows the input/output specifications of the assignment. Failure to follow any instruction will incur significant penalty. The details of code verification will be shared on Piazza (similar to previous assignments).

Evaluation Criteria

1. The main part of the assignment is worth 10 marks. Evaluation is via a final competition by changing problem size and parameters. The points awarded will be your normalized performance relative to other groups in the class.

2. **2.** The bonus part of the assignment can fetch you up to 4 marks. If you do this part, please submit an additional writeup.pdf outlining your approach to this part.

What is allowed? What is not?

1. You may work in teams of two or by yourself. If you work in a team of two then make sure you mention the team details in the write-up. You cannot use the partner from previous assignments. Our advice: this is a fairly challenging assignment and you may have to try multiple ideas before one succeeds. Having a partner will help in brainstorming and sharing of the load.

2. While you are allowed one of three languages C++, python, java. There is no time constraint in the assignment (however your code must finish in reasonable time). We suggest you use C++ for the assignment as it requires a time bound response.

3. Your code must be your own. You are not to take guidance from any general purpose AI code or problem specific code meant to solve this or related problem.

4. It is important to develop your algorithm using your own efforts. Please do not google search for solutions to the assignment.

5. You must not discuss this assignment with anyone outside the class. **Make sure you mention the names in your write-up in case you discuss with anyone from within the class outside your team.** Please read academic integrity guidelines on the course home page and follow them carefully.

6. You get a zero if your player does not match with the interaction guidelines in this document.

7. We will run plagiarism detection software. Any team found guilty will be awarded a suitable penalty as per IIT rules.