Reinforcement Learning

Assignment #1 Report

Student name
Pedram Abdolahi Darestani
Zahra Jabari

Student IDs
202383919
202291677

Date
June 13th , 2024

Introduction

This assignment was aimed at designing and evaluating a reinforcement learning agent on two different k-armed bandit problems: stationary and non-stationary rewards. The stationary type is explored in the first part of the assignment and the non-stationary type in the second. The details of the implementation and the link to the repository containing the assignment codes can be found in the appendix.

Part 1

As stated above, this part is dedicated to the k-armed bandit problem with a stationary reward system. For this type, multiple types of agents were used and their details and performances are shown below.

Problem specifications

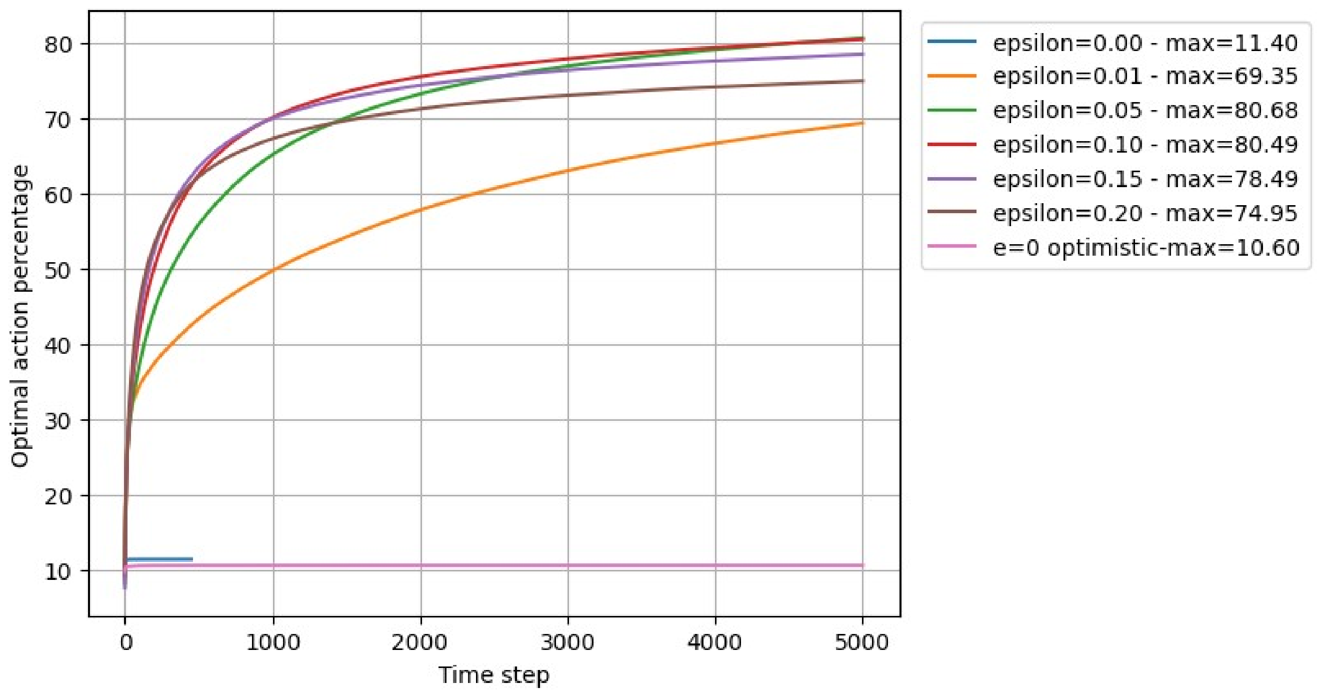
- Number of arms (k) = 10
- Stationary rewards
- Means of the arm rewards are drawn from N(0,1)
- The reward of action i is randomly drawn from N(mu_i, 1) where mu_i is the mean of that action's reward
- Incremental update of rewards estimation
- Agent evaluation based on the aggregate of 1000 different initializations
- Maximum time steps for each agent decided through trial and error (5000)

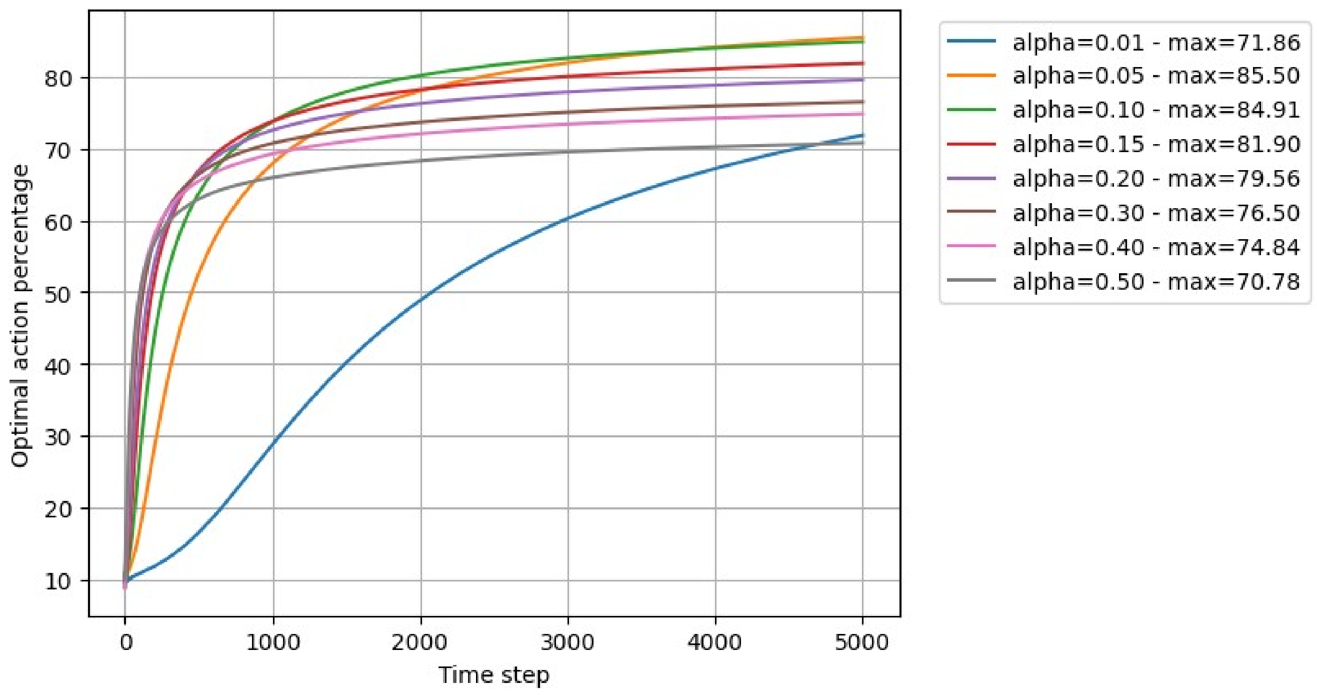
Agent specifications

1. Greedy, non-optimistic initial values
   - initial action reward values = 0
2. Epsilon greedy, non-optimistic initial values
   - epsilons tested = [0.01, 0.05, 0.10, 0.15, 0.20]
3. Greedy, optimistic initial values
   - initial action reward values = 10
   - the value of 10 is selected since it has an acceptable p-value of 0.0001 for the N(0,1) distribution. As a result, the agent will highly probably face much higher rewards than the mean of rewards for any action, in the beginning.
4. Gradient bandit algorithm
   - alphas tested = [0.01, 0.05, 0.10, 0.15, 0.20, 0.30, 0.40, 0.50]

Results

The "Percentage of Optimal Actions Taken vs Time Step" plots are drawn for each of the agents and their hyper parameters. The epsilon greedy and optimistic agents are shown in the upper figure and gradient agents are shown in the lower one.





By analyzing the plots the following conclusions can be arrived at:

- The greedy non-optimistic agent fails to learn much from the environment. It yields low average reward throughout all time steps and a small fraction of its actions were optimal.
- The epsilon-greedy agent outperforms all the other types of agents. The epsilon that provided the highest terminal average reward and ratio of optimal actions was found to be 0.05. Lower and higher values worsened the performance of the

agent. The agent with epsilon = 0.05 managed to take the optimal action about 80.68% of the time. The second best value of epsilon was 0.10 and yielded just slightly lower percentage of 80.49%.
    - The agent’s performance ranking for different the epsilon values is as below:
        - 0.05 > 0.10 > 0.15 > 0.20 > 0.01
    - Note that comparison of % of optimal actions is more reliable between agents due to the different random initialization of action reward means. An agent can display higher values for average rewards at the terminal state but lower optimal action ratio due to having higher action reward means.
- The greedy optimistic agent initially yielded high average reward values but this average reward plummeted and stayed low for most of the time steps. The optimal action ratio however, was always low and remained low for the entirety of the time steps.
- The gradient agent yielded optimal action percentage of 85.5% at alpha=0.05, which is the highest of all the models. The next best alpha was 0.10 which achieved 84.91% optimal action percentage.
    - The agent’s performance ranking for different the alpha values is as below:
        - 0.05 > 0.10 > 0.15 > 0.20 > 0.30 > 0.40 > 0.01 > 0.50
    - The model with alpha=0.05 learned slower than the one with alpha=0.10 but managed to outperform it after about 4000 time steps.

Final remarks

Four different methods of greedy, epsilon greedy, optimistic greedy, and gradient bandit were compared in this part.

- What did we do to tune each of the methods?

The three methods of epsilon greedy, optimistic greedy, and gradient bandit came with hyper-parameters.

- **Epsilon greedy and gradient bandit:** hyper-parameters are epsilon and alpha. To tune these, multiple different values were tested in order to find their optimal value, and then the hyper-parameter that yielded the best performance was selected for that method.
- **Greedy optimistic:** the hyper-parameter was the initial reward estimations of the agent. This value was selected based on the knowledge that the mean of the rewards for every arm was drawn from N(0,1). Then, rewards for each arm would be drawn from N(mu_i, 1). As a result, the initial reward of 10 is befitting since it is well above the probable action rewards for all of arms.

- Which method performs the best and why?

Between the four given methods, the gradient method was found to be the best performing (*on average,* based on 1000 different initializations). This method, with the optimal hyper-parameter, was able to follow the optimal choice about 85.5% of the time.

The next best method was the epsilon greedy method which only did so at about 80% of the time.
The reason for this observation could be that the gradient method selects actions based on softmax of preferences instead of reward values and epsilons. This allows for higher probability of selection for the next best options and lower probability for the worst options observed so far. This is in contrast to the epsilon greedy method that assigns equal selection probability to all the sub-optimal arms. Furthermore, the gradient method makes use of the idea of *baseline* in order to better guide the preference adjustments. By knowing what the mean achieved reward by the agent has been until a time step, it can better decide if the achieved reward by the current action is in the right direction. Other methods just proceed to update the action value estimates without considering this.

Part 2

This part is dedicated to the k-armed bandit problem with a non-stationary/moving reward system. The problem and agent specifications are given below.

Problem specifications

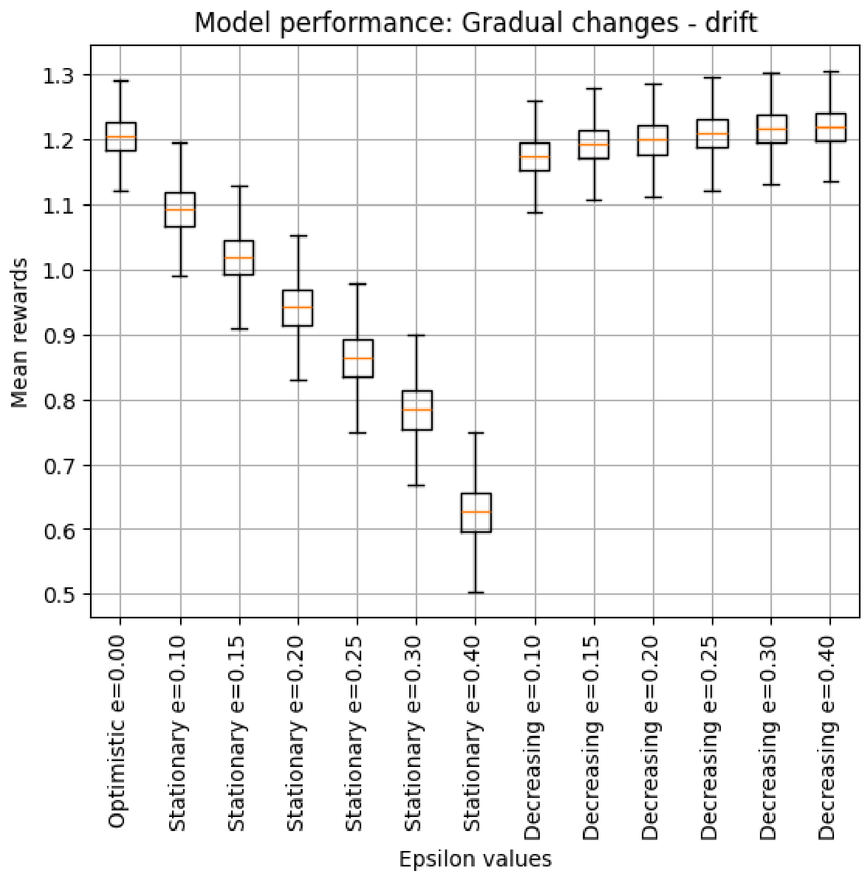
- Number of arms (k) = 10
- Non-stationary / moving rewards
  - drift change: $\mu_t = \mu_{t-1} + \varepsilon_t$ where ε_t is N (0, 0.001^2)
  - mean-reverting change: $\mu_t = \kappa\mu_{t-1} + \varepsilon_t$ where $\kappa = 0.5$ and ε_t is N (0, 0.01^2)
  - abrupt change: permute the means with probability of 0.005
- Means of the arm rewards are drawn from N(0,1)
- The reward of action i is randomly drawn from N(mu_i, 1) where mu_i is the mean of that action's reward
- Incremental update of rewards estimation
- Agent evaluation based on the aggregate of 1000 different runs
  - The initial reward means for the runs are drawn from N(0, 1).
  - The initial reward means are set to be identical across models, while the randomness of mean changes is preserved. Only this way the performance of the models in following the reward movement can be compared. (If every model type starts from a different mean vector, it might have a different experience following the movement, thus, comparison will be inaccurate.)
- Maximum time steps for each agent was set to 20,000

