Wet1 - 046203

Zohar Rimon - 319010534 , Yair Amar - 209455179

07.05.2020

1 Solving 8-Puzzle with Dijkstra's Algorithm

1.1

For a general NxN board: $|S| = (N * N)!$ and for a 3x3 board: $|S| = 0.5 * 9! = 181440$. This is too much states to run minimization over all of them in each iteration. Instead of building the entire graph in advance, we can save only the states that have anon-infinite distance to the target. This is because we do not need to minimize over the infinite - distanced state. In order to implement this, whenever we reach a node, we expand the graph to it's neighbors (if they are not yet in the graph).

1.2

CODE

2 Solving 8-Puzzle with A^*

2.1

We will define our admissible heuristic for the 8-Puzzle as the sum of MD between the current state and the target state x's and y's: $h[v] = MD(v_x, t_x) + MD(v_y, t_y) = \Sigma_{i=1}^{8} |v_{ix} - t_{ix}| + |v_{iy} - t_{iy}|$ where $v_{iy}$is the y coordinate of digit 'i' in state v. This heuristic is admissible ($h[v] \leq d[u,t]$) 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 target corresponding location and we would get $h[v] = d[u,t]$.

2.2

CODE

2.3

This is obviously an admissible heuristic because: $h_{NC}[v] = \Sigma_{i=1}^{8} I(v_{ix}! = t_{ix}||v_{iy}! = t_{iy}) \leq \Sigma_{i=1}^{8} |v_{ix} - t_{ix}| + |v_{iy} - t_{iy}| = h_{MD}[v] \leq d[u,t]$ The first inequality is due to the fact that if a digit is not in the same place than $|v_{ix} - t_{ix}| + |v_{iy} - t_{iy}|$ is greater or equal to 1. With the "num-incorrect" heuristic we visited 1816 states, compared to 218 with the MD heuristic. The MD is much better, the run time was almost 8 times faster with it. We think that the MD heuristic is better because it reaches a stricter bound to d[u,t].

2.4

We tested the algorithms on the following puzzle (which takes 27 steps to complete.):

$$\begin{pmatrix} 8 & 6 & 7 \\ 2 & 5 & 4 \\ 3 & 0 & 1 \end{pmatrix} \tag{1}$$

The Dijkstra's algorithm took 23.7 seconds to complete and it visited 176,184 states.
The A^* algorithm (with the manhattan distance heuristic) took 0.36 seconds to complete and it visited 2194 states.

2.5

The purpose of the heuristic function is to "push" the algorithm in the right direction and look at more relevant states. The α parameter will control the trade-off between the actual graph-distance and the heuristic.
For $\alpha = 0$ we get the Dijkstra's algorithm, that does not have any heuristic function.
For $\alpha = 1$ we get the A^* algorithm.
For $\alpha = \infty$ the heuristic will no longer be admissible and the algorithm will not converge to the best solution. This can be intuitively understood, because the algorithm will choose the minimal node, only with respect to the heuristic and not the graph-distance.
For $\alpha \geq 1$ we risk that the heuristic function will not be admissible and the algorithm will not converge to the optimal solution.

2.6

We compared the results on the "hard puzzle":

$$\begin{pmatrix} 8 & 6 & 7 \\ 2 & 5 & 4 \\ 3 & 0 & 1 \end{pmatrix} \tag{2}$$

For $\alpha = 0$: 30.85 seconds, 176,184 states (the running time is different from Dijkstra's algorithm because we are calculating the manhattan distance, but the number of state visitations is the same), and path length of 27 (Optimal).
For $\alpha = 1$: 0.35 seconds, 2194 states (Like the A^* algorithm) and path length of 27 (Optimal).
For $\alpha = \infty$ (actually 100, for bigger alphas we didn't cant any change in the number of visited states, thus we achieved the wanted effect): 0.018 seconds, 138 states and path length of 41 (very not optimal)

We can see that as we increase α the number of visited nodes and the running time is decreasing. But if we make α too large (and the heuristic becomes non-admissible), we get a non-optimal solution.

