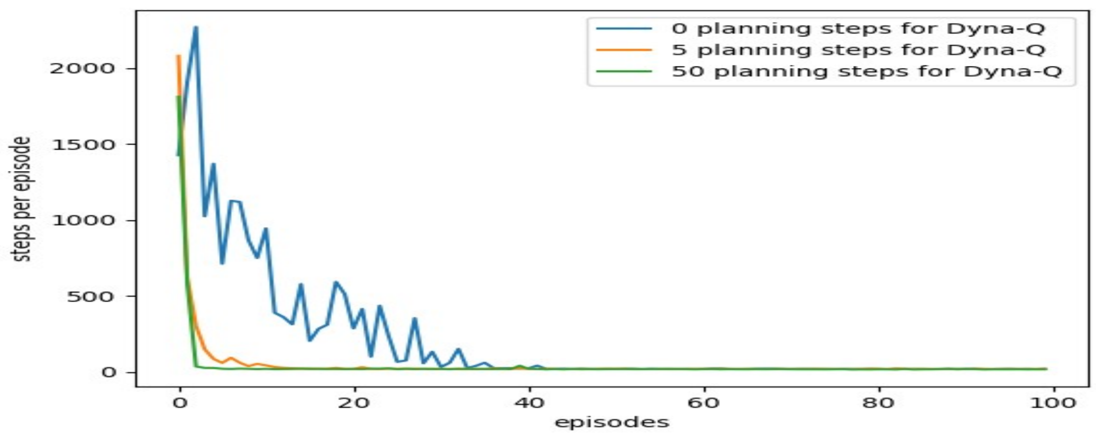
TheoryAnswers(Divya Saxena 1001773376)

Question 1:

a) **Dyna-Q**
Grid world: 15 * 25
Start State: [2,0]
Goal State: [[0,8]]
Obstacles: [[1, 2], [2, 2], [3, 2], [0, 7], [1, 7], [2, 7], [4, 5]]

b) **Dyna-Q Learner**
Following graph shows the steps per episode vs episodes for Dyna-Q

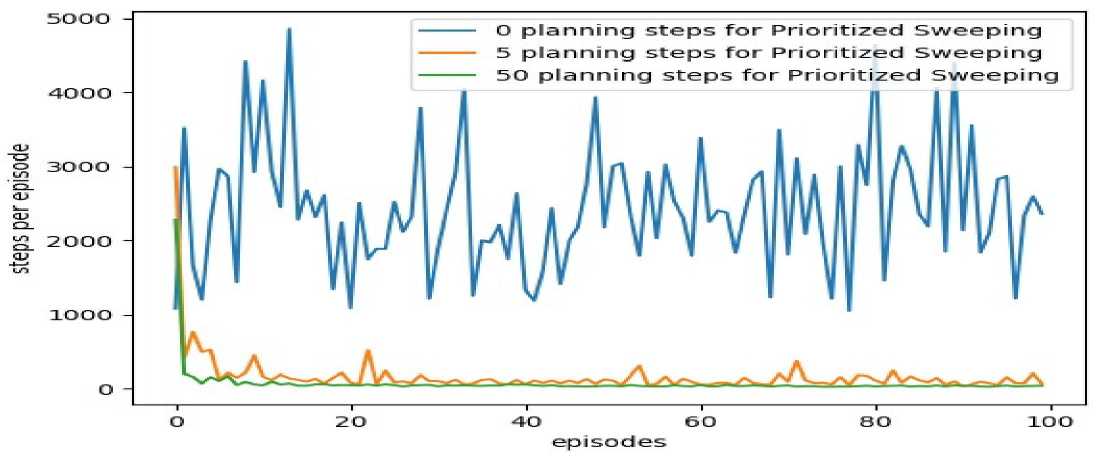


c) **Planning steps** taken for Dyna-Q are 0,5,50 respectively.
By seeing above graph, it can be observed that the model performs better when planning steps has been increased from 0 to 50. The model performs very badly for plannning step 0. It learns very slowly. While increasing planning step to 5, it learns significantly early. And when planning steps increased to 50, the model learnt very early.

Question 2:

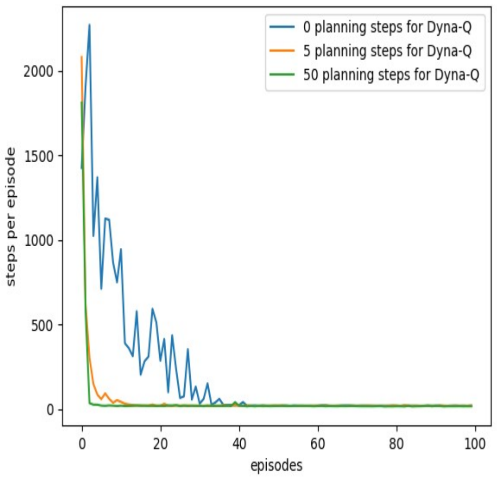
a) **Bellman error(i.e. prioritized sweeping).**
Grid world: 15 * 25
Start State: [2,0]
Goal State: [[0,8]]
Obstacles: [[1, 2], [2, 2], [3, 2], [0, 7], [1, 7], [2, 7], [4, 5]]

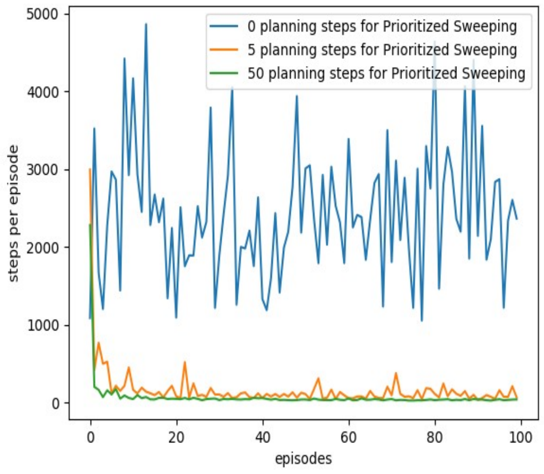
b) **Prioritized Sweeping Learner**
Above graph shows the steps per episode vs episodes for Prioritized Sweeping. Planning steps taken are 0,5,50 respectively like Dyna-Q to compare results.



TheoryAnswers(Divya Saxena 1001773376)

Comparison Dyna-Q and Priorotized Sweeping.





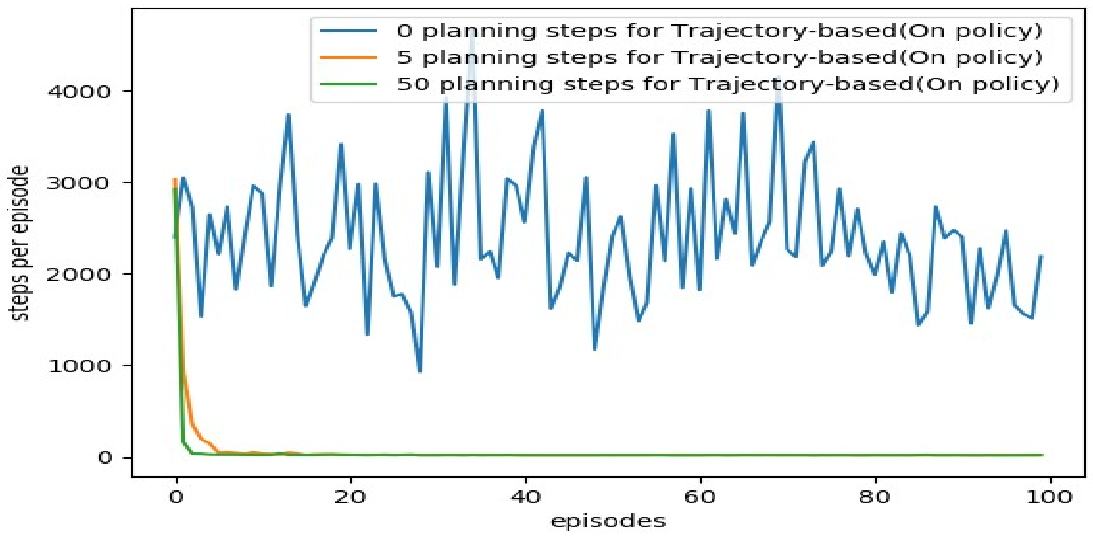
By seeing graphs above, it can be concluded that for planning step 0 prioritized sweeping performs really bad as compared to Dyna-Q. Prioritized learns better for planning step 5 and 50 as it works backwards from states whose values have just changed. It maintain a queue of state-action pairs whose values would change a lot if backed up, prioritized by the size of the change