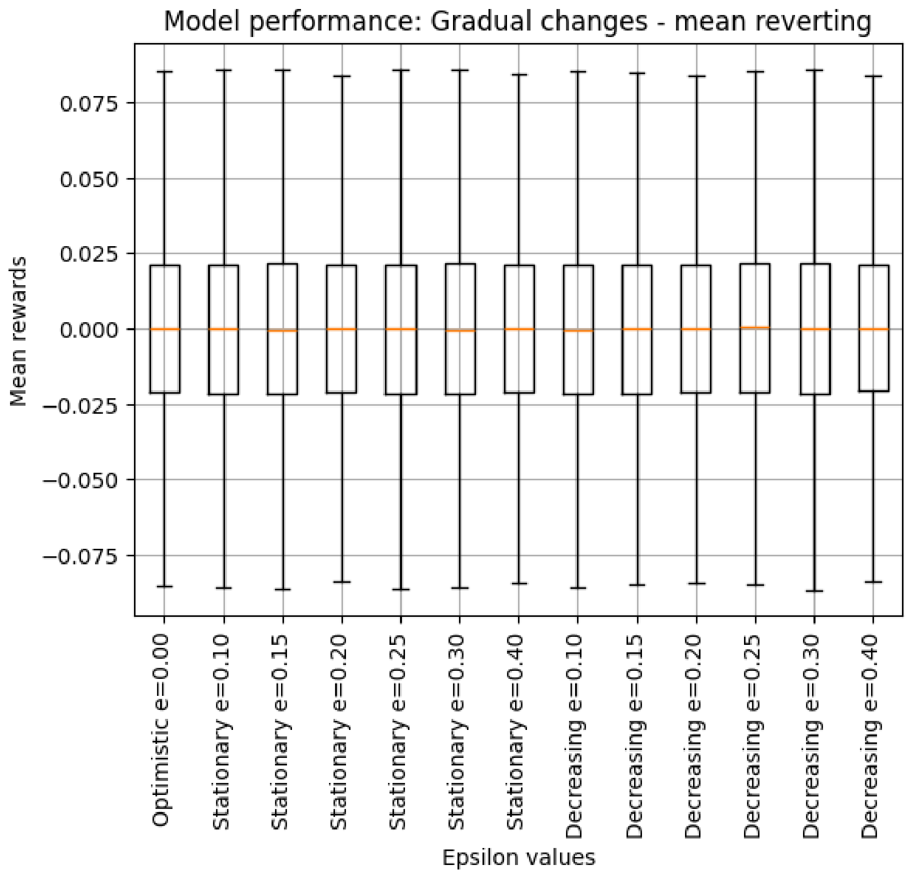
Agent specifications

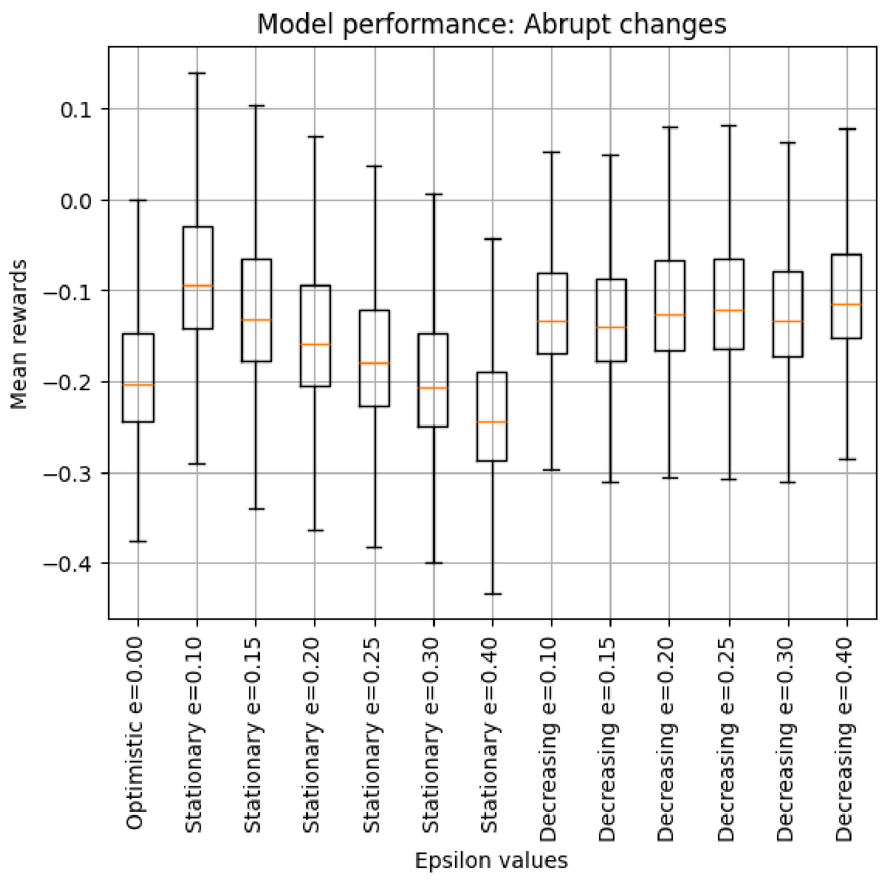
1. Greedy, optimistic initial values
   - initial action reward values = 10
2. Epsilon greedy, fixed step size
   - epsilons tested = [0.10, 0.15, 0.20, 0.25, 0.30, 0.40]
3. Epsilon greedy, decreasing step size
   - epsilons tested = [0.10, 0.15, 0.20, 0.25, 0.30, 0.40]
   - After every time step the value of step size decreases by 1%.

Results

The evaluation of the agents in this part is done through observation of terminal reward distributions using histograms and box plots. For every type of the problem (drift change, mean-reverting change, and abrupt change), a total of 11 models with different hyper parameters were tested, which amounts to 33 different models. Even though carrying out the hyper-parameter testing through pilot runs could be less computation-intensive, testing each hyper-parameter in the actual problem could be a more robust method of checking its effectiveness. The terminal reward distribution plots for the models are given in the appendix, and their box plot comparison is given below. Note that every model went through 1000 runs, and each run 20,000 time steps.







By analyzing the given plots the following conclusions can be arrived at:

- Gradual changes
  - Model rankings: Decreasing epsilon > Optimistic greedy > Stationary epsilon
  - Among decreasing step-size epsilon greedy models, the one with epsilon=0.40 achieved the highest mean and median. The performance of next best epsilon values was comparable to the best model.
- Mean reverting changes
  - All the models yielded a mean and median of 0 for their terminal average rewards.
  - The spread of the data points for all the models are quite similar and symmetrical with respect to the mean reward of 0.
  - There is no clear "best" model in this case. Depending on the problem, one person might favor slightly lower or higher spread over the other ones.
- Abrupt changes
  - All models yielded a negative mean terminal reward.
  - The best model was stationary step-size epsilon=0.10 greedy; it yielded the highest mean and median for average terminal rewards acquired.
  - Hyper-parameter tuning highly affects the performance of the models.

- Incremental simple average updates of action value estimates are not the best method of updating for moving rewards. It is better to use updates that attempt to forget the far past.

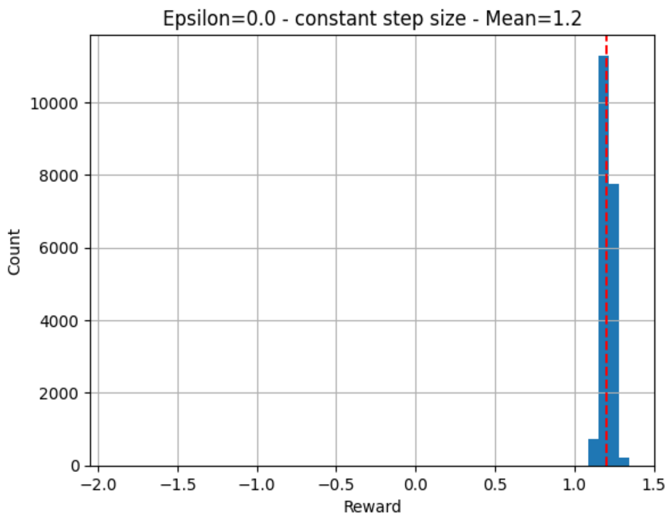
Appendix

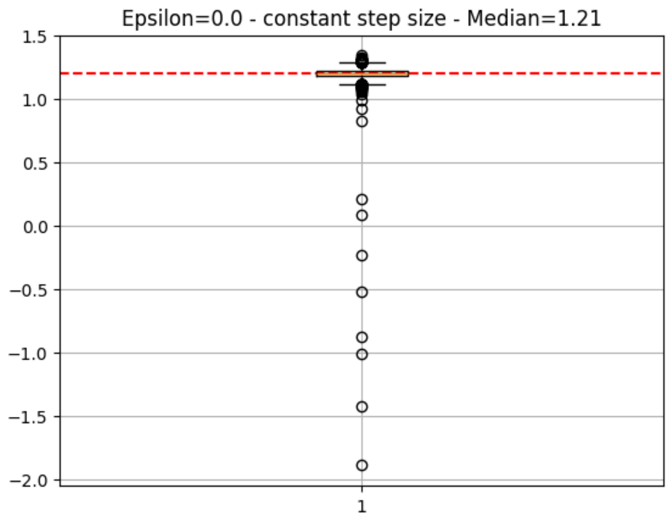
The repository for this assignment can be found in the link below:
https://github.com/P-A-D/RL_Assignment_1

The terminal reward distribution plots of **Part 2** can be found below.