3 Solving Cart-Pole with LQR

3.1

CODE

3.2

As seen in class we used the following R and Q:

$$Q = \begin{pmatrix} w_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & w_2 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, R = w_3 \tag{3}$$

where:

$$\mathbf{x} = \begin{pmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{pmatrix}, u = F \tag{4}$$

This R and Q demand that θ, x, F will be small. The cost on θ is due to the fact that we want the algorithm to reach $\theta = 0$. We also want the algorithm to use little force and not to move by a large amount from the center of the map. Empirically, we chose:

$$w_1 = 0.7, w_2 = 1, w_3 = 0.05, \tag{5}$$

3.3

Empirically we found: $\theta_{unstable} = 0.33\pi$
In Figure 1, first, we can see that for the non stable initial angle, the path is indeed not stable. Furthermore we can see that for larger initial angle, the system takes longer to stabilize itself

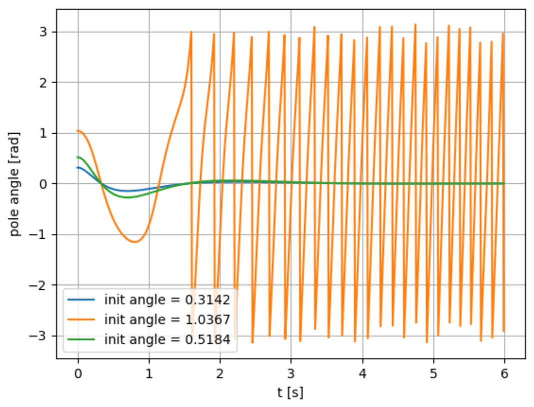

Figure 1: Plotting of the actual pole angle with different initial angles using online-decided control

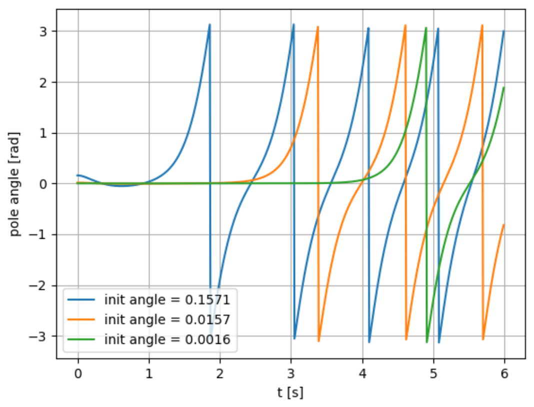

Figure 2: Plotting of the actual pole angle with different initial angles using predetermined control

3.4

Obviously, the better control is the one calculating the current action based on the actual current state. This is due to the fact that we have a cumulative error from linearization error from each state.
We can see in figure 2, that even for small initial angles, the system eventually is unstable.

3.5

We only changed the weight w_3 that is controlling the cost of using high value actions (force in our case). Empirically we raised w3 to 0.5. In Figure 3 we can see that $\frac{1}{2}\theta_{unstable}$ also leads to unstable result, this is due to the limitation on the force allowed by the simulation.

4 Bonus - Cart-Pole Swing-Up with iLQR

4.1

First, we will linearize the system for a general timestep t. The non-linear dynamics:

$$(M+m)\ddot{x} - ml\ddot{\theta}\cos\theta + ml\dot{\theta}^2\sin\theta = F$$
$$l\ddot{\theta} - g\sin\theta = \ddot{x}\cos\theta$$

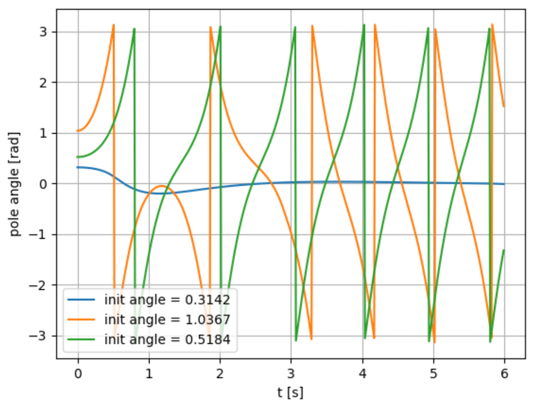

Figure 3: Plotting of the actual pole angle with different initial angles with limited force

