دانشگاه تهران
پردیس دانشکده‌های فنی
دانشکده برق و کامپیوتر

بهینه سازی توزیع شده و یادگیری
Distributed Optimization and Learning

پروژه 3
Third Project

عرفان میرزایی
Erfan Mirzaei

810199289

دکتر کبریایی
Dr.Kebriaei

بهمن ماه 1400
January 2022

Contents

Section 1) Problem Definition 5
Section 2) Single-Agent Multi-Armed Bandit 7
2.1. 2-Armed Bandit 7
2.1.1. Easy Problem 8
2.1.2. Medium Problem 9
2.1.3. Hard Problem 11
2.2. 10-Armed Bandit 14
2.2.1. Medium Problem 14
2.2.2. Hard Problem 16
Section 3) Multi-Agent Multi-Armed Bandit 18
3.1. 2-Armed Bandit 19
3.1.1. Easy Problem 19
3.1.2. Medium Problem 21
3.1.3. Hard Problem 23
3.2. 10-Armed Bandit 25
3.2.1. Medium Problem 25
3.2.2. Hard Problem 27
Section 4) Conclusion 29
References: **30**

Abstract

We implemented bandit learning algorithms for single-agent and multi-agent scenarios in this project. To this end, we used a non-stationary environment where some reward functions change disruptively.

In the first part, we considered different single-agent multi-armed bandits with 2 and 10 arms. In both cases, we designed environments with different difficulty levels, i.e., discriminability of rewards. We used Epsilon-greedy, Upper-Confidence Bound(UCB), Policy-gradient, Thompson Sampling, and Actor-Critic algorithms in this part.

In the second part, we considered multi-agent multi-armed bandit scenarios. Similar to the former, we considered different numbers of arms with different reward probability distributions. In this part, we used Joint Action Learners(JAL), Free Maximum Q-value(FMQ), Distributed Q-learning, and Multi-agent Actor-Critic Algorithm.

Section 1) Problem Definition

In this project, we want to implement Bandit learning algorithms for single-agent and multi-agent scenarios.

Today, bandit problems have many applications, such as A/B testing, advertisement placing, Network Routing, etc. However, in this project, we considered the dynamic pricing problem as the main application of our work. Online retailer companies use dynamic pricing to determine the best price for their products automatically. Customers arrive sequentially, and the learner agent sets the price. If the price is lower than the user's valuation, the user will purchase the product. Among the exciting characteristics of this problem are (a) there is no actual product valuation; instead, only a binary signal that the price was too low or too high is observed, and (b) the pricing structure is monotonic. An item priced at $10 would certainly sell for $5, but whether it would sell for $11 is unknown.

Another example is a music recommendation system. The song Spotify recommends should not be the same throughout the year; we might expect love songs to be popular in February, party music in the summer, and holiday songs in the winter. As tastes change, Spotify's recommendations will also change.

A bandit problem is a sequential game between a learner and an environment. The game is played over n rounds, where n is a positive natural number called the horizon. In each round $t \in [n]$, the learner first chooses an action A_t from a given set A, and the environment then reveals a reward $X\_t \in$ R.

Of course, the learner cannot peek into the future when choosing their actions, which means that A_t should only depend on the history $H\_t{-}1 = (A1, X1, \ldots, At{-}1, Xt{-}1)$. A policy is a mapping from history to actions. The bandit agent interacts with an environment at each round and adopts its policy. An environment based on the history of the last round and the selected action of the learner returns a reward signal.

The learner's goal is to maximize the sum of its rewards in the n rounds by choosing the proper action. The critical assumption is that the environment is unknown to the learner. Thus, the critical distinction between the bandit and classic optimization problems is that the objective function is not explicitly known for the agent.

For evaluating the learner's performance, it is common to use regret. The regret of the learner relative to a policy π (not necessarily that followed by the learner) is the difference between the total expected reward using policy π for n rounds and the total expected reward collected by the learner over n rounds.

In this case, a more appropriate regret model is relative to the best arm at each time step. Let's formalize this notion: at each time t = 1; … ; T, each of the arms a = 1; … ;K has some true (unknown to the learner) mean t(a). The learner's notion of regret compares to the best arm at every time step,

$$R_T := \sum_{t=1}^{T} \max_{a \in [K]} \mu_t(a) - \mathbb{E}\left[\sum_{t=1}^{T} \mu_t(a_t)\right]$$

