HW1 Wet in 046203 Planning and Reinforcement Learning

Submitter 1: David Valensi 342439643
Submitter 2: Yaniv Galron 206765646

Question 1: Solving 8-Puzzle with Dijkstra's Algorithm

1. We have 9 numbers that can be placed in each of the 9 possible locations on the grid. Therefore, the number of possible states is $9!$. In fact, the number of reachable states is half of that number and equals $S = \frac{9!}{2} = 181440$. This number is too big to run minimization of all the states in each iteration. Instead, we will not save all the states but only the nodes we reach to and its neighbors (that are not already in the graph)
2. CODE

Question 2: Solving 8-Puzzle with A* Algorithm

1. The admissible heuristic we will use for the 8-Puzzle will be
$$h[v] = MD(v_x, t_x) + MD(v_t, t_y)$$
when we sum over all tiles x,y current position vs x,y end position. This heuristic is admissible ($h[v] < d[v,t]$) because at each action only one of the digits is moved by a distance of '1' to the x or y direction. The shortest path possible is If we could move each digit directly to the end position and we would get $h[v] = d[v,t]$
2. CODE
3. This heuristic is admissible also and we can know that from the fact that if a tile is not in the correct place than the value of this function will be 1 and the value of the admissible heuristic we used before will be greater of equal to 1 so $h_{NC}[v] \leq h_{MD}[v] \leq d[u,t]$
With this "num – incorrect" heuristic the algorithm visited 1816 states, compared to 218 with the MD heuristic. The MD visits less states and takes 9 times less time. We think this is because that the MD heuristic is more bounded and closer to the real cost of the states. $d[u,t]$. The NC heuristics doesn't give any clue about the direction to take while MD heuristics gives better score for a good tile move toward its destination.
4. The following eight-puzzle instance take 27 moves to solve:
$$\begin{matrix} 8 & 6 & 7 \\ 2 & 5 & 4 \\ 3 & - & 1 \end{matrix}$$
Dijkstra algorithm took: 25.78 seconds to complete and it visited 176184 states
A* algorithm took 0.37 seconds and it visited 2194 states
5. Let's compare the result on the given puzzle.
For $\alpha = 1$ $number\ of\ states\ visited$: 218 $time$: 0.03 $seconds$ $plan\ length$ 19
For $\alpha = 0$ $number\ of\ states\ visited$: 30,242 $time$: 4.83 $seconds$ $plan\ length$ 19
Same number of visited states but longer time (3.59 seconds)
For $\alpha =$
6 $same\ as\ infinity\ because\ number\ of\ states\ doesnt\ change\ with\ bigger\ \alpha\ we\ get$
$number\ of\ states\ visited$: 53 $time$: 0.0059 $plan\ length$ 21

We can see that as we increase alpha the number of visited states and the running time is decreasing but if we make alpha too large the heuristic becomes non-admissible, and we got a non-optimal solution.

Question 3: LQR

1. As shown in class we can linearize the inverted pendulum on a cart around $\theta = 0$:

$$\dot{X} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{m}{M}g & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{g}{l}\left(1+\frac{m}{M}\right) & 0 \end{pmatrix} X + \begin{pmatrix} 0 \\ \frac{1}{M} \\ 0 \\ \frac{1}{Ml} \end{pmatrix} u_t = \bar{A}X + \bar{B}u_t \ .$$

When approximating with first order:
$X_{t+dt} = X_t + dt * \dot{X}_t \Longrightarrow X_{t+dt} = AX_t + Bu_t$ with $A = I + dt * \bar{A}, B = dt * \bar{B}$
We fill the code accordingly.

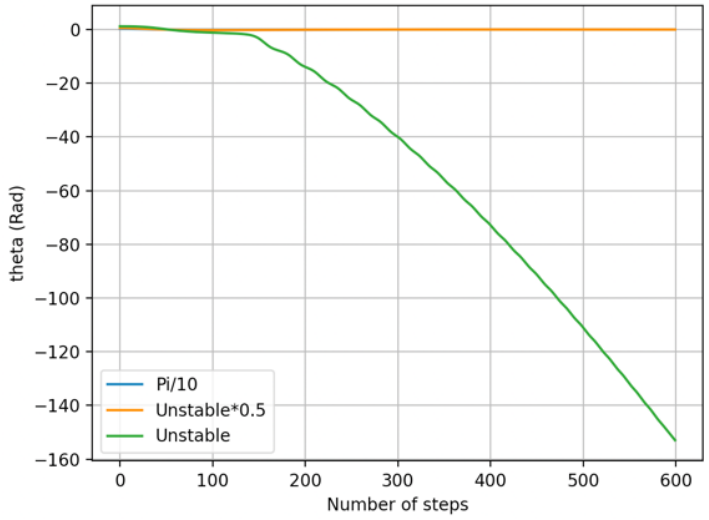
2. We choose the following cost parameters to stabilize the pendulum with initial position $\theta \in \left[-\frac{\pi}{10}, \frac{\pi}{10}\right]$:

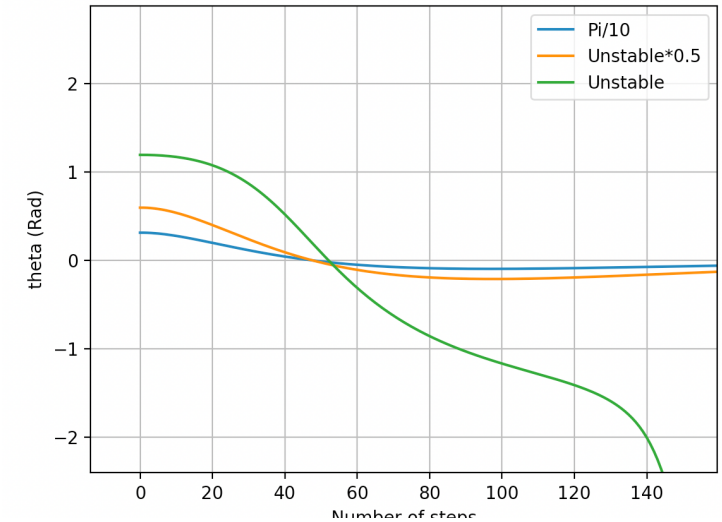
```
Q = np.matrix([
    [0.5, 0, 0, 0],
    [0, 1, 0, 0],
    [0, 0, 1, 0],
    [0, 0, 0, 1]
])

R = np.matrix([1])
```

We recall that Q matrix is applied to the state vector $X = \begin{pmatrix} X \\ \dot{X} \\ \theta \\ \dot{\theta} \end{pmatrix}$ and R is applied to the u_t action.
We consider in the cost function:

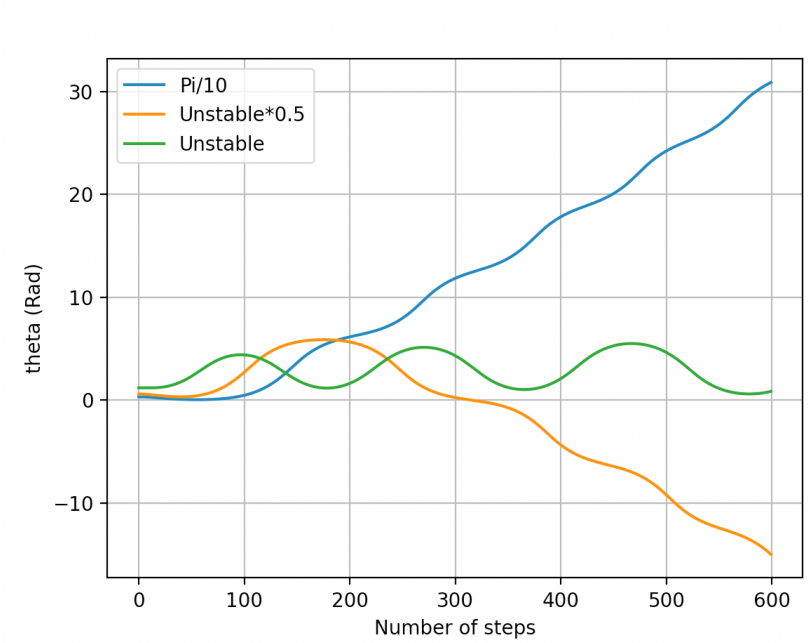
- Cart position, we don't want the cart to get away from its start point too much,
- Cart velocity, we want the cart to be relatively stable and not increase velocity,
- Pendulum angle, of course we want to have small pendulum angle,
- Angular velocity since we want stability in the angle also.
- The force of the action, since applying the force on pendulum may have a cost in real-life and we don't want to over utilize it.

3. We find that $\theta_{unstable} = 0.38 * \pi$ under these cost parameters.
   The plots of theta over time are as follows:





   As we can see, for $\theta \in [0.1\pi, 0.5 * \theta_{unstable}]$, the pendulum converges to stable upright position whereas for $\theta = \theta_{unstable}$, the pendulum keeps diverging.
   The difference is due to the limitations of the LQR model, it approximates the pendulum dynamics as a first order-approximation around $\theta = 0$. Then, when $\theta = \theta_{unstable}$, the approximation doesn't hold as well as for small angles.
4. The plots using the feedforward control law are as follows:



This LQR control law doesn't converge to the upright pendulum position even for small initial angles like 0.1π.

We conclude easily that the feedback LQR control is better than the feedforward LQR control. In feedforward control law, the actions are computed only based on predicted states of the pendulum. However, the pendulum dynamics are approximated (with first order approx..) during LQR computation. This creates a difference with the real state computed by the simulator (using full dynamics) and the predicted actions don't fit at all to the actual states.

5. Using our previous cost parameters, and limiting the maximal force to 4, we find that the pendulum converges to upright position when initial theta is 0.1π.

   Also, we observe with some trials that $\theta_{unstable} = 0.115\pi$.

   However, it is obvious that if we increase the action cost parameter R, then the LQR control law will likely stabilize with larger initial theta (increasing $R = 1$ to $R = 200$ makes the initial angle $\theta_0 = 0.115\pi$ stable) since the larger R, the smaller forces the LQR controller applies.

   Here are the plots with $\theta_{unstable} = 0.115\pi$: