1. Blackjack:

a. The Markov decision process is defined as follows:

States space: each state represent the (X,Y) where x is the sum of cards holds by the player and y is the exposed card of the dealer. After the dealer give cards we start from the appropriate state. The terminal state is where also the player and the dealer do not want to play anymore.

The size of the state space is $|S| = 200$ because there are 10 options for y (2-11) and 19 options for x (4-22) and another 10 states for natural win.

Action space: for each state the possible actions are $a \in \{hits, sticks\}$.

Transform function: if the player choose "sticks" the MDP transform stick with the current (X,Y) and this is the dealer turn.

but, if the player choose "hits" then the MDP transform to $(X+\delta,Y)$ where:

$$\delta = \begin{cases} 2,3,4,5,6,7,8,9,11 \text{ w.p } \frac{4}{52} = \frac{1}{13} \\ 10 \text{ w.p } \frac{16}{52} = \frac{4}{13} \end{cases}$$

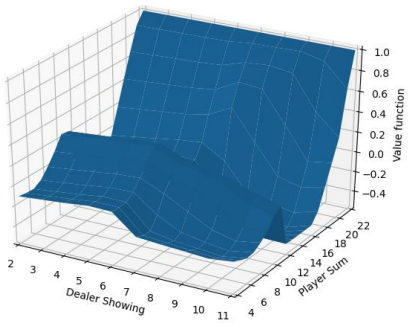
Rewards: the player gets reward only when the game ends: if the player starts the game when $X = 21$ it is "neutral" and gets 1 if $X > 21$ then loses and gets -1. For winning using hits gets 1 for a draw gets 0 and for loss gets -1.

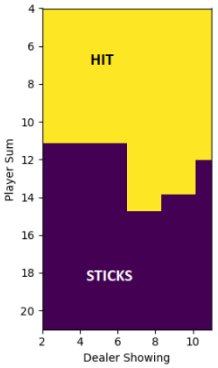
The objective is to maximize the player's reward.

The bellman equation is:

$$v^*(x,y) = \max_{a \in \{hit, stick\}} \left\{ r(x,y) + v^*\left(f(x,y,a)\right) \right\} = \max \left\{ \underset{a=stick}{r(x,y)}, \underset{a=hits}{v^*(x+\delta,y)} \right\}$$

b.



