15-887 Homework 3

Assigned on: 10/19/16
Due date: 11/02/16

Homework Submissions: Code and report should be packaged in a zip file and e-mailed to mmv@cs.cmu.edu, maxim@cs.cmu.edu, and ruisilva@cmu.edu. Title of your e-mail should be "15-887 Homework 3 - your andrew id". Name your compressed file as "{your andrew id}_hw3.zip".

1 Execution

1.1 [5pts]

For ARA* with a finite $\epsilon > 1$ and consistent heuristics, consider the following statements and mark ALL that are true:

a. Every search iteration (i.e., every call to ComputePathwithReuse as shown in slides) expands at least one state

b. Every search iteration performs at least as many expansions as the number of states expanded by weighted A* with the corresponding inflation factor ϵ.

c. A search iteration may perform zero state expansions (that is, expand no states at all) yet guarantee that a previously found solution is at most ϵ-suboptimal.

d. The last search iteration for $\epsilon = 1$ is guaranteed to perform at least as many state expansions as what normal A* with un-inflated heuristics would perform.

e. None of the above.

1.2 [15pts]

Consider LRTA* with $N = 1$ expansion limit (slide 14 in the lecture on Real-time Heuristic Search). Prove that after every update of the heuristic function, the heuristics remain to be consistent (assuming they were consistent at the very beginning).

2 Policy Iteration [15 pts]

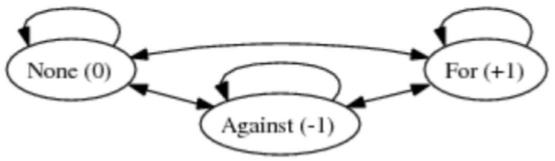
Consider an MDP with three states capturing scoring in robot soccer: *None*, *Against*, and *For* with reward 0, -1, +1, respectively.

Also consider three actions capturing playing strategies:

1. Balanced: 5% chance we score; 5% chance opponent scores.

2. Offensive: 25% chance we score; 50% chance opponent scores.

3. Defensive: 1% chance we score; 2% chance opponent scores



a	**T(*, a, For)**	**T(*, a, Against)**	**T(*, a, None)**
Balanced	0.05	0.05	0.9
Offensive	0.25	0.5	0.25
Defensive	0.01	0.02	0.97

The actions imply the following transition probabilities among the three states, where * means any of the three states:

(a) What is the total number of policies of this MDP?

(b) With discount factor 0.5, solve this MDP using policy iteration.

(c) For the specific given MDP, will different discount factors change the optimal policy? Provide a brief answer.

3 Q-learning [35pts]

You are to implement the Q-learning algorithm. Use a discount factor of 0.9. We have simulated an MDP-based grid world for you. The interface to the simulator is to provide a state and action and receive a new state and to provide a state and receive the reward from that state. The world is a grid of a pre-specified number of cells, which you should represent in terms of the x and y coordinates of each cell. There are four possible actions, *North(N)*, *South(S)*, *East(E)*, and *West(W)*. All the actions are non-deterministic.

The simulator, implemented in C++, is available on the website. The interface is:

1. my_next_state(State, Action)
2. my_reward(State)

where State and Action are defined in mdp-simulation.h. More instructions can be found in the README included with the simulator. Please submit the learned policy in terms of a C++ function and a Q-table that we can query if needed. In your written submission, please include:

i) a readable representation of the learned policy

ii) a graph that shows the reward gathered by successive episodes of length 100 steps starting in some random position, as learning progresses.

4 Policy Reuse [10pts]

How does the π-reuse strategy contribute with a similarity metric between policies? Provide a brief answer.

5 Experience Graphs [5pts]

Consider planning with E-graph as was explained in the lecture. Furthermore, assume you are running A* with un-inflated h^E function. Please mark ALL the statements below that are true:

a. Experiences in E-graph cannot form disconnected components in order to guarantee completeness.

b. Experiences should be no worse than ϵ^E suboptimal in order to guarantee bound on suboptimality.

c. Assuming you start with all experiences in E-graph that are optimal paths between their corresponding start and end states, all the subsequent plans generated by the E-graph planner will be optimal as well.

d. None of the above.

6 Dependent vs. Independent Variables [15pts]

Consider a 2D (x,y) path planning problem for a point robot that has a battery power enough to make at most N steps. Naturally, as the cost function to optimize, you decided to use the number of steps the robot needs to make to reach its goal. As discussed in class, this way you can still run A* on the original 2D path planning problem without bringing in the battery power level as additional state variable. The only change to A* is to disallow all edges that result in states with g-values larger than the initial battery power. Suppose now you decided to speed up the search by inflating heuristics (i.e., running weighted A*) while maintaining the property of A* that it expands each state at most once (that is, it uses CLOSED list to avoid re-expansions of states). Will the algorithm be complete? Prove its completeness or find a counterexample where it will fail to return a solution even though it exists.