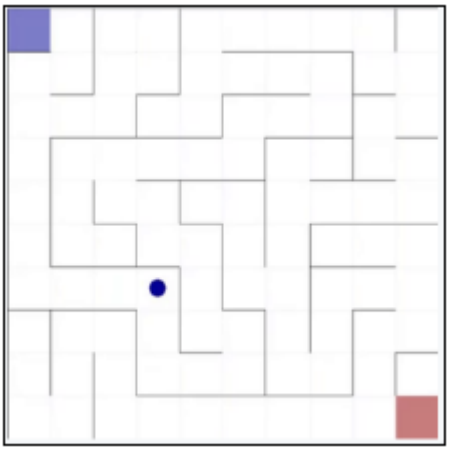
Reinforcement Learning – Mid Semester Assignment

Nadav Shaked 312494925, Michael Glustein 203929500

<u>The Problem:</u>

Given is a maze of size NxN, we must train an agent (blue dot) on the given environment to find the shortest path from the starting point (blue square) to the ending point (red square).



Action Space: [‘N’, ‘S’, ‘E’, ‘W’]

Stochastic policy with probability of 0.9.

Reward: $-\frac{0.1}{num\ of\ states}$ for each step, 1 for final state. (initially)

<u>Solution:</u>

We used two algorithms (with a few variants) to solve the problem.

In the first algorithm “Monte Carlo”, we run the agent a complete episode based on a policy and after every episode we evaluate and update the policy. We continue to iterate until convergence. We implemented this algorithm with the first visit method (that estimates the values as the average returns of the first visit to the state).

The second algorithm “Q-learning” we evaluate the policy throughout the run after each step. Each step is evaluated based on the expected value of the following step. This lets us improve during the iteration.

When choosing our action, we wish to maximize our value. For evaluation, we implemented both a state value policy and an action-state value policy.

In addition, we added the option of “exploration”, where we could start our iterations from random states in the environment. This allows us to explore states we may not have reached.

We also implemented another method to ensure we continue to explore, where we chose our action based on our policy with a probability of $(1-\epsilon)$, we implemented the ϵ-soft policy with $\pi(a|s) \geq \epsilon/|A|$ for all actions. We added an option for the epsilon to decrease as the number of iterations rises, that way as we converge, we could become “greedier” and trust our policy. We allowed to combine the two epsilon-policies.

Reinforcement Learning – Mid Semester Assignment

Nadav Shaked 312494925, Michael Glustein 203929500

We also implemented the stochastic policy as defined in the problem.

<u>Results:</u>

<u>Parameter explanation:</u>

For different policies we used some parameters to help get improved results.

Epsilon – implementing epsilon policies (in addition we added a $is\_epsilon\_decrease$ boolean value for decreasing epsilon in later episodes)

Gamma – "weight" of future value.

Alpha – for Q-learning algorithm only, alpha is our "learning rate", this constant gives "weight" to the latest episode.

Stochastic Probability – probability the agent will move according to policy.

Exploration (bool) – if true, we start the iteration at a random state.

We used a constant value to limit the maximum number of steps in each iteration. The reason is that sometimes the agent finds himself in a "loop" and doesn't improve, in this case, it is best that we continue.

For the 30x30 maze (only) we added the possibility to adjust rewards of two states.

We ran multiple runs with different algorithms/parameters to achieve better results.

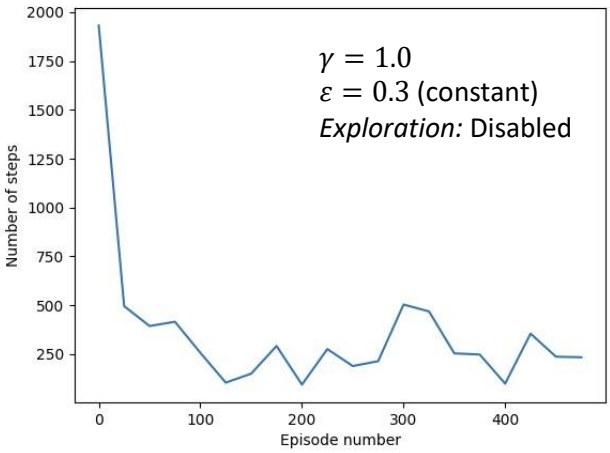
Reinforcement Learning – Mid Semester Assignment

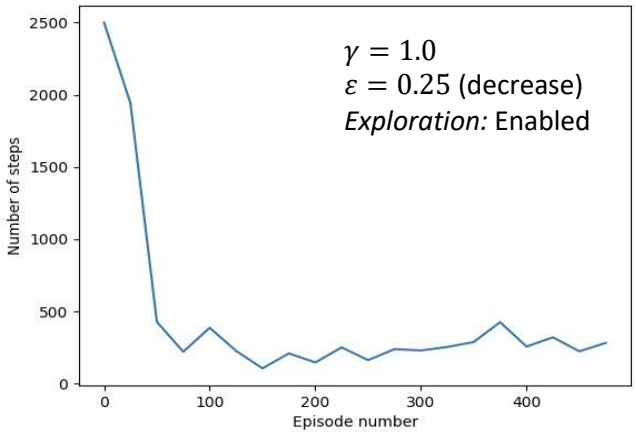
Nadav Shaked 312494925, Michael Glustein 203929500

Maze 15x15

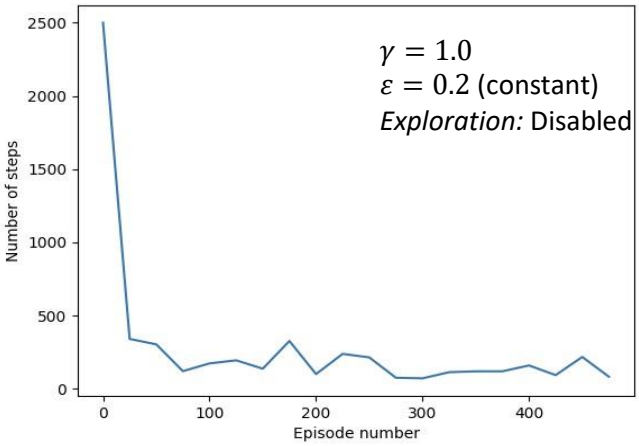
Monte Carlo Algorithm

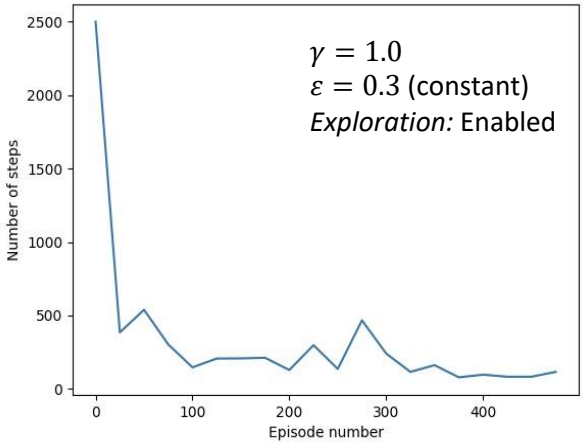
State Values Method





Action-State Values Policy



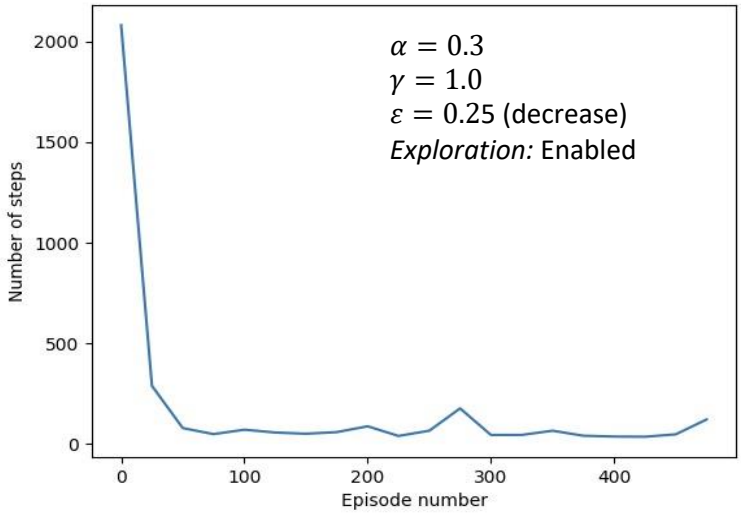


Reinforcement Learning – Mid Semester Assignment

Nadav Shaked 312494925, Michael Glustein 203929500

Q-learning Algorithm

Action-State Values Policy



Rewards

We ran Q-learning with the same values while adjusting the rewards (of all states) to inspect the changes.

We reached best results when giving a value of -0.0238 to the non-terminal states. We reached on average 32.08 steps.

Maze 30x30

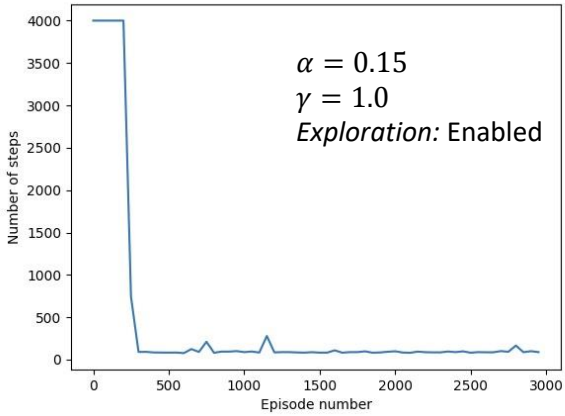
We placed two extra "rewards" in strategic locations.

At location $(18, 10)$ we placed a reward of $-\frac{0.1}{4|A|}$ and at location $(17,26)$ we placed a reward of value $-\frac{0.1}{2|A|}$. The rewards are placed strategically at half-way and ¾-way, to lead the agent to the end. The rewards have a negative value, not to tempt the agent to stay put, but have a better reward than the other states, to encourage the agent to reach them.

We found the best results were achieved using Q-Learning algorithm. With the following parameters, we achieved satisfying results.

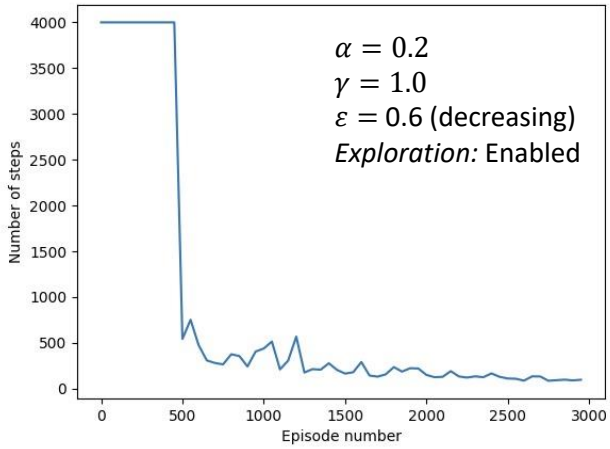
Reinforcement Learning – Mid Semester Assignment

Nadav Shaked 312494925, Michael Glustein 203929500



We had another idea of placing an exaggerated reward of the final step, we gave it a value of 200.

We gave it a "high" epsilon that decreases over episodes, so the agent explores the map and could reach the final state "by accident" quicker. As we update our policy, we will put more emphasis on the policy, by lowering the epsilon.



<u>Summary</u>

As expected, the Q-learning algorithm performed better than the Monte Carlo algorithm, it also converged to less steps and also converged quicker, in less episodes. The algorithm was also more stable and was more consistent with its results.

Better results were achieved when using the state-action evaluation, rather than just the state. The state-action takes into consideration also the best action to take in each state, thus giving the agent more information.

Reinforcement Learning – Mid Semester Assignment

Nadav Shaked 312494925, Michael Glustein 203929500

We found that using “epsilon” policies help explore, but it is best to have the epsilon decrease with time, as we eventually want to put more emphasis on our policy. Beginning in random states also helped us explore the environment.

For Q-learning algorithm we found results were best with $\alpha \approx 0.2$, meaning we give high weight to the previous step.

We saw that adding strategic rewards along the way to encourage the agent is helpful (but misplaced rewards could harm) and that adjusting the reward values has implications to the results.

Overall, the results were aligned with our expectations. However, there was some level of variance between runs, this is in partial due to the stochastic environment.