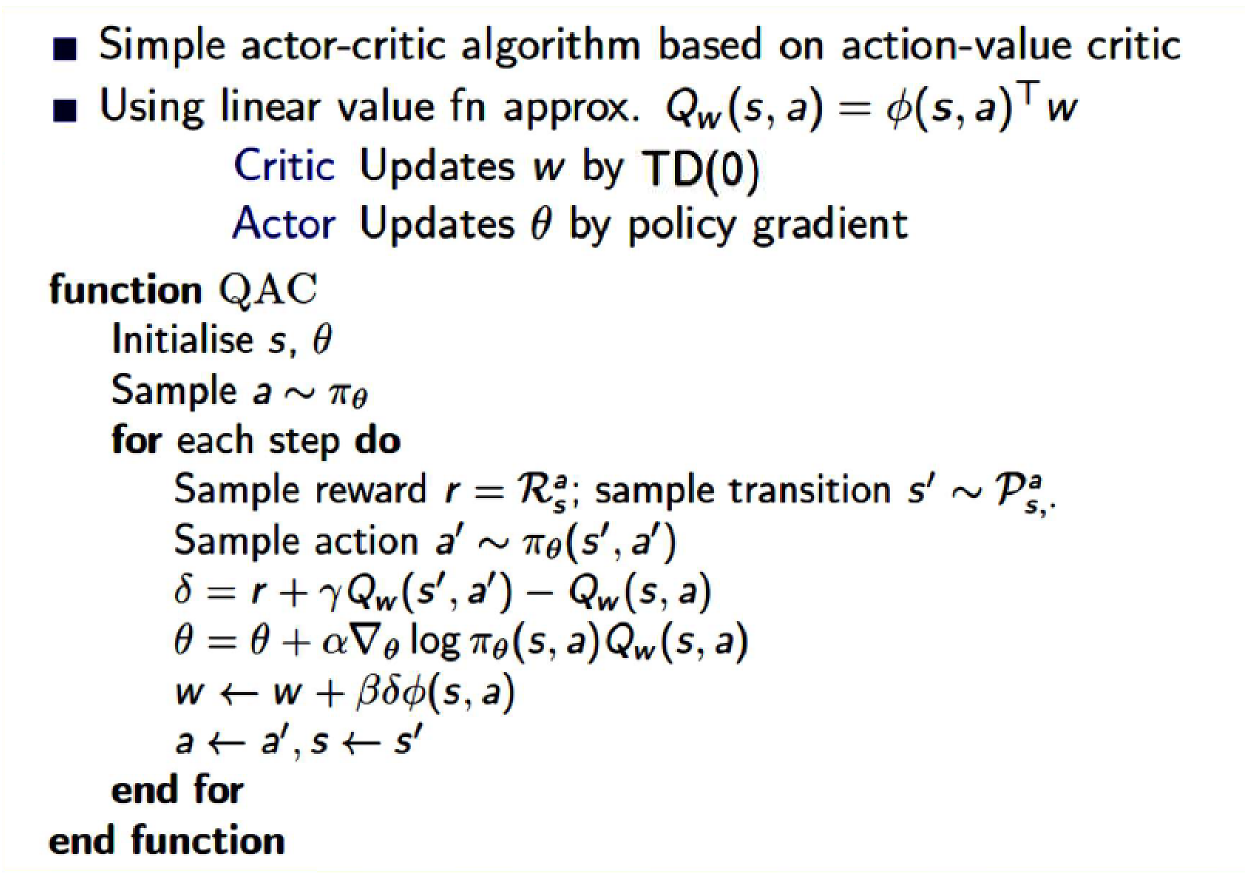
where at is the arm selected by the algorithm at time t. We will measure regret in terms of the number of switches L, where a switch occurs whenever some arm's mean changes (even if that does not change the identity of the best arm). The regret in (1) is known as the dynamic regret, in contrast to the static regret in stationary MAB problems.

A simple problem setting is that of stochastic stationary bandits. In this case, the environment is restricted to generate the reward in response to each action from a distribution that is specific to that action and independent of the previous action choices and rewards. Sometimes we consider the case where the rewards are stochastic but not stationary. The objective of this project is to address non-stationariness in the bandit algorithms. Bandit problems can be compared to supervised learning, though both are static optimization. It is notable that we do not have an objective function in bandit problems. Also, One disadvantage of supervised learning is that data validity is deprecated when the task is non-stationary.

Section 2) Single-Agent Multi-Armed Bandit

This section considers different single-agent multi-armed bandits with 2 and 10 arms. Using reward discrimination, we design environments at different difficulty levels. We use Epsilon-greedy, Upper-Confidence Bound(UCB), Policy-gradient, Thompson Sampling, and Actor-Critic algorithms. Each algorithm was trained for 10,000 steps in each environment. We repeat this procedure 50 times and plot the mean and standard deviation. Also, it is notable that we consider moving averages in all algorithms whenever possible due to the non-stationariness of our environment.