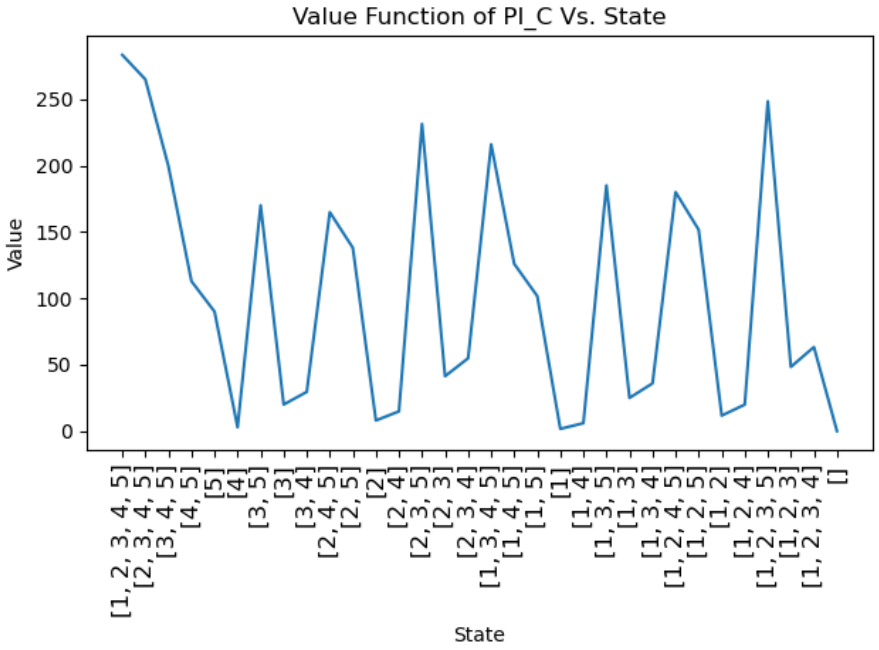


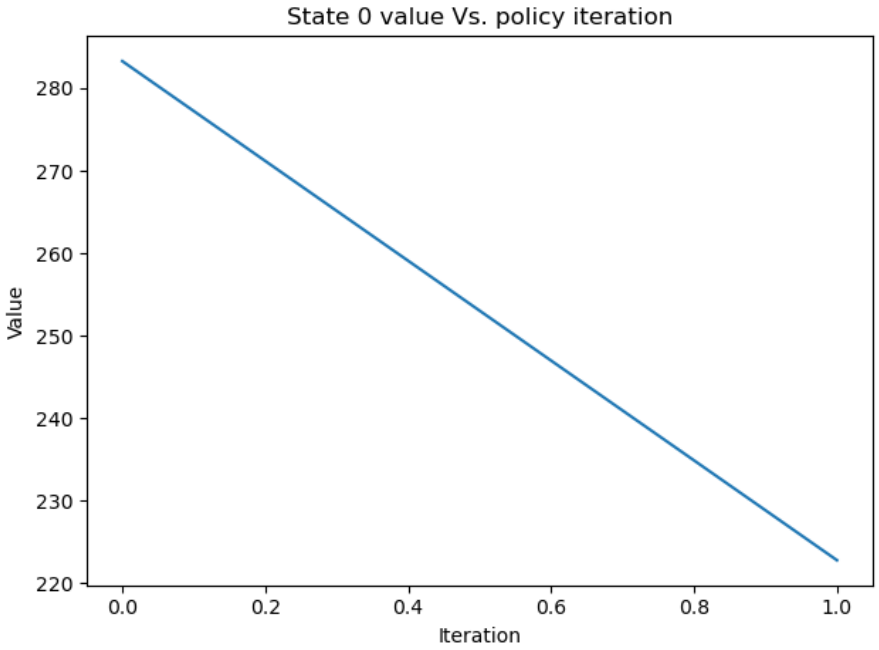
2. **The $c\mu$ revisited:**

a. The number of states of the problem is: 2^N because for each job there are two options or it already done or it is in the system.
The number different actions is N one for each different job. Not all are available in all states, for example for the initial state, that includes all the jobs, all are available but after one of the jobs done only N-1 are available, and so on.

b. ***Code is available at the end of document.***

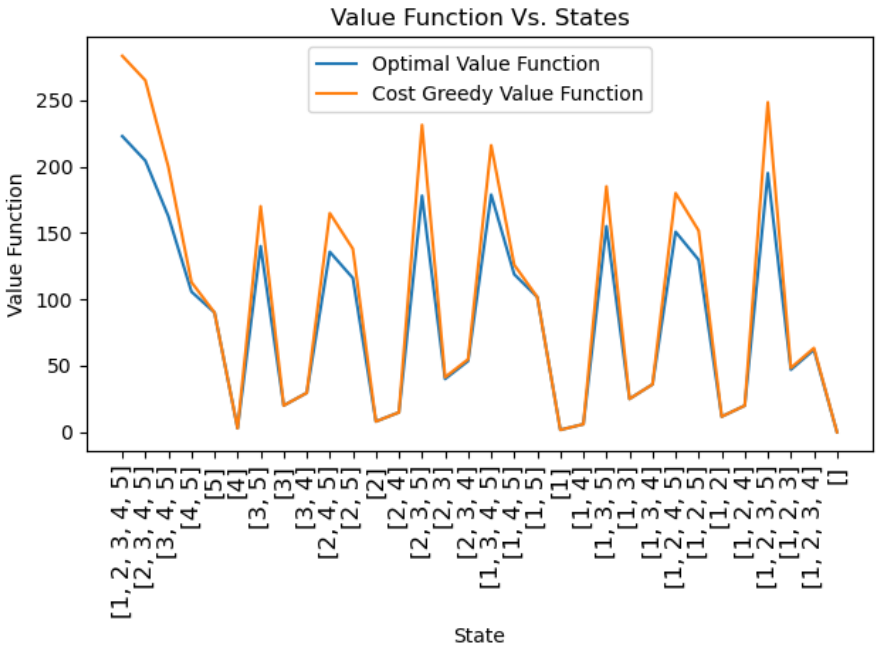
c.





d. After one iteration we get the optimal policy and one more step to make sure the policy is not changed from last one, means one iteration.

e.