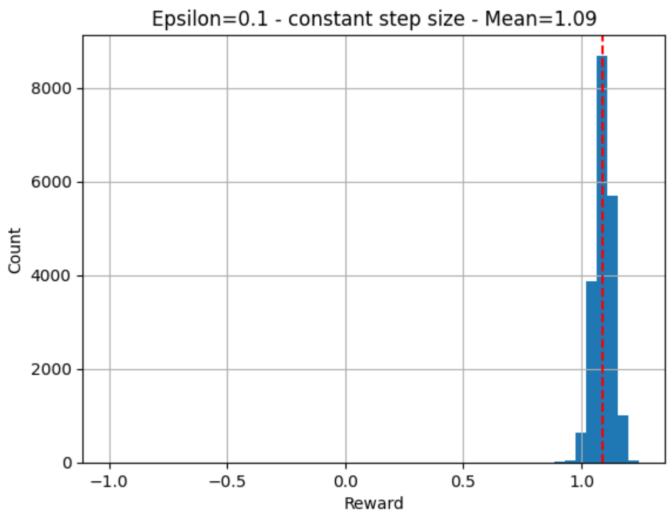
Gradual changes – drift

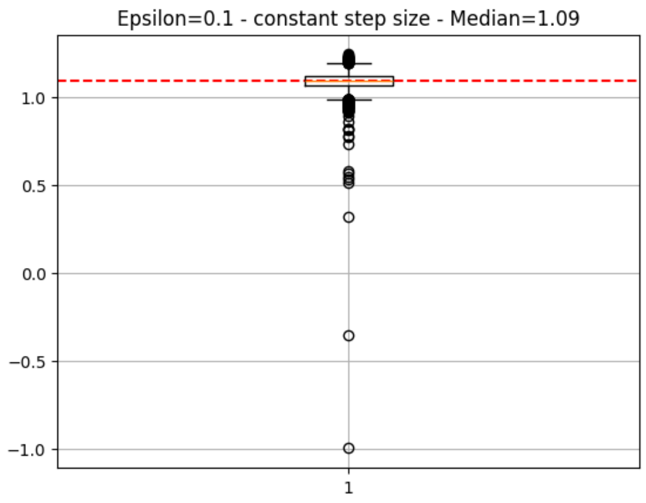
optimistic greedy



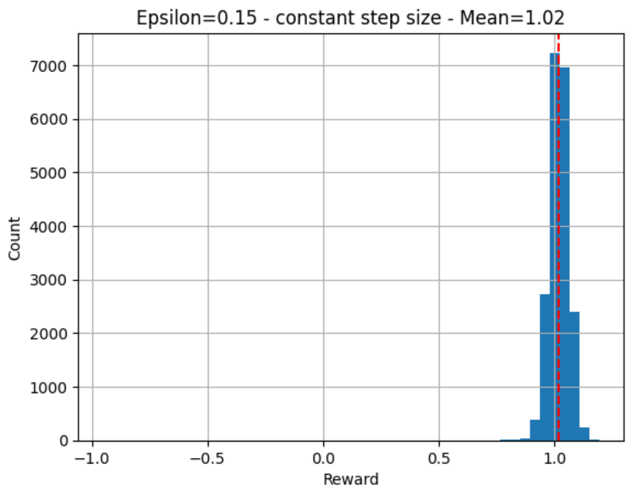


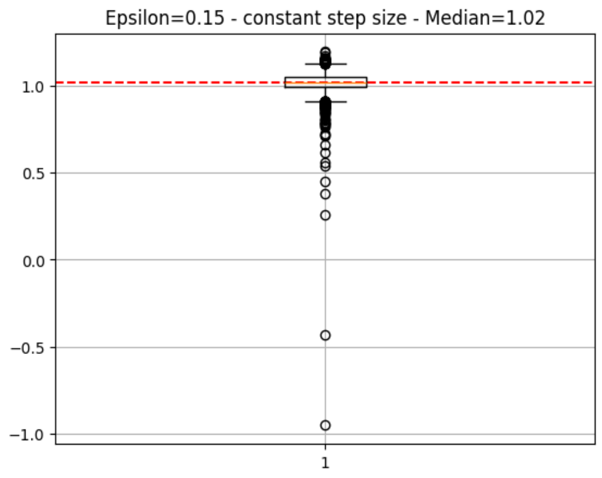
stationary epsilon = 0.1 greedy



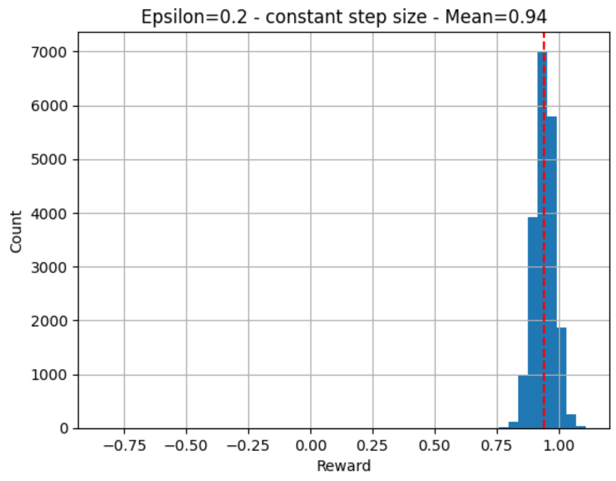


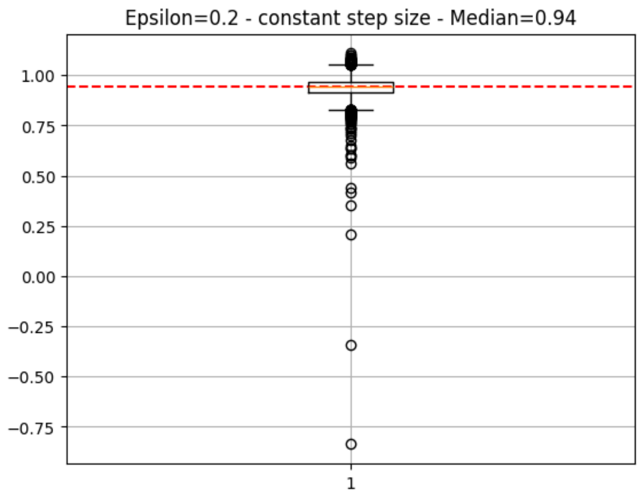
stationary epsilon = 0.15 greedy



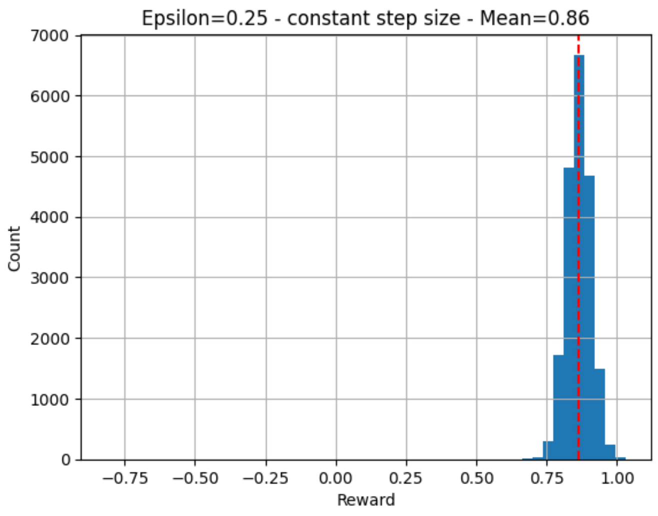


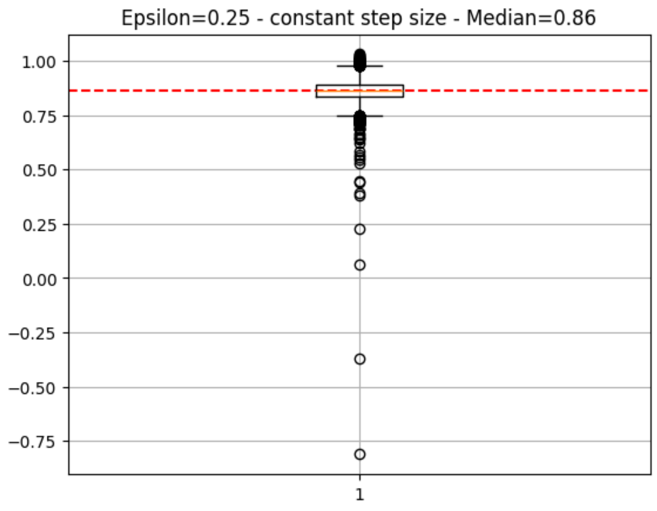
stationary epsilon = 0.2 greedy



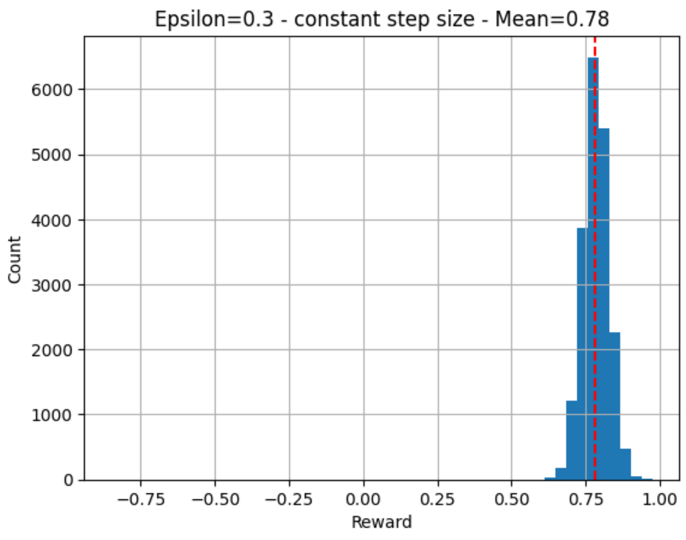


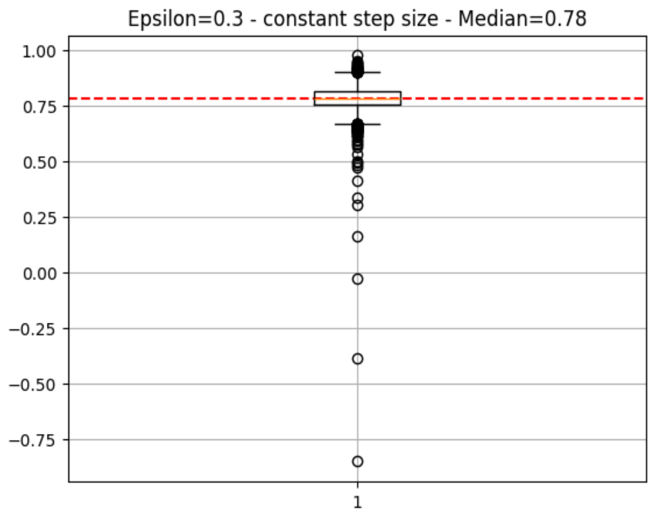
stationary epsilon = 0.25 greedy



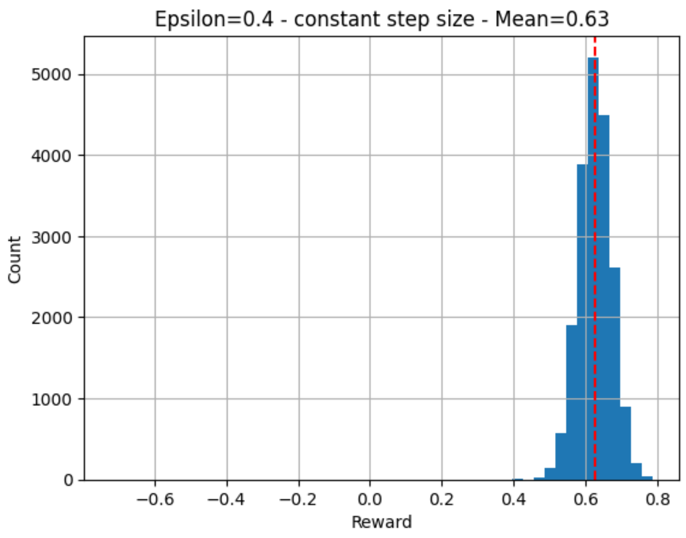


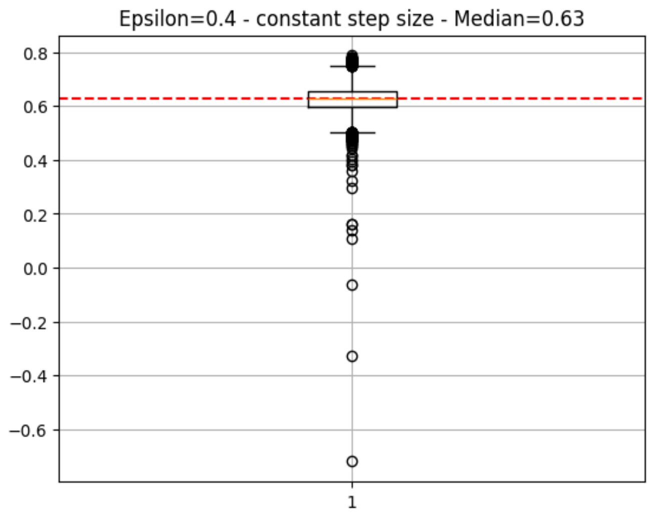
stationary epsilon = 0.3 greedy



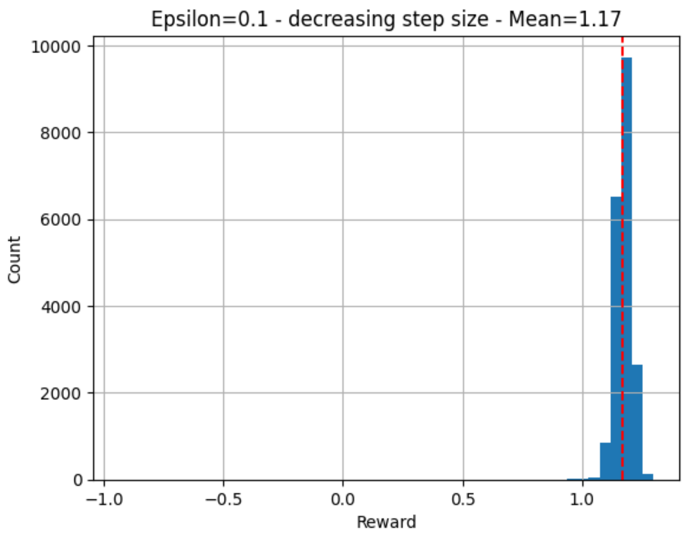