Extracting $\ddot{x}$ and $\ddot{\theta}$:

$$\ddot{x} = \frac{u + ml(\sin\theta)\dot{\theta}^2 - mg\cos\theta\sin\theta}{M + m - m\cos^2\theta}$$
$$\ddot{\theta} = \frac{u\cos\theta - (M+m)g\sin\theta + ml(\cos\theta\sin\theta)\dot{\theta}^2}{ml\cos^2\theta - (M+m)l}$$

linearization around $x_t, \dot{x}_t, \theta_t, \dot{\theta}_t$ and u_t:

$$\begin{aligned}
\ddot{x} \approx\ & \frac{u_t + ml(\sin\theta_t)\dot{\theta}_t^{\ 2} - mg\cos\theta_t\sin\theta_t}{M + m - m\cos^2\theta_t} \\
& + \left[\frac{gm\sin^2(\theta_t) - gm\cos^2(\theta_t) + \mathrm{lm}\,\dot{\theta}_t^{\ 2}\cos(\theta_t)}{-m\cos^2(\theta_t) + m + M} - \frac{2m\sin(\theta_t)\cos(\theta_t)\left(-gm\sin(\theta_t)\cos(\theta_t) + \mathrm{lm}\,\dot{\theta}_t^{\ 2}\sin(\theta_t) + u_t\right)}{\left(-m\cos^2(\theta_t) + m + M\right)^2}\right][\theta - \theta_t] \\
& + \left[\frac{2l\cdot m\dot{\theta}_t\sin(\theta_t)}{-m\cos^2(\theta_t) + m + M}\right]\left[\dot{\theta} - \dot{\theta}_t\right] \\
& + \left[\frac{1}{-m\cos^2(\theta_t) + m + M}\right][u - u_t]
\end{aligned}$$

$$\begin{aligned}
\ddot{\theta} \approx\ & \frac{u_t\cos\theta_t - (M+m)g\sin\theta_t + ml(\cos\theta_t\sin\theta_t)\dot{\theta}_t^{\ 2}}{ml\cos^2\theta_t - (M+m)l} \\
& + [\frac{-\dot{\theta}_t^{\ 2} lm\sin^2\theta_t + \dot{\theta}_t^{\ 2} lm\cos^2\theta_t - g(m+M)\cos\theta_t - u_t\sin\theta_t}{lm\cos^2\theta_t - l(m+M)} \\
& + \frac{2lm\sin\theta_t\cos\theta_t\left(\dot{\theta}_t^{\ 2} lm\sin\theta_t\cos\theta_t - g(m+M)\sin\theta_t + u_t\cos\theta_t\right)}{\left(lm\cos^2\theta_t - l(m+M)\right)^2}]\,[\theta - \theta_t] \\
& + \left[-\frac{2m\dot{\theta}_t\sin\theta_t\cos\theta_t}{-m\cos^2\theta_t + m + M}\right]\left[\dot{\theta} - \dot{\theta}_t\right] \\
& + \left[\frac{\cos\theta_t}{lm\cos^2\theta_t - l(m+M)}\right][u - u_t]
\end{aligned}$$

In a matrix form (We will not write the matrices explicitly for obvious reasons):

$$\dot{\mathbf{x}}_{\mathbf{t}} = \bar{A}_t\mathbf{x}_{\mathbf{t}} + \bar{B}_t u_t + \bar{D}_t \qquad (6)$$

And going from continuous time to discreet time:

$$\mathbf{x_{t+1}} = \mathbf{x_t} + dt \cdot \dot{\mathbf{x_t}} = \mathbf{x_t} + dt \cdot \left[\bar{A}_t \mathbf{x_t} + \bar{B}_t u_t + \bar{D}_t\right] = A_t \cdot \mathbf{x_t} + B_t \cdot u_t + D_t \tag{7}$$

Where:

$$\begin{aligned} A_t &= I + dt \cdot \bar{A}_t \\ B_t &= dt \cdot \bar{B}_t \\ D_t &= dt \cdot \bar{D}_t \end{aligned}$$

We gave up after trying to find the optimal Solution for K and V :(