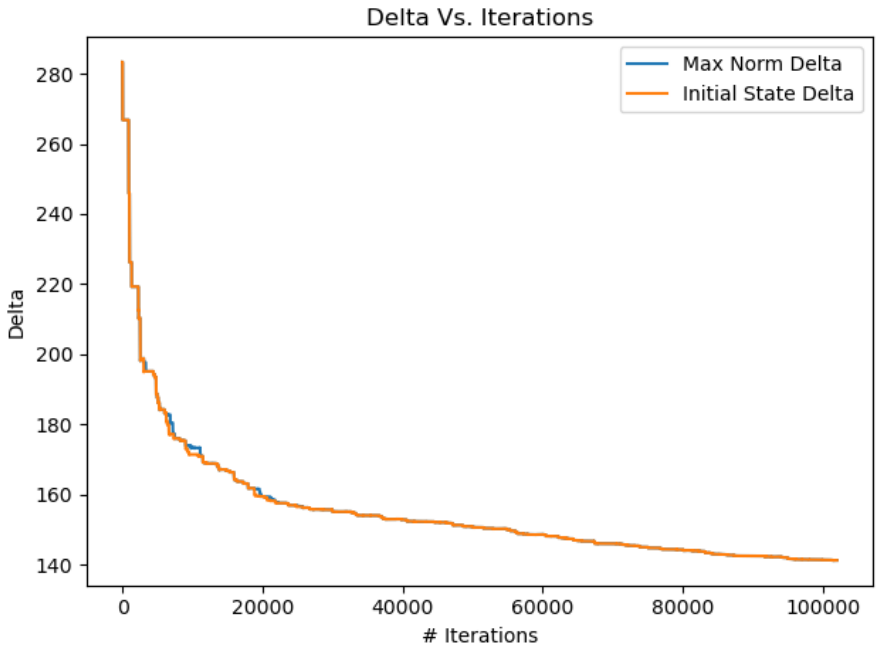


As we can see from the graph for all the states the optimal policy values are smaller than the greedy one, for state with only on job left the value is equal.

f. ***Code is available at the end of document.***

g. Because for this deterministic policy we absolutely not going to visit all the states in the MDP we decided to choose randomly initial state for every episode, that we can convergence the value function not only for reachable states from the state that includes all jobs.

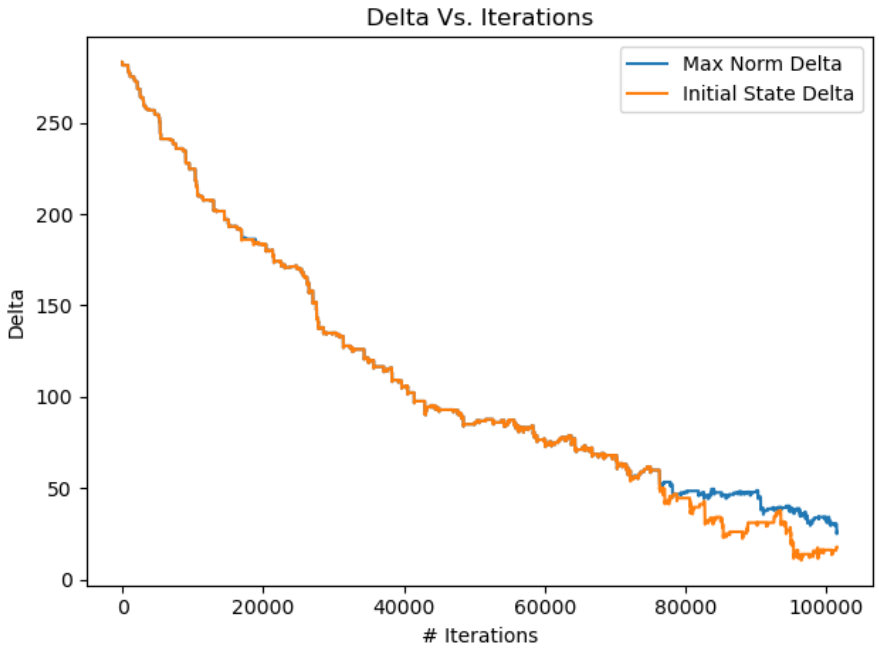
For $\alpha_n = \frac{1}{\#visits}$:



The motivation for that step size is to make it decrease every iteration in order the value function of the TD(0) to convergence the real value function,

it reflected in the result by that all the states convergence together.