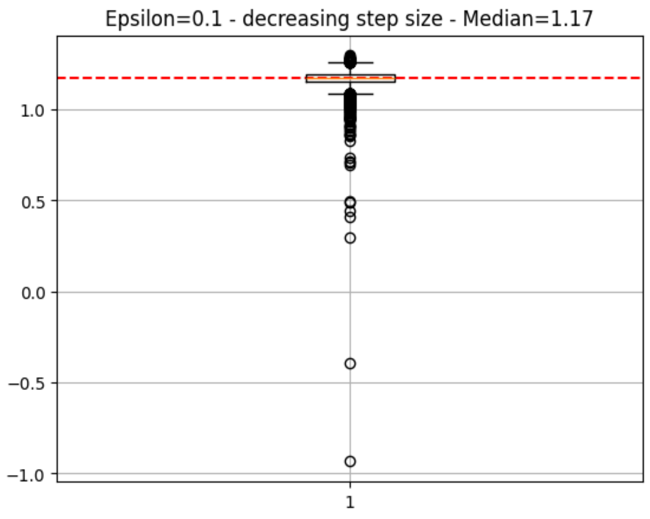
stationary epsilon = 0.4 greedy



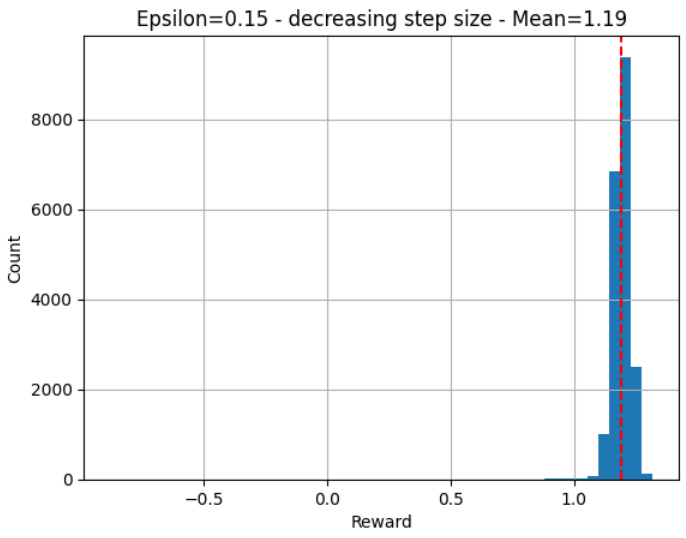


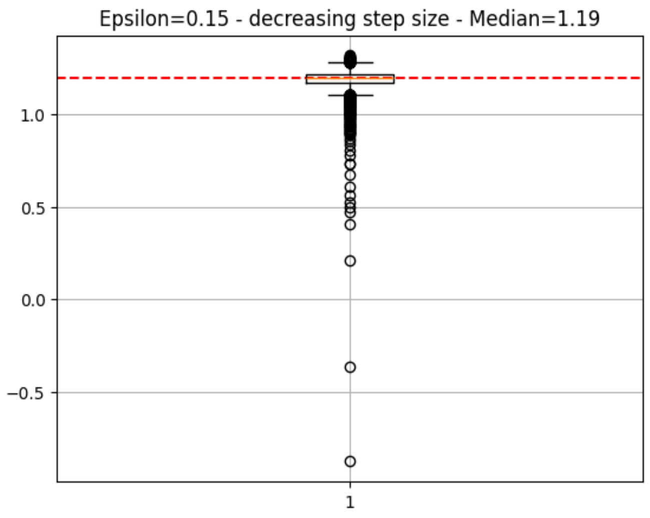
decreasing epsilon = 0.1 greedy



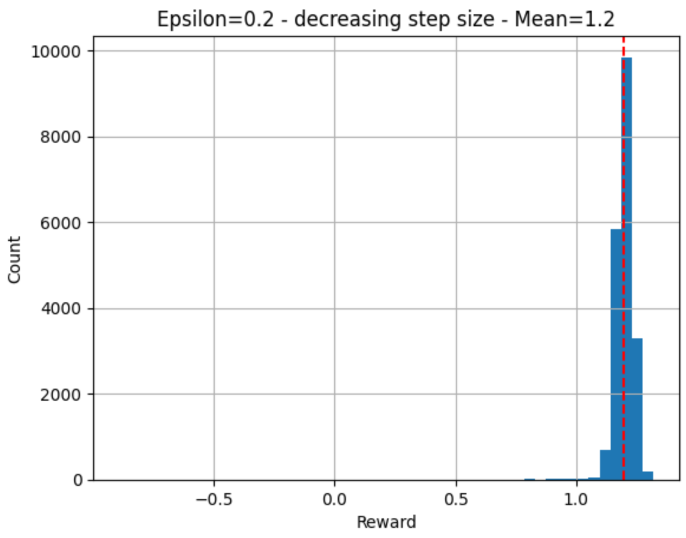


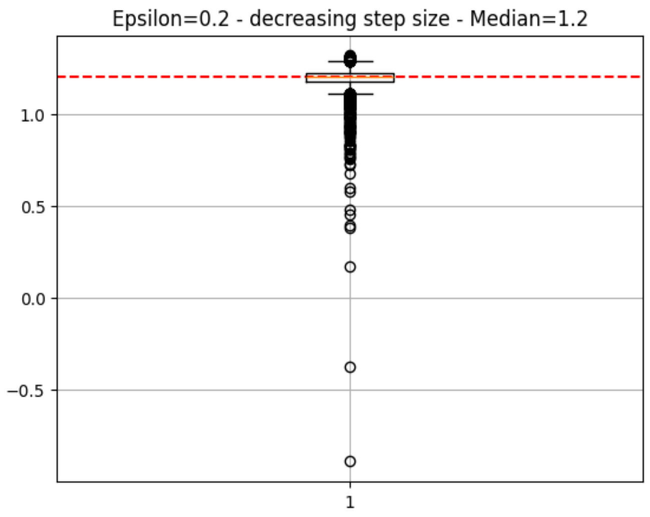
decreasing epsilon = 0.15 greedy



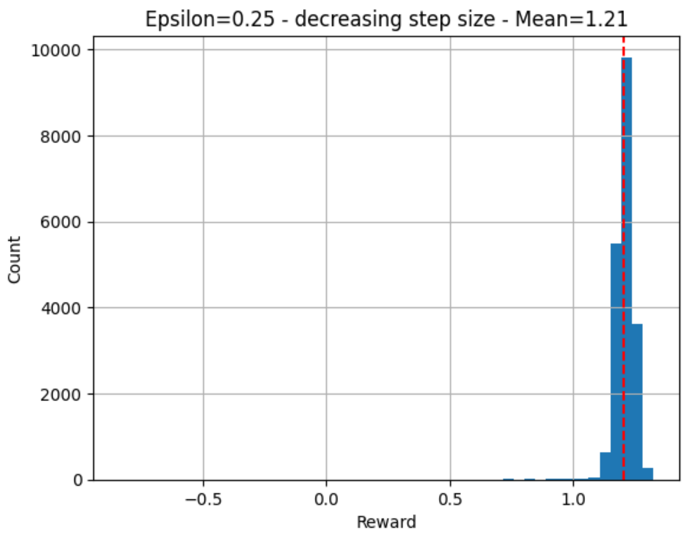


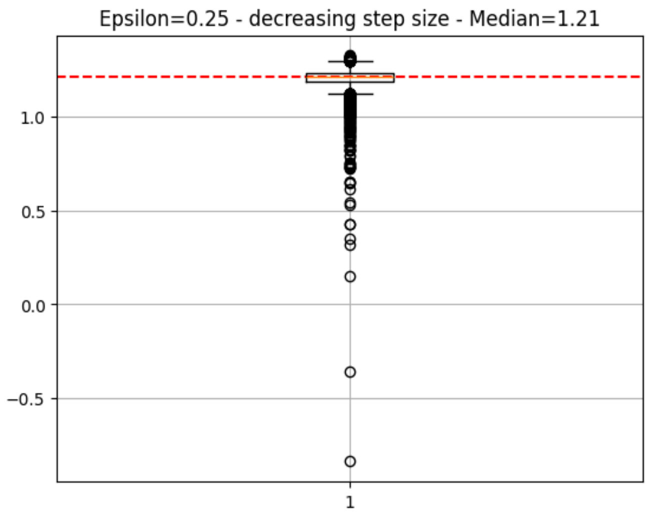
decreasing epsilon = 0.2 greedy



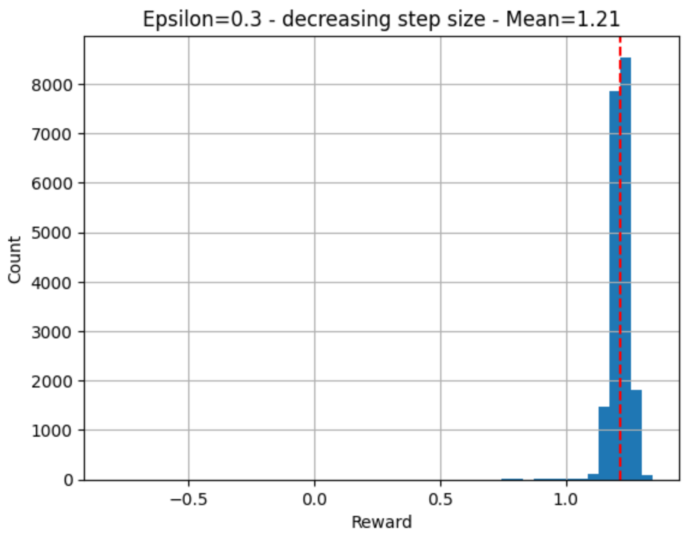


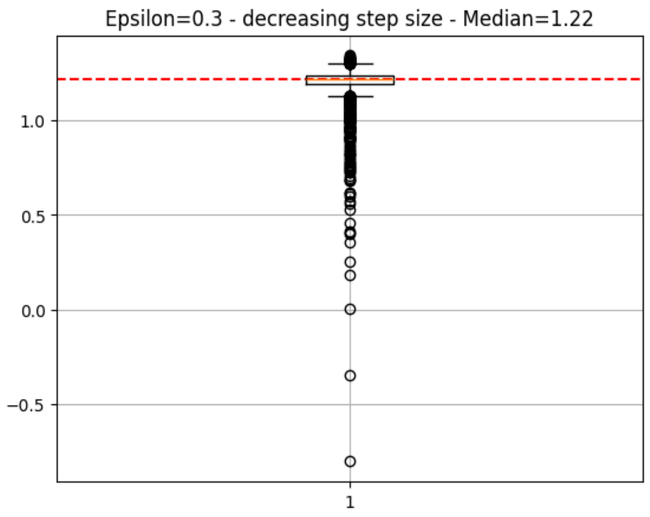
decreasing epsilon = 0.25 greedy



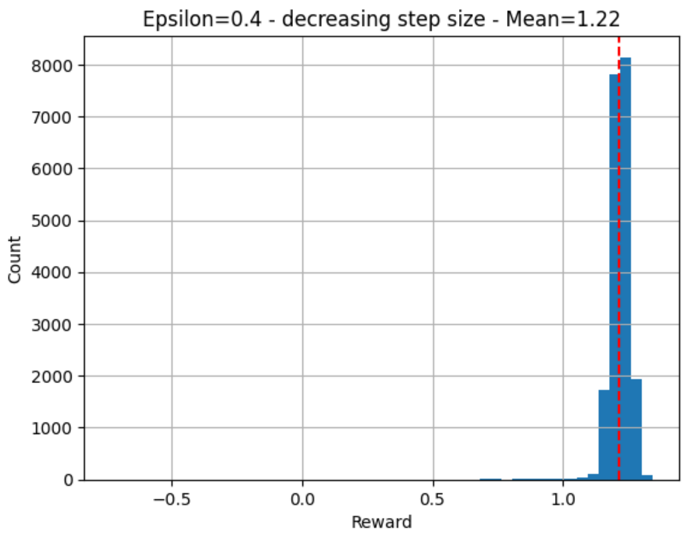


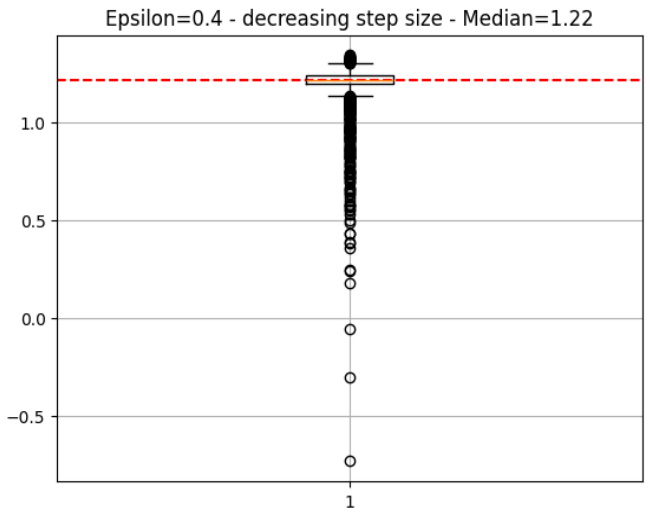
decreasing epsilon = 0.3 greedy





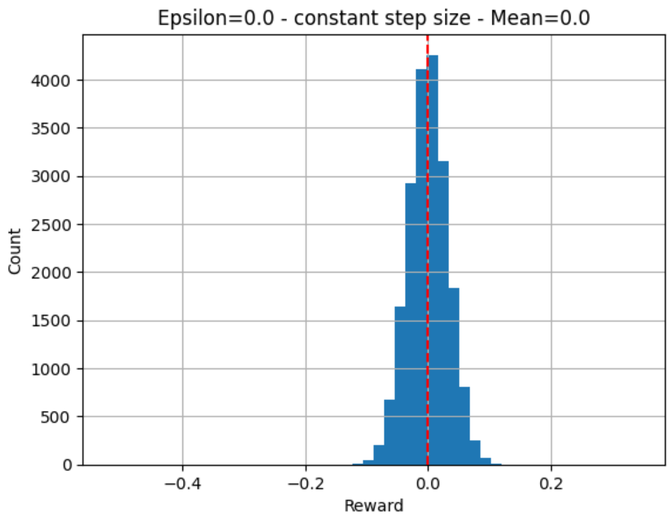
decreasing epsilon = 0.4 greedy

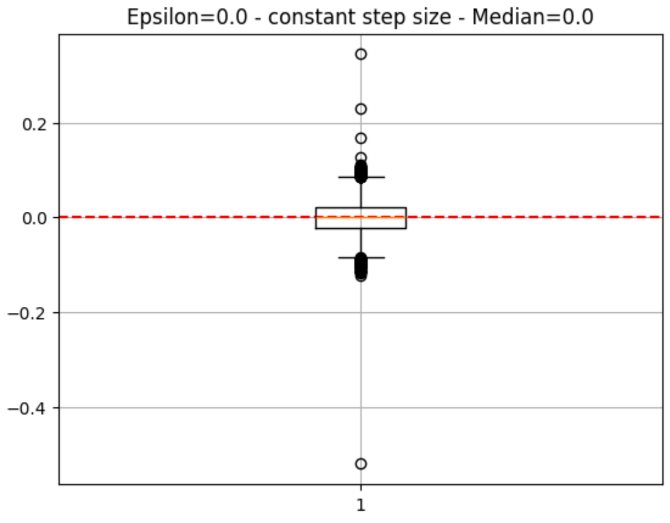




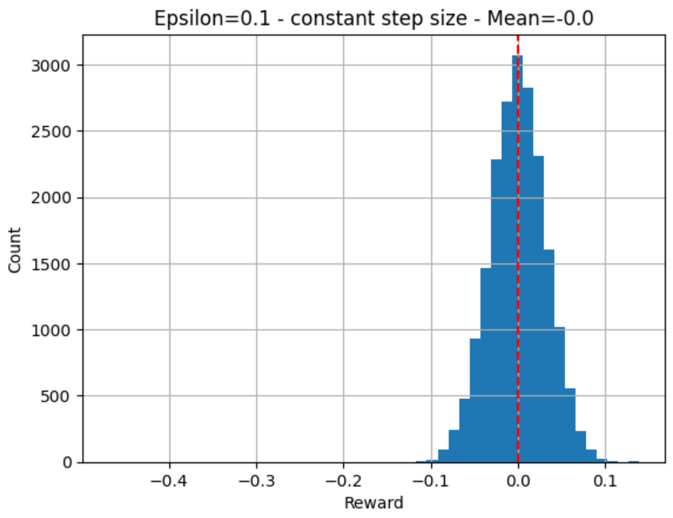
Gradual changes – mean reversion

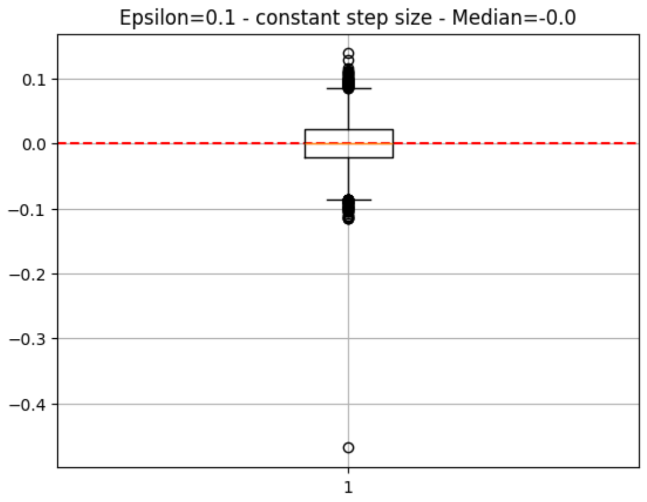
optimistic greedy



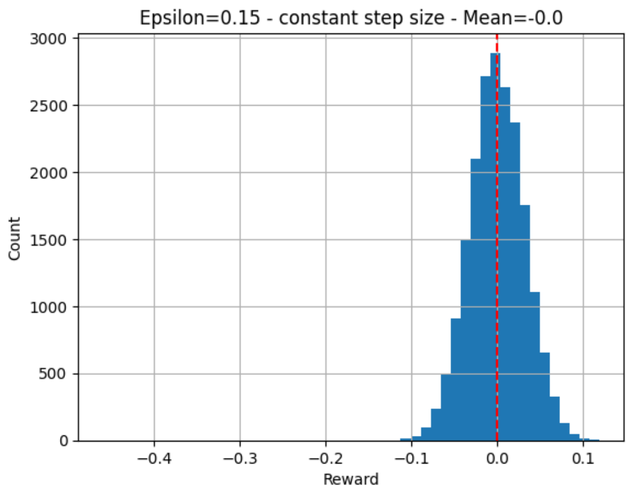


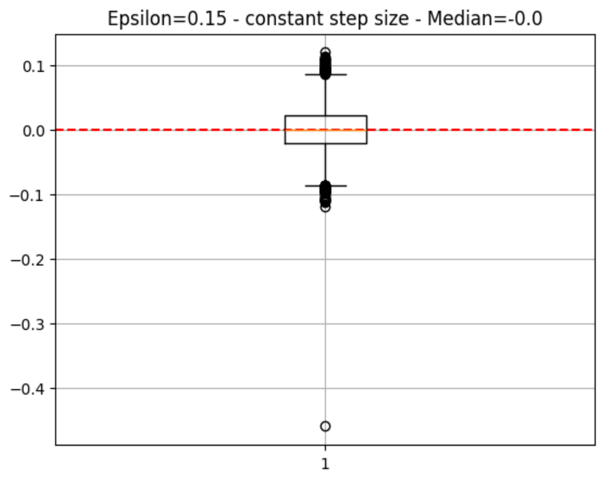
stationary epsilon = 0.1 greedy



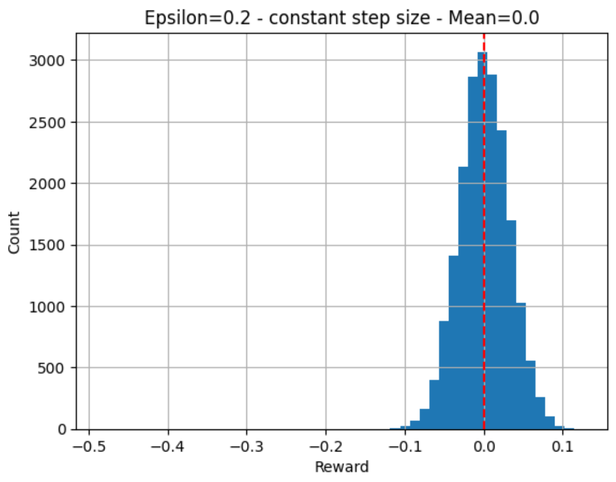


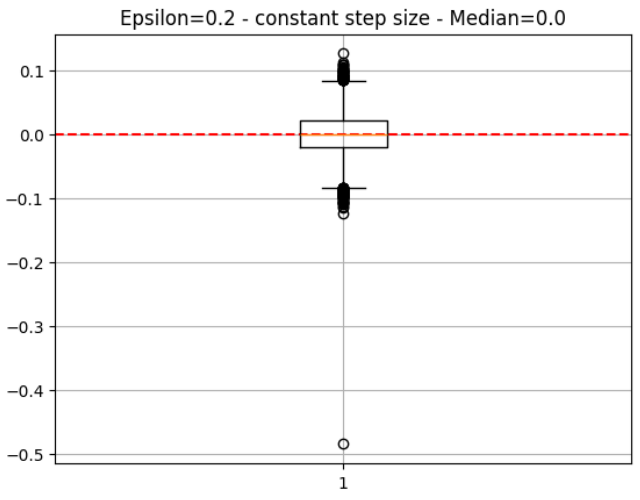
stationary epsilon = 0.15 greedy



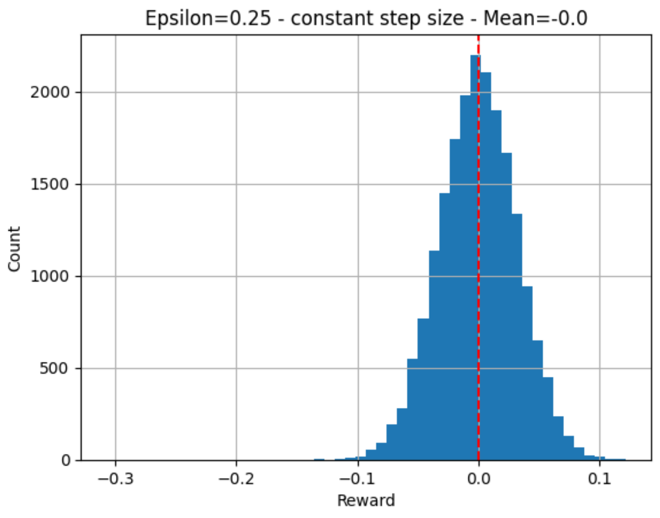


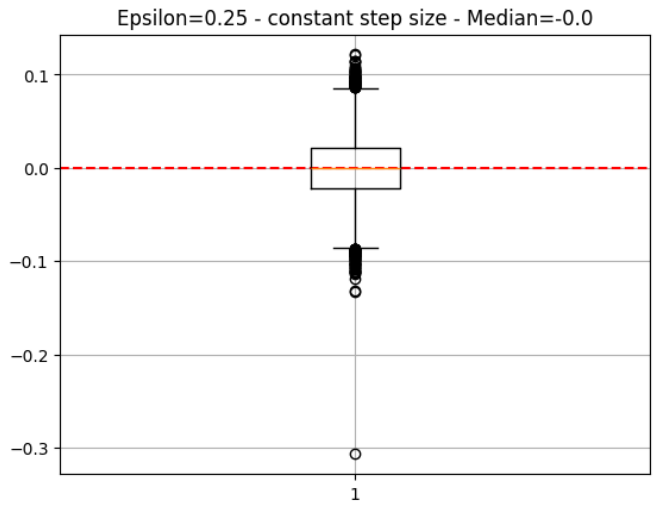
stationary epsilon = 0.2 greedy



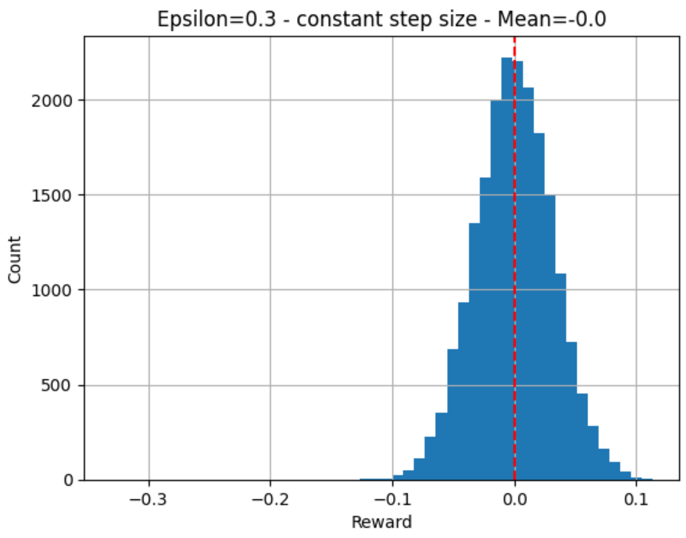


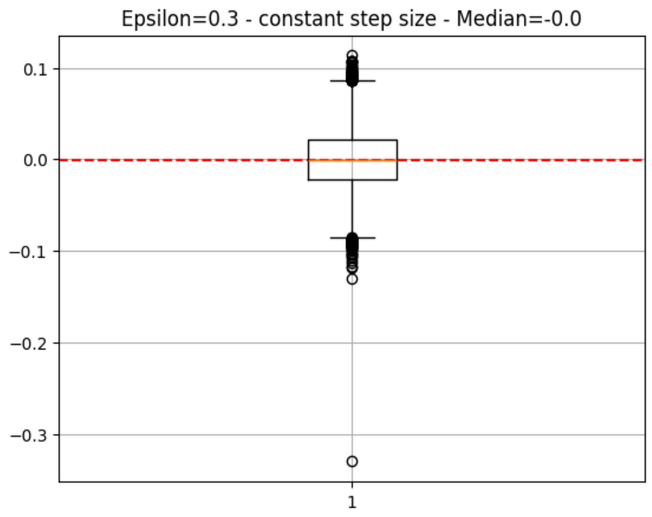
stationary epsilon = 0.25 greedy



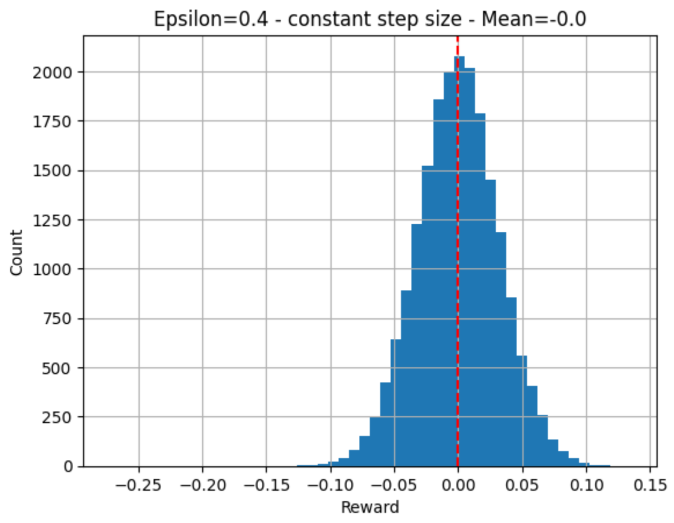


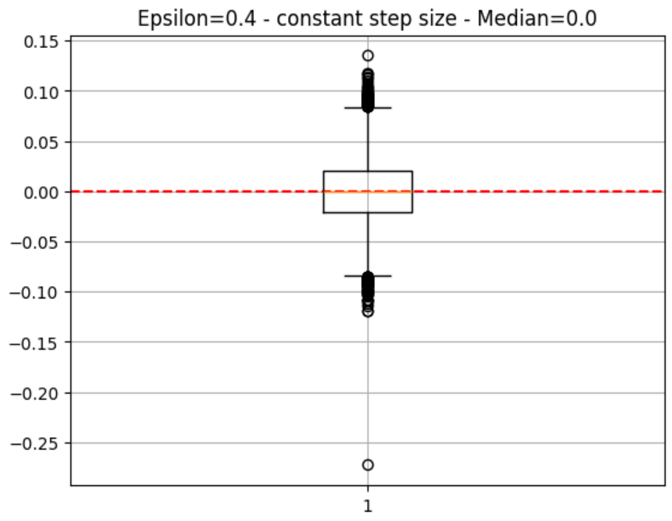
stationary epsilon = 0.3 greedy





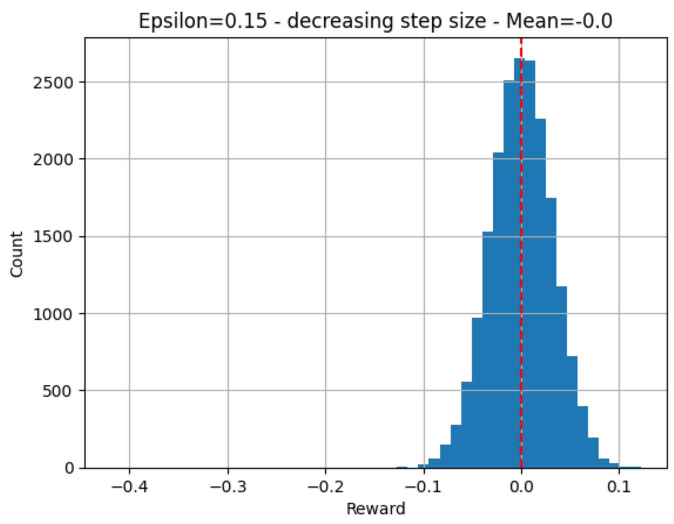
stationary epsilon = 0.4 greedy

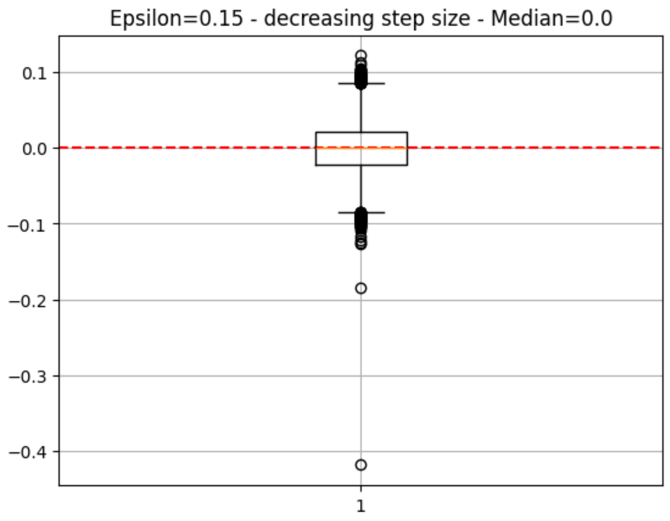




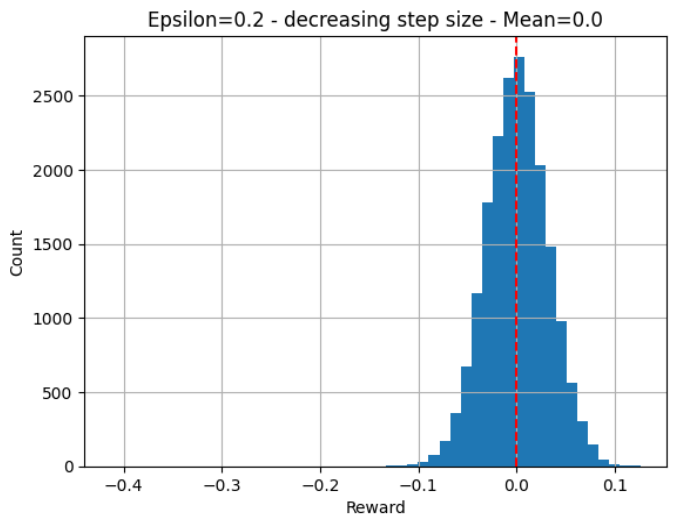
decreasing epsilon = 0.1 greedy

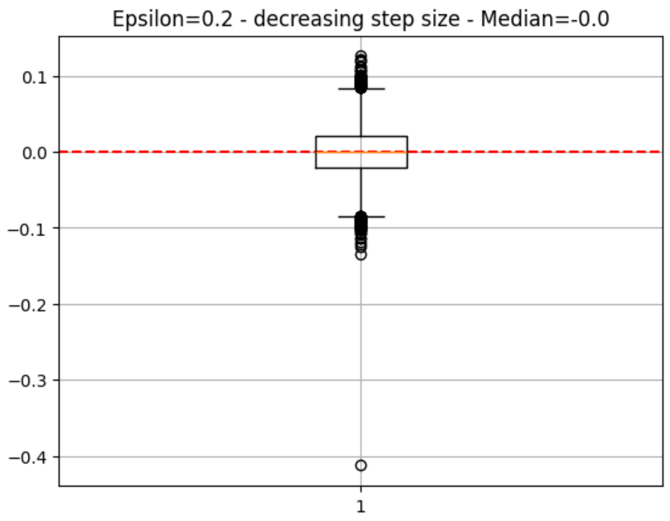
decreasing epsilon = 0.15 greedy



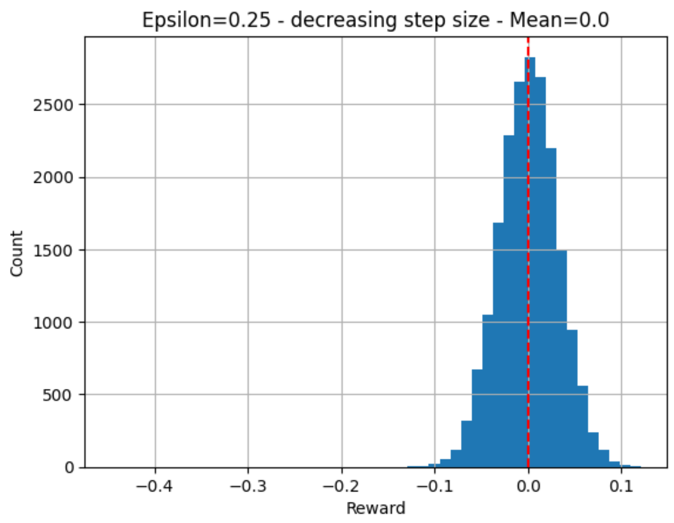


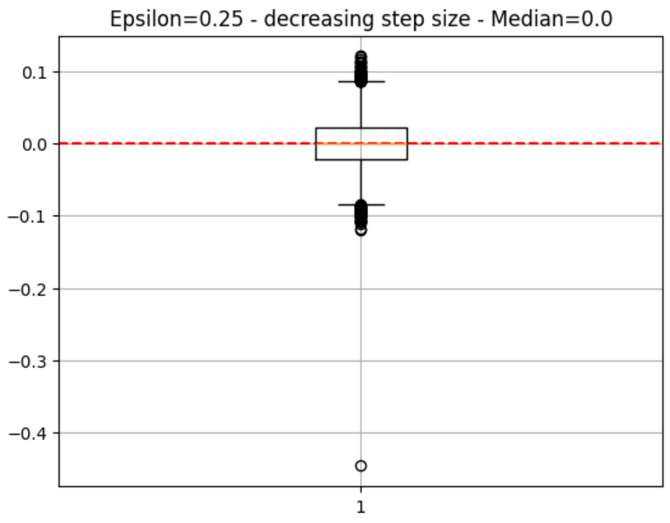
decreasing epsilon = 0.2 greedy



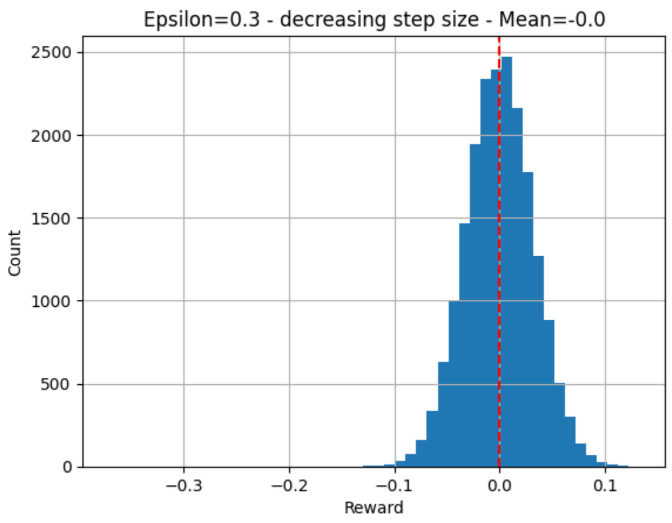


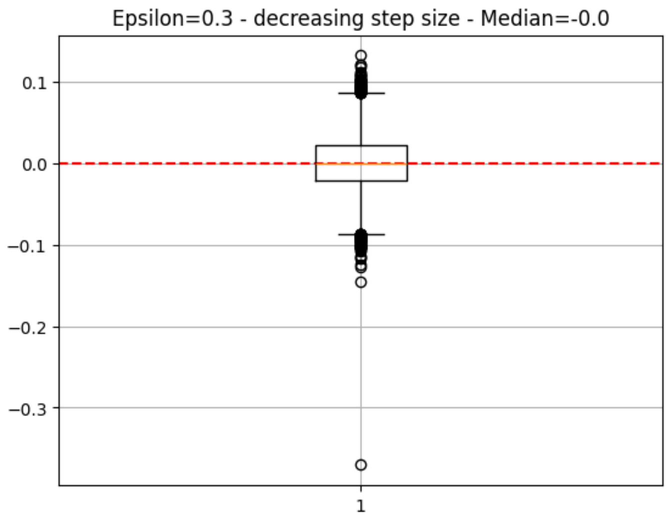
decreasing epsilon = 0.25 greedy



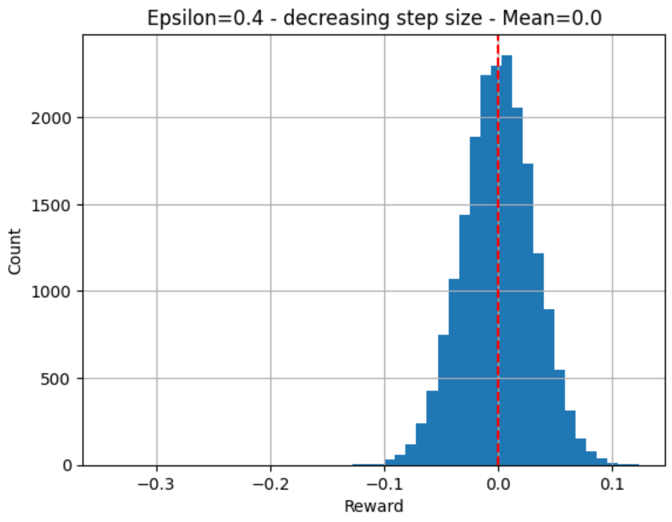


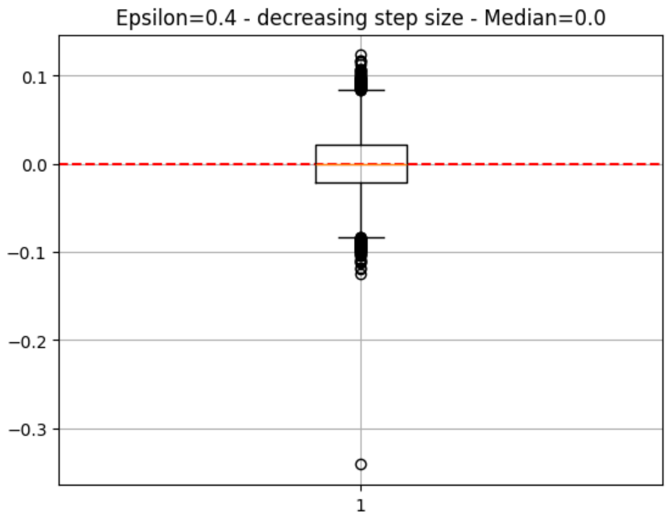
decreasing epsilon = 0.3 greedy





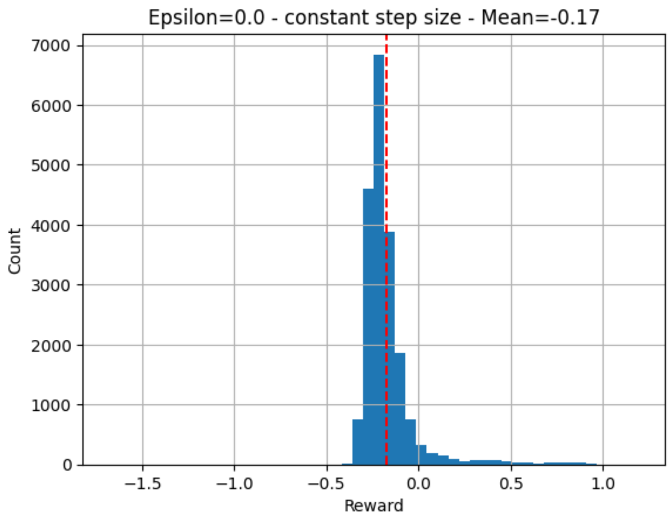
decreasing epsilon = 0.4 greedy

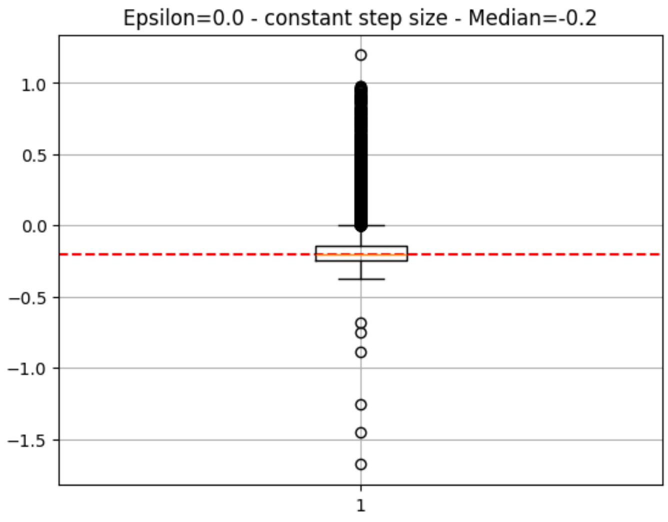




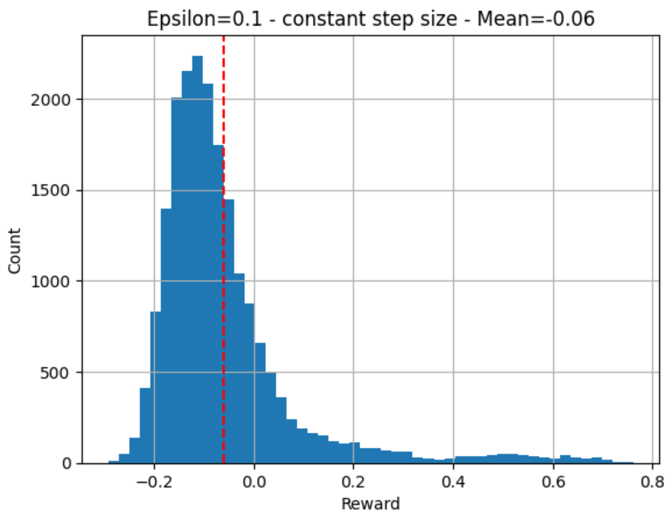
Abrupt changes

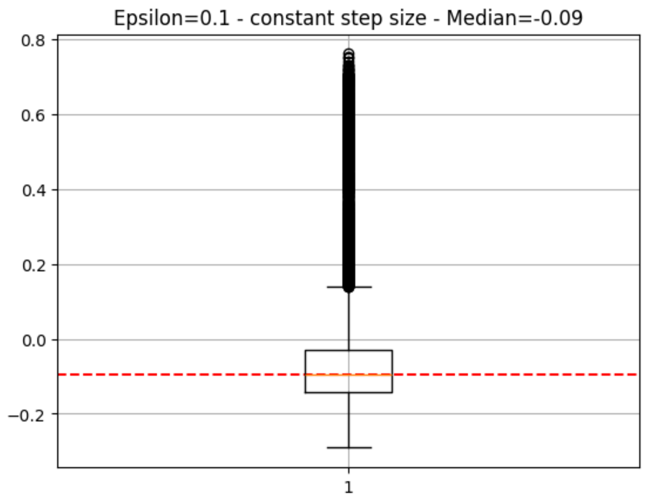
optimistic greedy



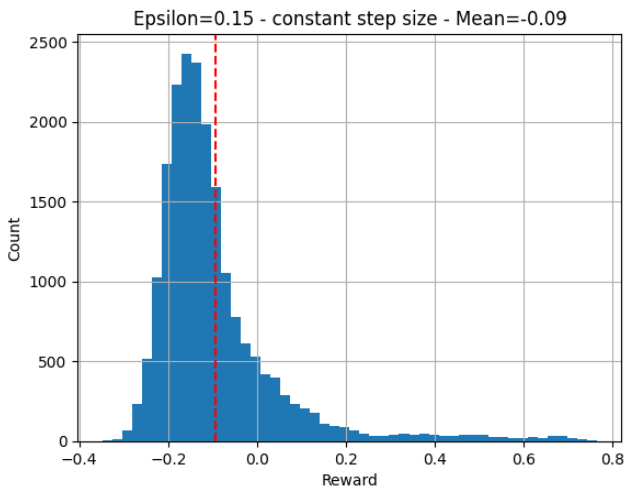


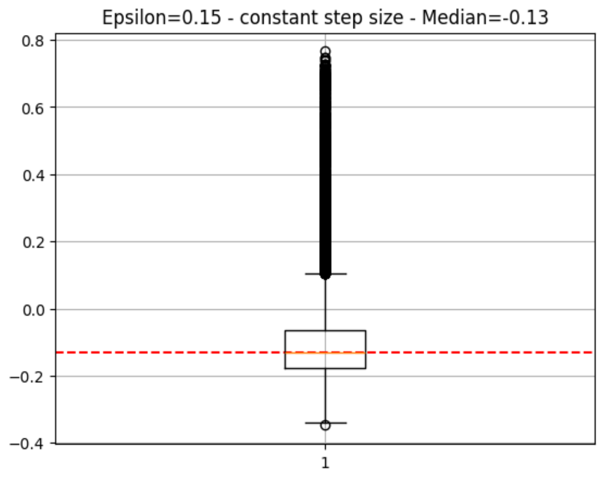
stationary epsilon = 0.1 greedy



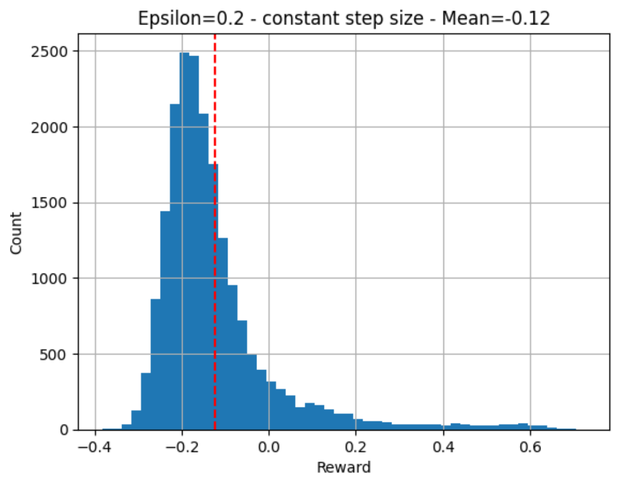


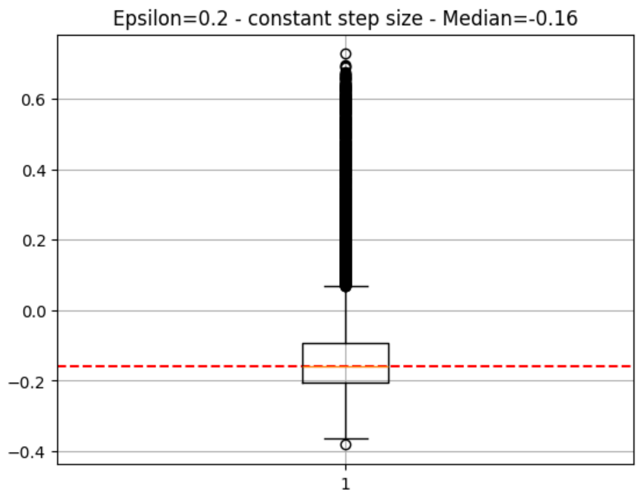
stationary epsilon = 0.15 greedy



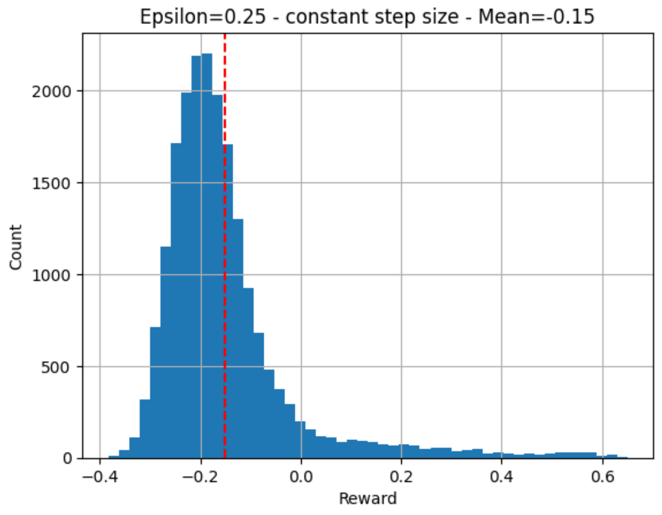


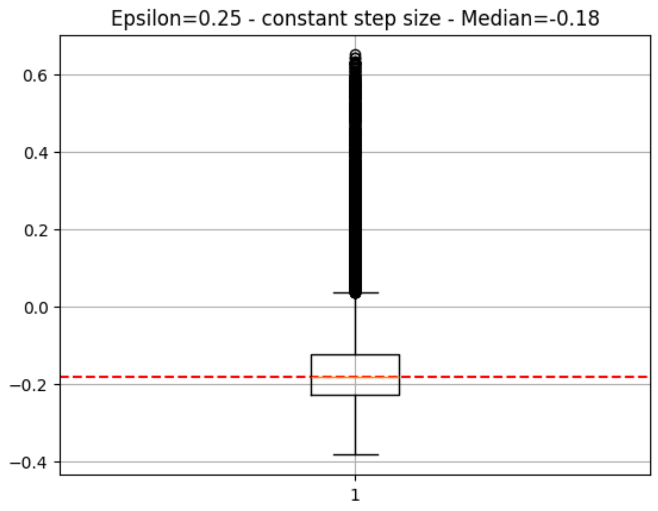
stationary epsilon = 0.2 greedy



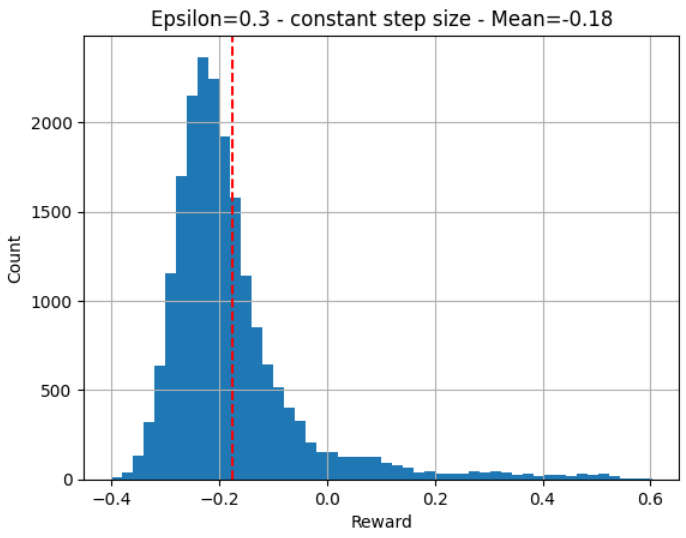


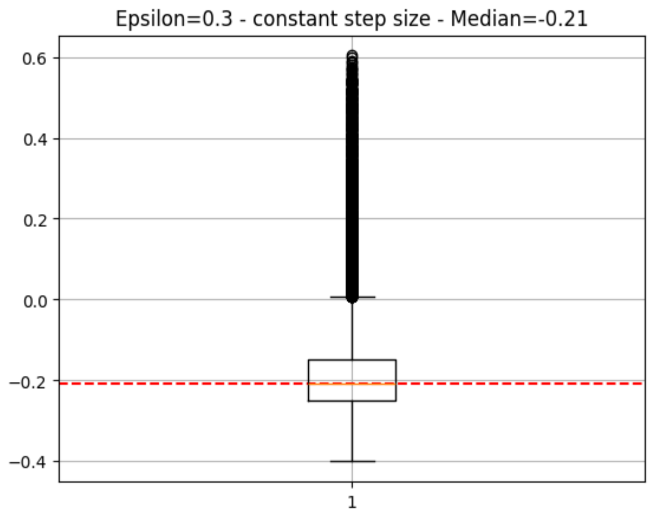
stationary epsilon = 0.25 greedy



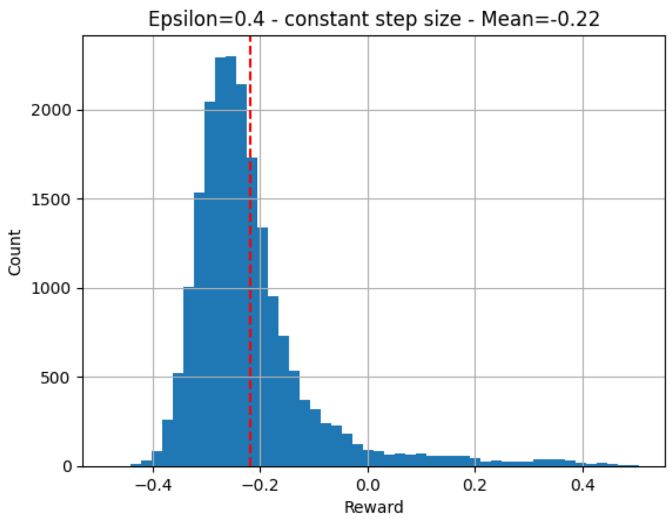


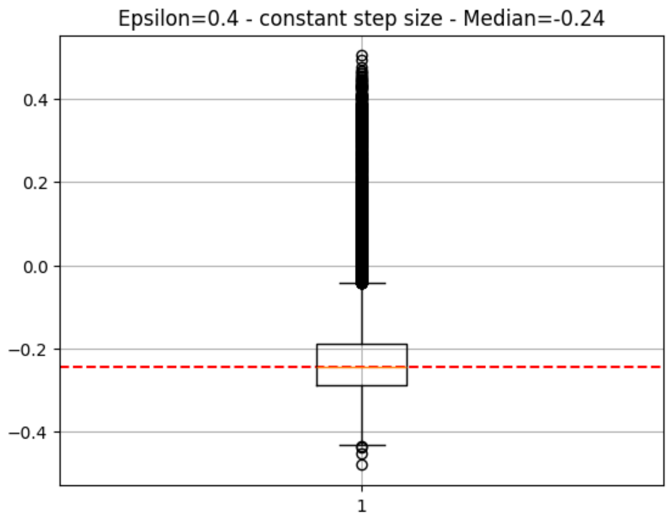
stationary epsilon = 0.3 greedy



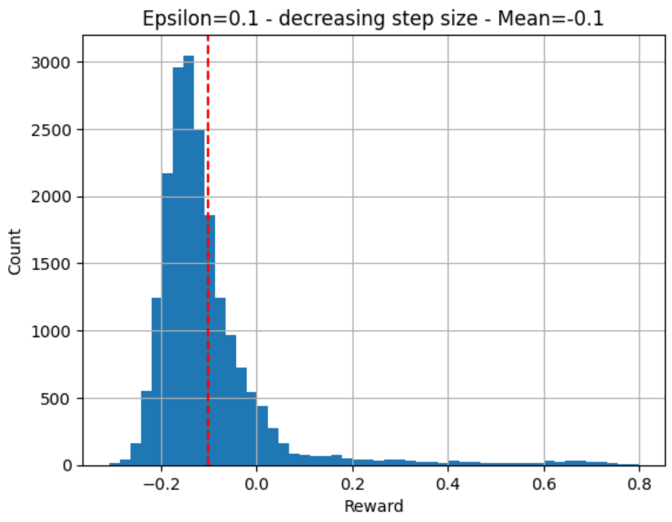


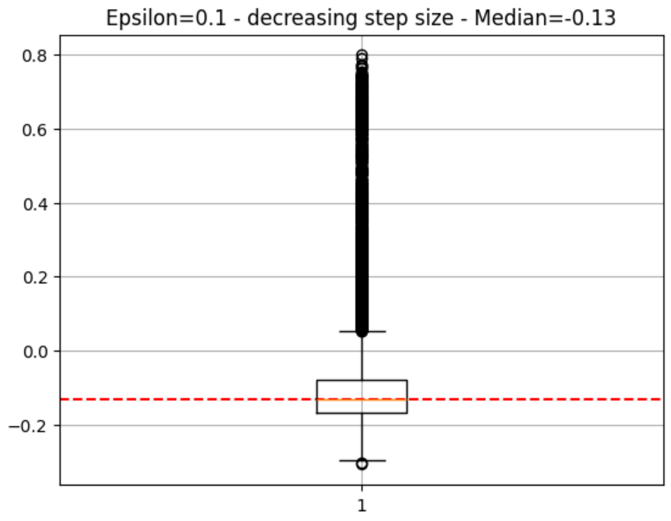
stationary epsilon = 0.4 greedy



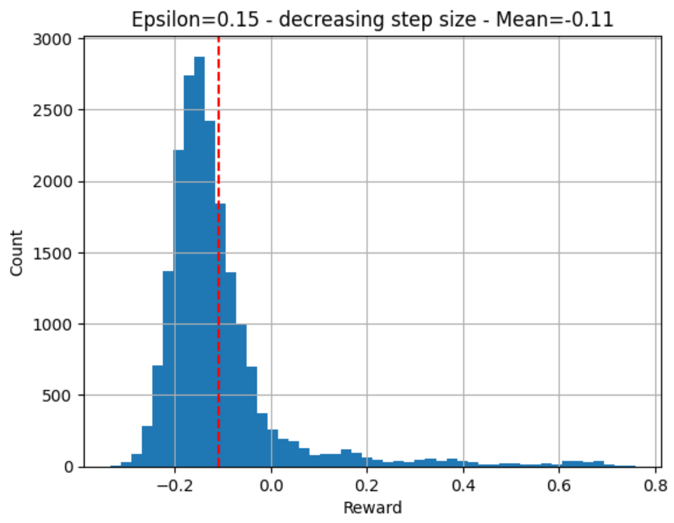


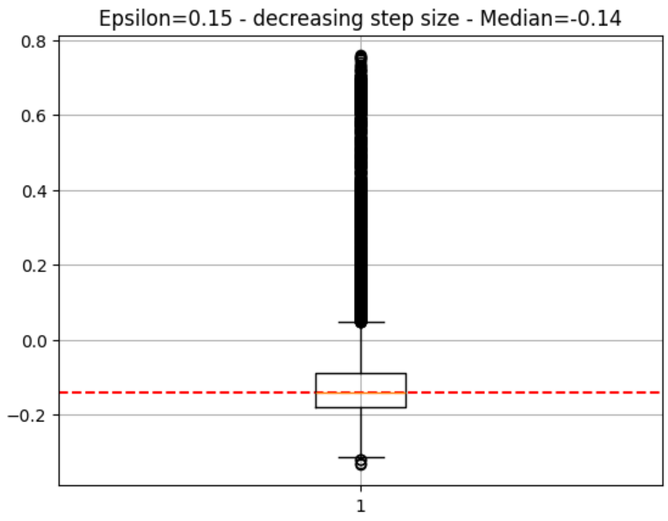
decreasing epsilon = 0.1 greedy



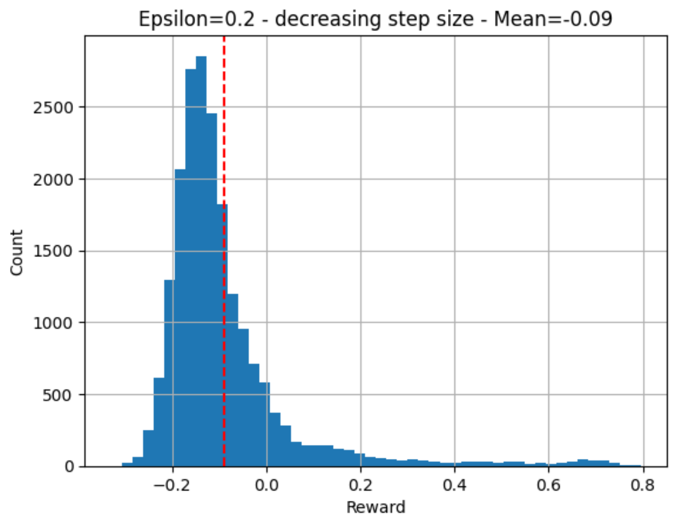


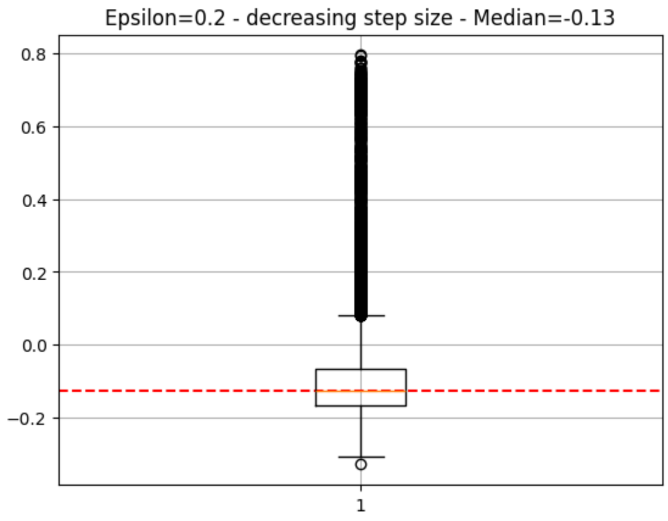
decreasing epsilon = 0.15 greedy



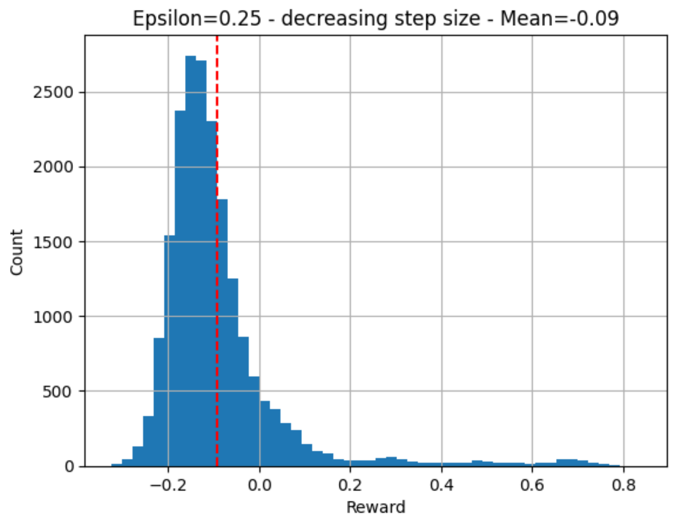


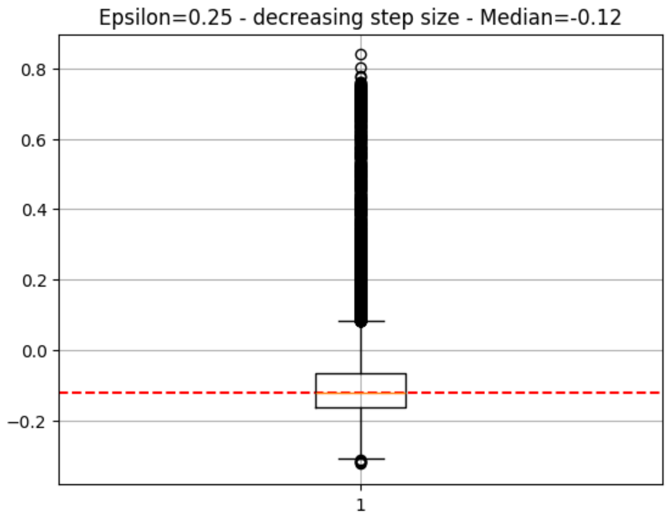
decreasing epsilon = 0.2 greedy



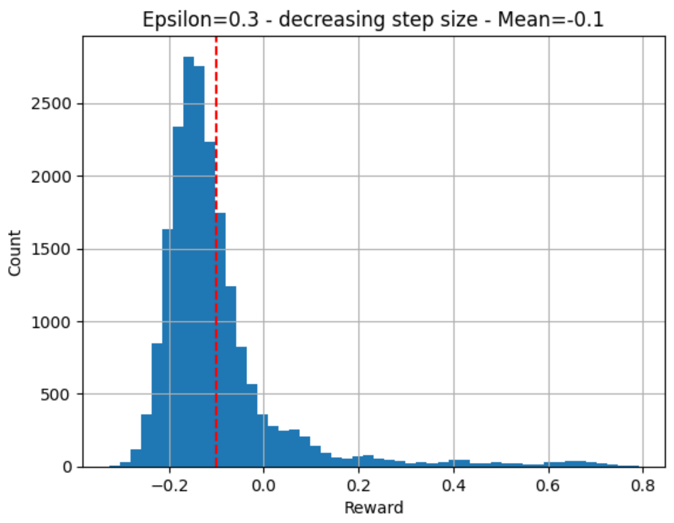


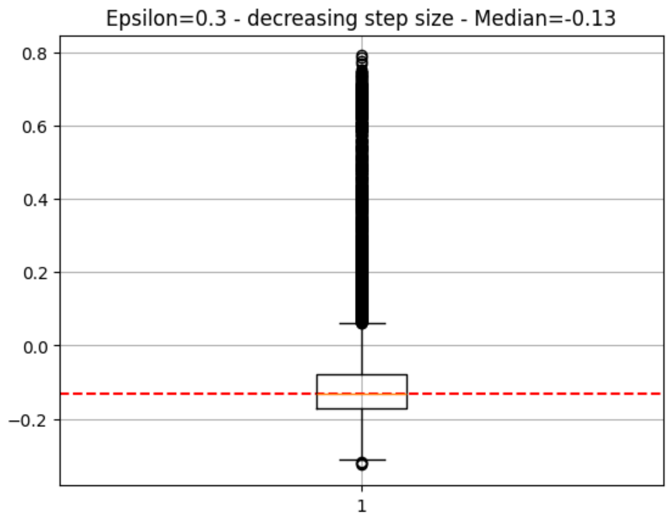
decreasing epsilon = 0.25 greedy



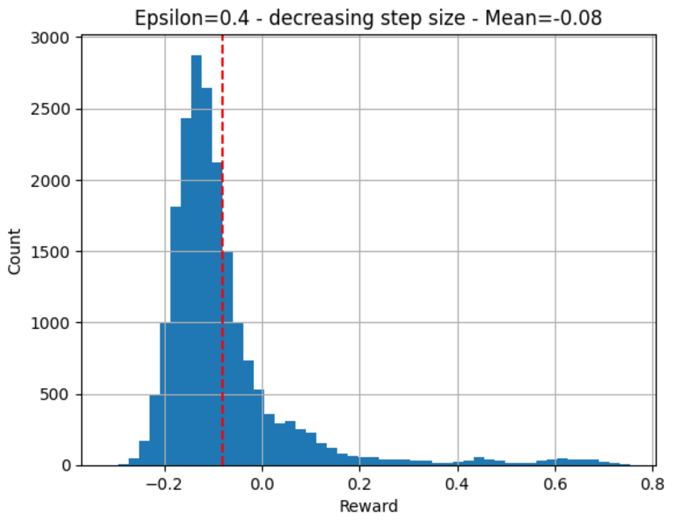
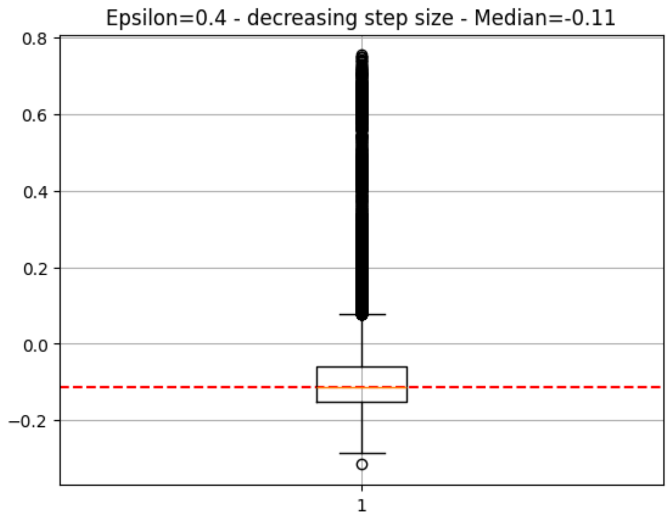


decreasing epsilon = 0.3 greedy





decreasing epsilon = 0.4 greedy


Epsilon=0.4 - decreasing step size - Mean=-0.08
Count
Reward

Epsilon=0.4 - decreasing step size - Median=-0.11