CSE 471: Introduction to Artificial Intelligence
Project #2

Due Date: September 25 2018, 11:59 PM

Instructions

Running your code

- You should use the online planner available at http://editor.planning.domains
  You can load your files by selecting 'Load' in the 'File' drop down menu, selecting your domain/problem file and hitting the load button.
  To run the planner, click on the 'Solve' menu, select your domain and problem file from the drop down menu and hit 'Plan'
- More information about the planner can be found here : http://solver.planning.domains/ , if you are interested.
- Your domain/problem file, if correctly designed will yield a solution quickly (in under 10s) on the online planner. Timeouts could mean that something is wrong with your domain or problem file.
- You may also configure an offline planner, if you prefer to do so, by following the setup instructions found here: http://lapkt.org/index.php?title=Download

Submitting your solutions

- File naming convention for each question:
  q<question_number>_<domain|problem|plan>.<pddl|txt>
  For example, q1_domain.pddl, q2_problem.pddl, q1_plan.txt. Similarly for other questions.
- Files to be submitted for **each** question:
  PDDL Domain, PDDL Problem and Plan file (solution)
  Apart from this, you need to submit a single scatter plot (in PDF format) showing a graph of time taken to generate the plan vs question number. Rename the file as plot.pdf

Submit your files in a zipped folder named as $\langle ASU\_ID \rangle\_\langle last\_name \rangle\_\langle first\_name \rangle\_project2.zip$. For example, $1234567890\_kumar\_kislay\_project2.zip$. Submit your zipped file through canvas assignment submission link.

Questions

1. Below is a 4X4 maze with locations defined as X_iY_i, Pacman's current position and a goal state is also indicated. There are no walls and food pellets currently. These will be added later on once you start progressing in the assignment. Two files are required to represent any environment that you want to solve using PDDL:

i Problem file - which would contain information about your objects in the environment, your initial state and a goal state.

ii Domain file - which would contain list of predicates in use, types for objects (similar to data type in any programming language) and actions.

A problem file (q1_problem.pddl) representing the below knowledge is provided for your reference. You are required to write similar problem files for remaining questions. A partially filled domain file (q1_domain.pddl) is provided with one action definition. You are required to fill in the remaining actions to make pacman reach the goal state. Once you complete the domain file, you need to pass both of the files to the planner (refer instructions) to generate a plan. Make sure that the plan returned by the planner makes sense and reaches the goal. (9 Points)

X_1Y_1, Start	X_1Y_2	X_1Y_3	X_1Y_4, Goal
X_2Y_1	X_2Y_2	X_2Y_3	X_2Y_4
X_3Y_1	X_3Y_2	X_3Y_3	X_3Y_4
X_4Y_1	X_4Y_2	X_4Y_3	X_4Y_4

2. Create a maze (4X4) as shown below with pacman's current position, a goal state and two randomly selected locations as blocked (pacman can't move into this location). Blocked locations are student specific. To get your personalized blocked locations, run the provided python script : getBlockedLocations.py along with your ASU ID as an argument. For example, python getBlockedLocations.py 1234567890 where 1234567890 is your ASU ID (use your own asu ids here). After you get your blocked locations which would be in the form of X_iY_i, change the Initial State (:init) in the problem file (q2_problem.pddl. write your own file) to incorporate this information (i.e. say you got one of the blocked location as X_3Y_3, then there will be no predicate saying X_3Y_3 is right of X_3Y_2). You need to remove all such predicates that would be consistent with the above knowledge. Finally, you need to run planner (refer instructions) to see the generated plan and check if it makes sense (it shouldn't have any action that makes the pacman enter in any of the blocked location) and is able to reach the goal. (4 Points)

X_1Y_1, Goal	X_1Y_2	X_1Y_3	X_1Y_4
X_2Y_1	X_2Y_2	X_2Y_3	X_2Y_4
X_3Y_1	X_3Y_2	X_3Y_3	X_3Y_4
X_4Y_1	X_4Y_2	X_4Y_3	X_4Y_4,Pacman

3. Lets now add food pellets as pacman might be getting hungry after all the hard work we have put him through. With the same maze given in question no. 2, consider you have food pellets in the remaining locations (except for the pcaman's current position, goal state and the blocked locations). Your task will be to help pacman eat all the food pellets. For this, you would have to add an EatDot action in the domain file (q3_domain.pddl. Use q2_domain.pddl contents which

you wrote for question 2 and add the EatDot action) that will be executed only if the Pacman is at a location with a dot, resulting in the dot being eaten. Also, you will need to add a predicate DotAt indicating whether or not a dot is at a location and a predicate Eaten indicating whether or not a dot has been eaten. Use these predicates to create a goal condition that requires all available food pellets to be eaten. Once again, run the planner to compute a plan for eating all food pellets in your dot configuration and check if it makes sense (it shouldn't have any action that makes the pacman enter in any of the blocked location) and is able to eat all food pellets. (5 Points)

4. Now for more fun, we will add one more Pacman in the game. For this, you would have to change the problem file (q1_problem.pddl, q2_problem.pddl and q3_problem.pddl) and save it as q4_1_problem.pddl, q4_2_problem.pddl and q4_3_problem.pddl adding one more pacman object in the object declaration and a starting position randomly(make sure the random position is different from the start position of the first pacman, goal and blocked cells) for the other pacman. Once you have added another pacman, solve all the previous questions (Question 1,2,3) again with both the pacman reaching the goal location for question 1 and question 2, and for question 3 just eating all food pellets. (12 Points)