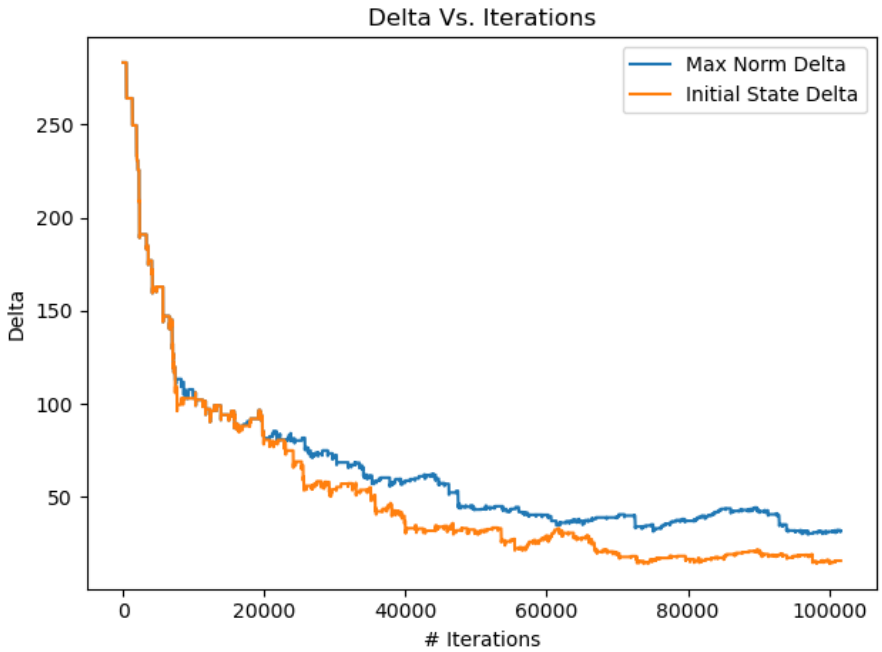
For $\alpha_n = 0.01$:



The motivation for that step size is to make the convergence faster, but as we can see the convergence is not uniform for all states.

The motivation for a constant step size is

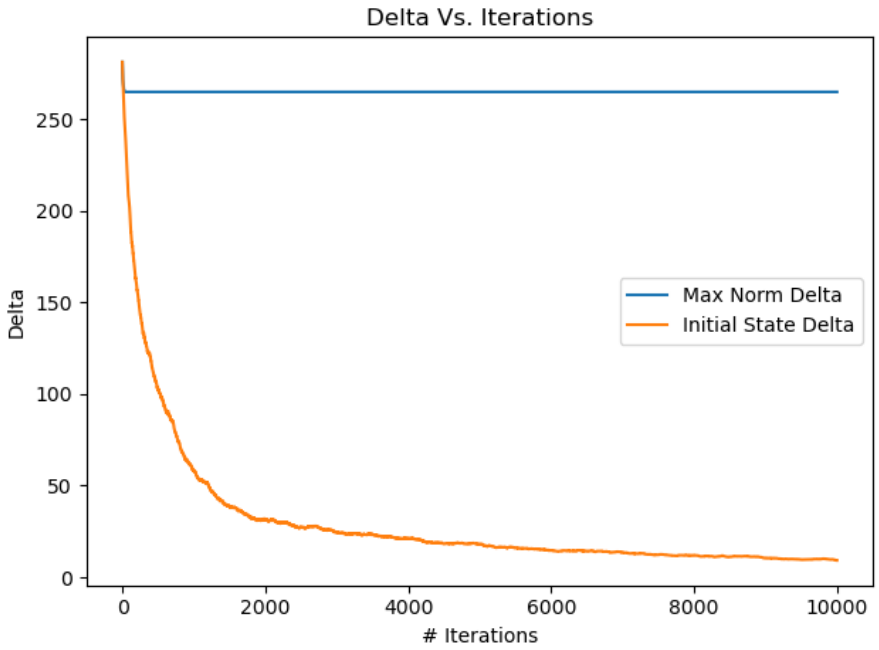
For $\alpha_n = \frac{10}{100+\#visits}$:



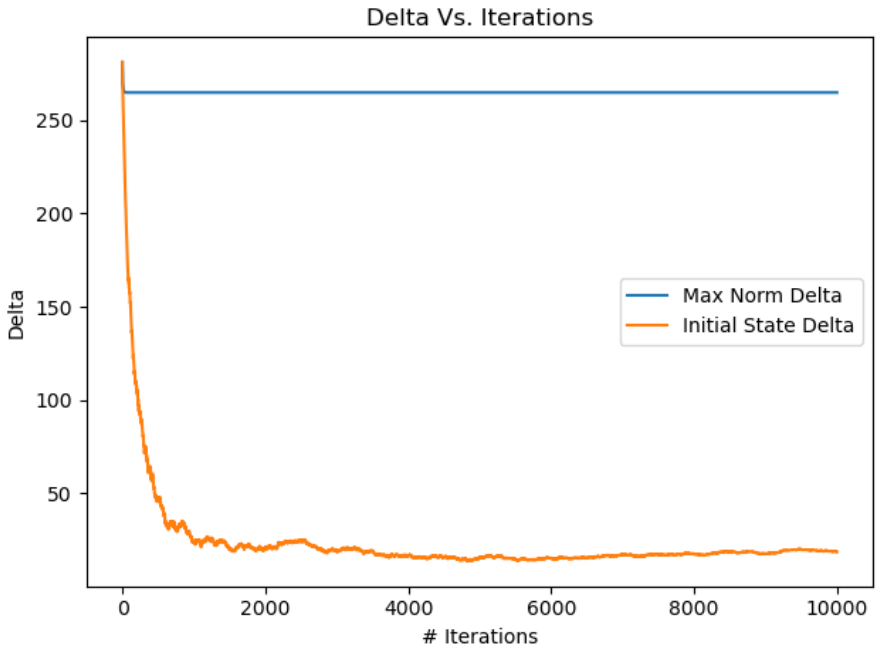
The motivation for that step size is to make the convergence faster and to make the estimated values even closer to the real values.

h. For all our runs we choose $\alpha_n = \frac{10}{100+\#visits}$.

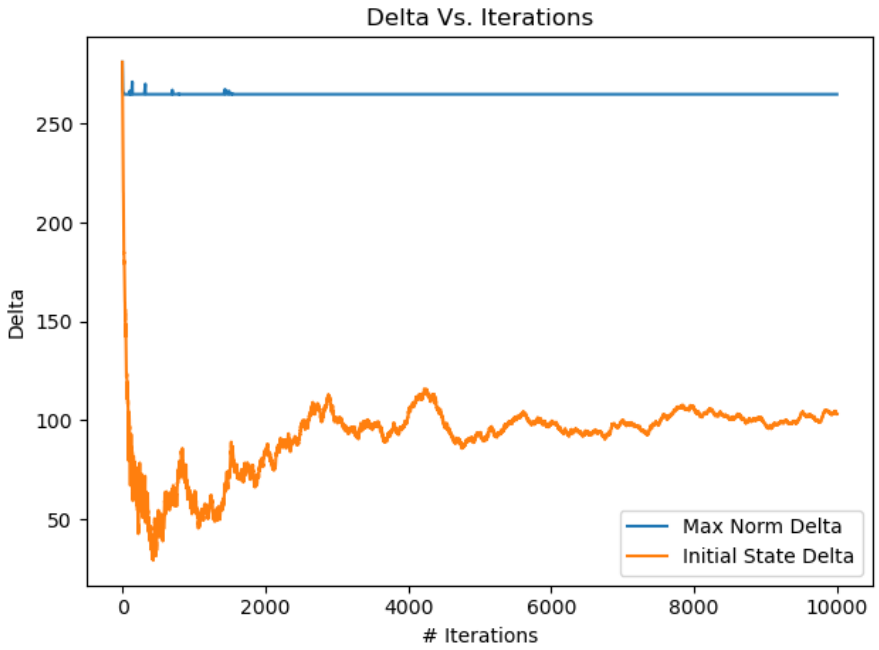
For $\lambda = 0$:



For $\lambda = 0.5$:

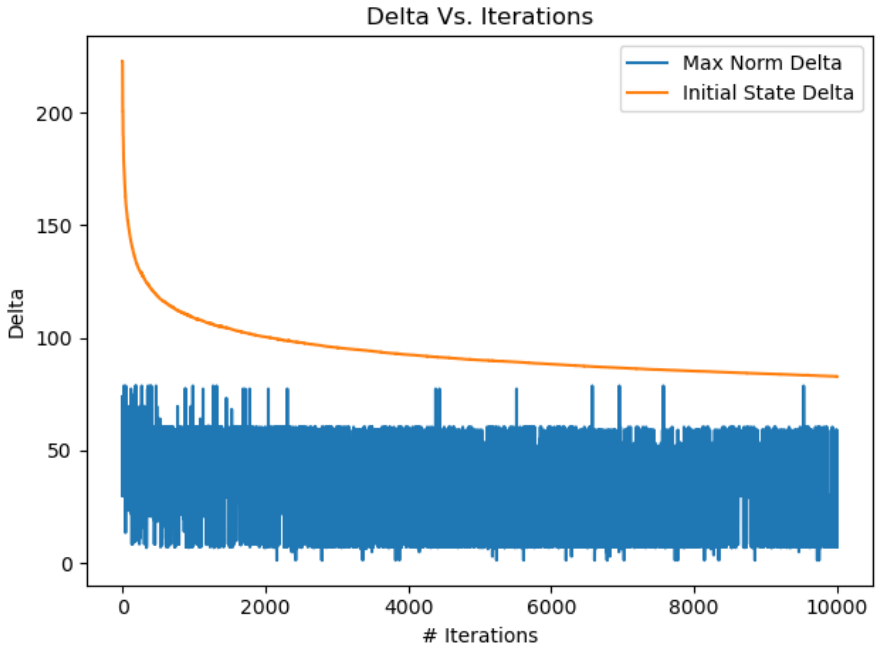


For $\lambda = 0.9$:

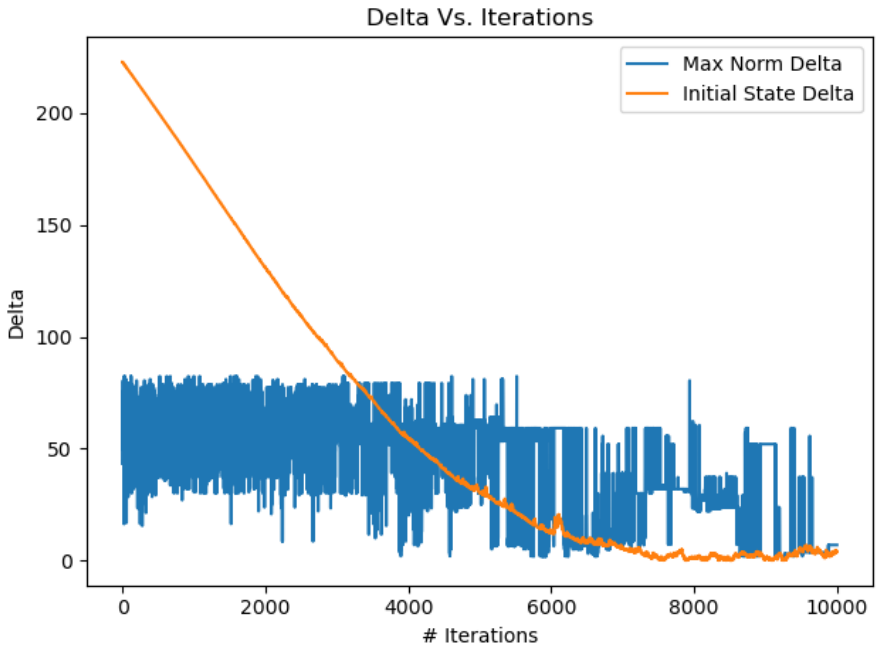


Notice that we see convergence only for initial state because using the propose policy make the algorithm not to evaluate all states.

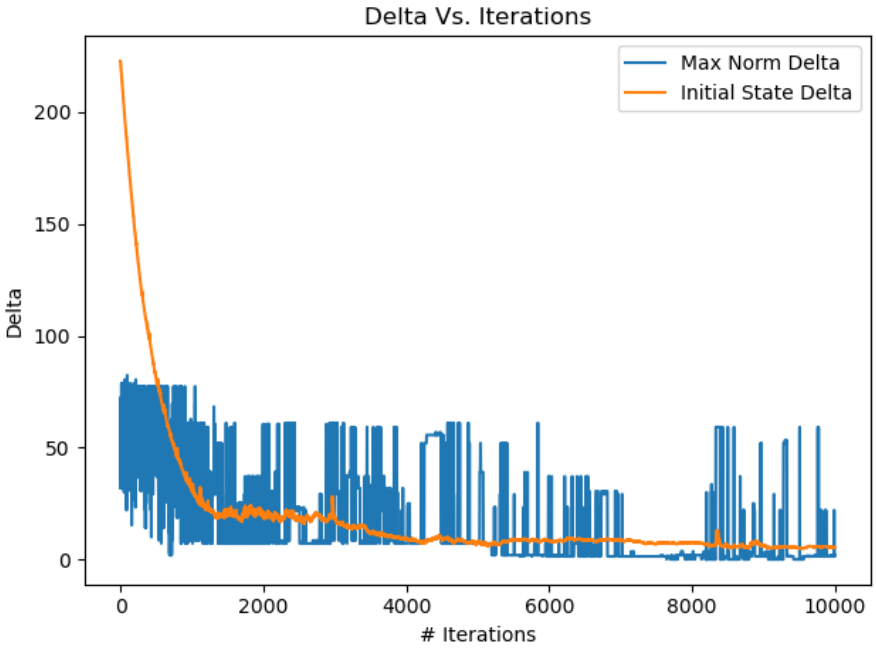
i. Q-learning with $\alpha_n = \frac{10}{100+\#visits}$ and $\varepsilon = 0.1$:

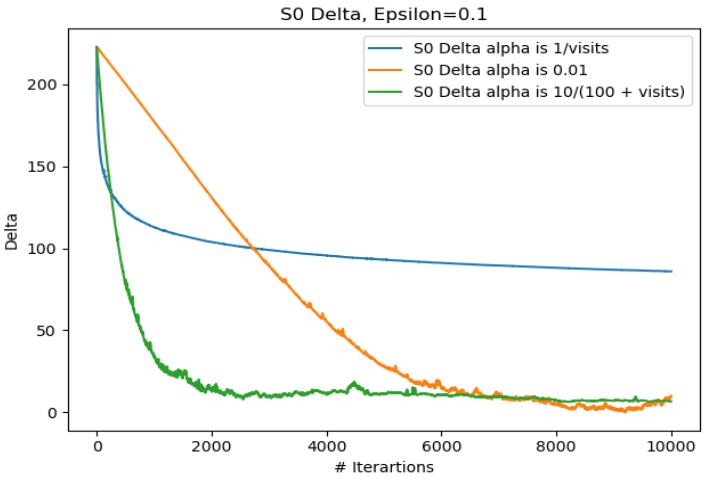


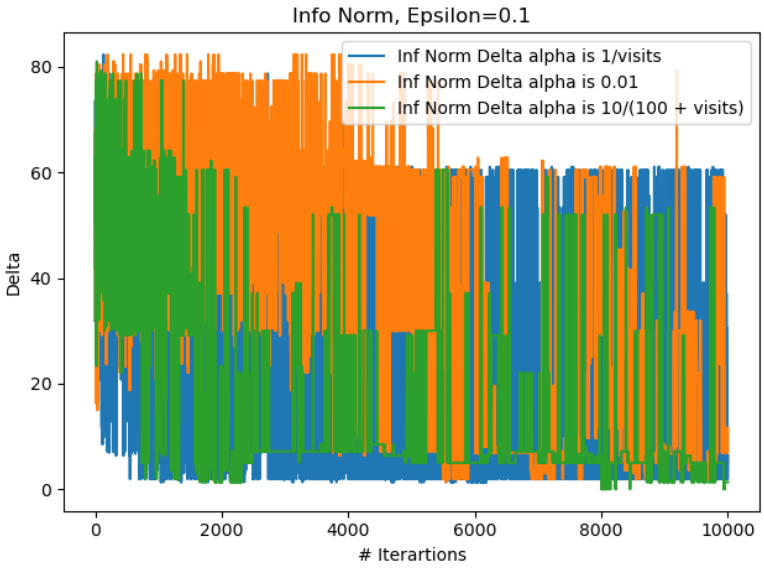
Q-learning with $\alpha_n = 0.01$ and $\varepsilon = 0.1$:



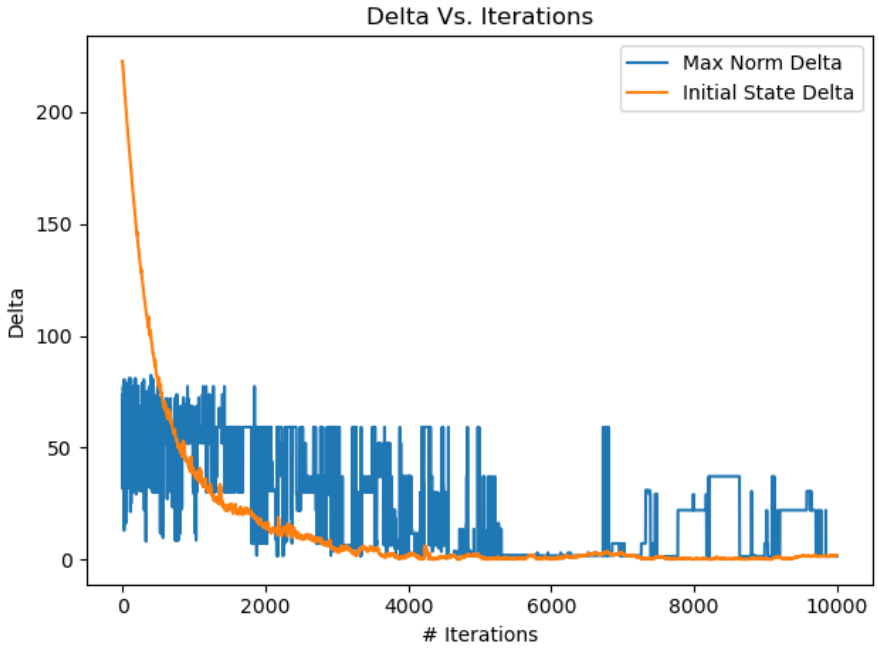
Q-learning with $\alpha_n = \frac{10}{100+\#visits}$ and $\varepsilon = 0.1$:

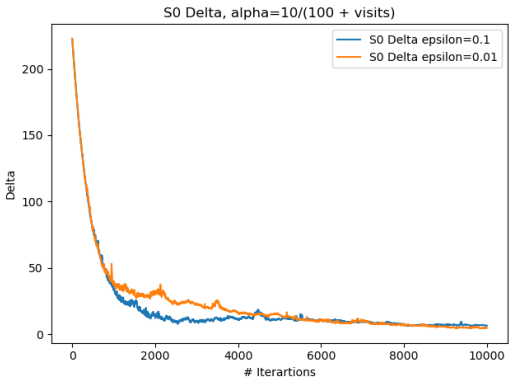

Delta Vs. Iterations
Max Norm Delta
Initial State Delta
Delta
Iterations
0
50
100
150
200
0
2000
4000
6000
8000
10000

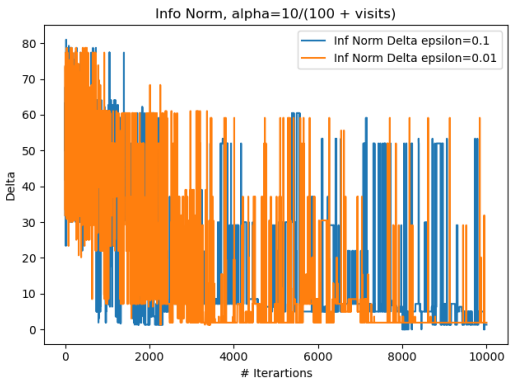

S0 Delta, Epsilon=0.1
S0 Delta alpha is 1/visits
S0 Delta alpha is 0.01
S0 Delta alpha is 10/(100 + visits)
Delta
Iterartions
0
50
100
150
200
0
2000
4000
6000
8000
10000


Info Norm, Epsilon=0.1
Inf Norm Delta alpha is 1/visits
Inf Norm Delta alpha is 0.01
Inf Norm Delta alpha is 10/(100 + visits)
Delta
Iterartions
0
20
40
60
80
0
2000
4000
6000
8000
10000

j. Q-learning with $\alpha_n = \frac{10}{100+\#visits}$ and $\varepsilon = 0.01$:







Because here we are using smaller epsilon that means the algorithm is highly prefer the current best action for each step, than to explore the environment therefore it changes paths rarely and the best action not necessarily updates frequently.