c) **Trajectory Sampling:**

Grid world: 15 * 25
Start State: [2,0]
Goal State: [[0,8]]
Obstacles: [[1, 2], [2, 2], [3, 2], [0, 7], [1, 7], [2, 7], [4, 5]]

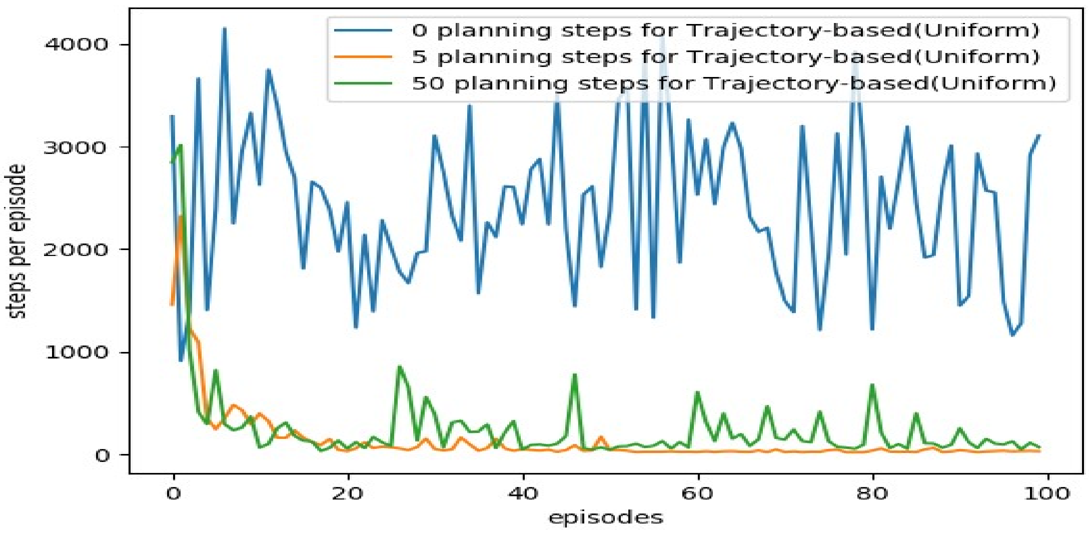
To understand better, I have implemented both Uniform state-action selection and on-policy state-action selection. Planning steps taken are 0,5,50 respectively.
On-policy: backed up along simulated trajectories
Uniform: cycled through all state-action pairs

The graphs are as below:
On-Policy:

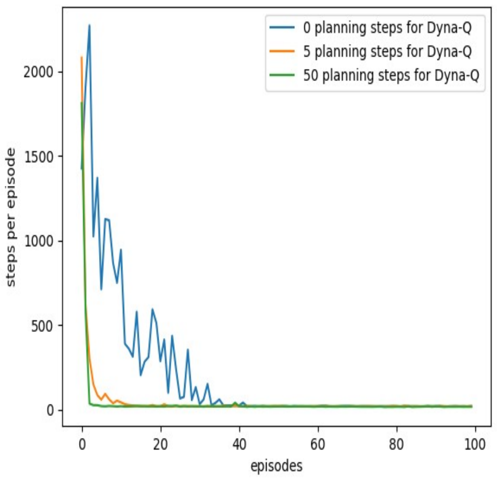


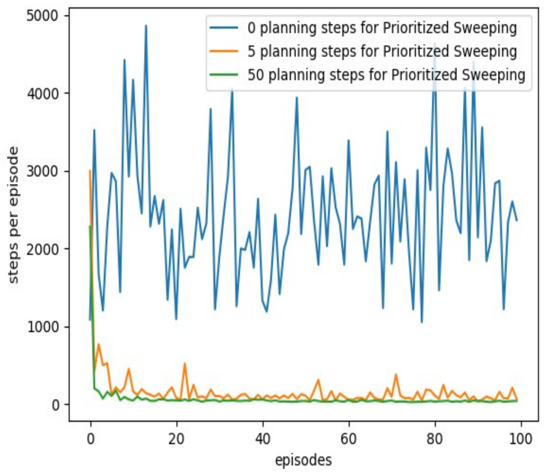
TheoryAnswers(Divya Saxena 1001773376)

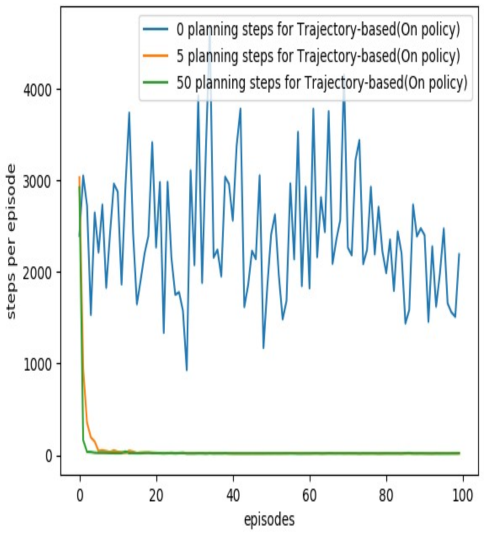
Uniform:

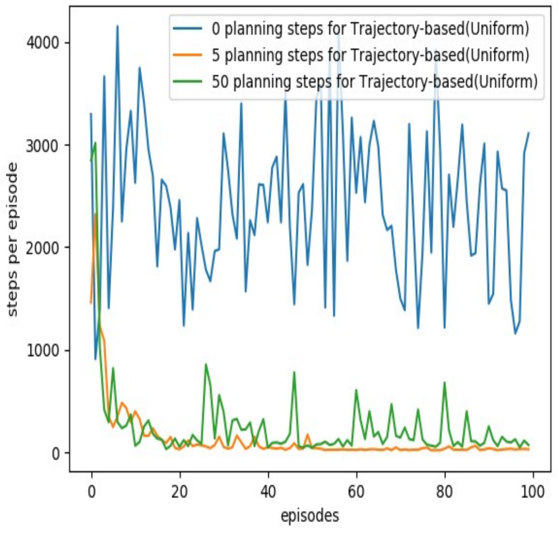


Comparisons among Dyna-Q, Prioritzed Sweeping and Trajectory based(on-policy,Uniform) algorithms: Planning steps taken are 0,5,50 respectively.









TheoryAnswers(Divya Saxena 1001773376)

By seeing the graphs, it can be concluded that in all above algorithms Dyna-Q learns the model more efficiently with lesser number of steps per episodes. When on-policy distribution is selected for trajectory based , it performs better than Uniform distribution as on-policy distribution can cause vast uninteresting parts of the state space to be (usefully) ignored. Prioritized learns better for planning step 5 and 50 as it works backwards from states whose values have just changed. It maintain a queue of state-action pairs whose values would change a lot if backed up, prioritized by the size of the change