For the Actor-Critic algorithm, we use the following algorithm:



- Simple actor-critic algorithm based on action-value critic
- Using linear value fn approx. $Q_w(s, a) = \phi(s, a)^\top w$
  - Critic Updates w by TD(0)
  - Actor Updates θ by policy gradient

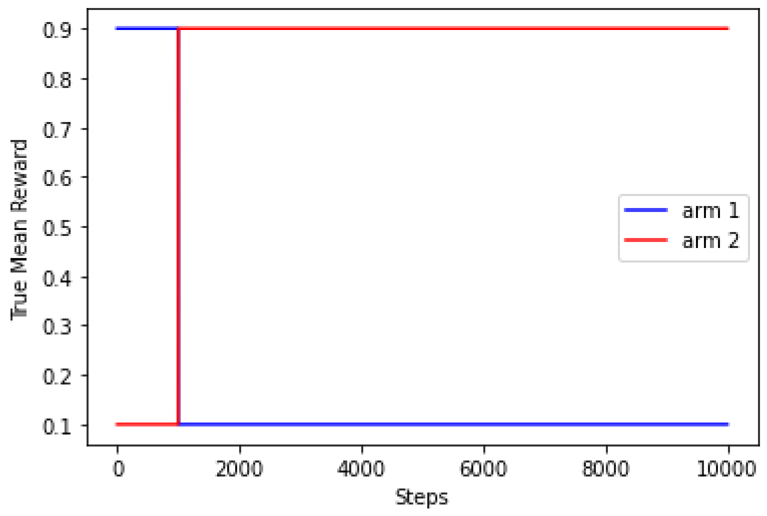
```
function QAC
    Initialise $s$, $\theta$
    Sample $a \sim \pi_\theta$
    for each step do
        Sample reward $r = \mathcal{R}_s^a$; sample transition $s' \sim \mathcal{P}_{s,\cdot}^a$
        Sample action $a' \sim \pi_\theta(s', a')$
        $\delta = r + \gamma Q_w(s', a') - Q_w(s, a)$
        $\theta = \theta + \alpha \nabla_\theta \log \pi_\theta(s, a) Q_w(s, a)$
        $w \leftarrow w + \beta \delta \phi(s, a)$
        $a \leftarrow a', s \leftarrow s'$
    end for
end function
```

2.1. 2-Armed Bandit

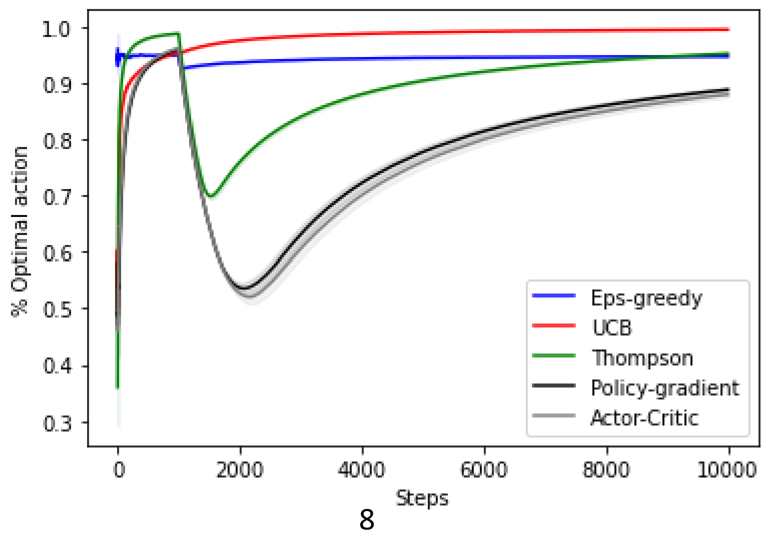
For this part, we consider a single-agent 2-armed bandit with Bernoulli reward functions in three levels: easy, medium, and hard. The specific information about these reward functions will be stated in the following parts.

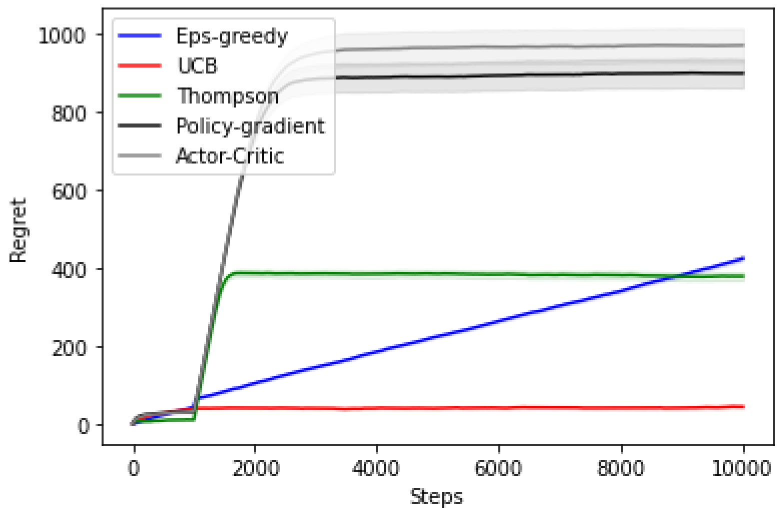
2.1.1. Easy Problem

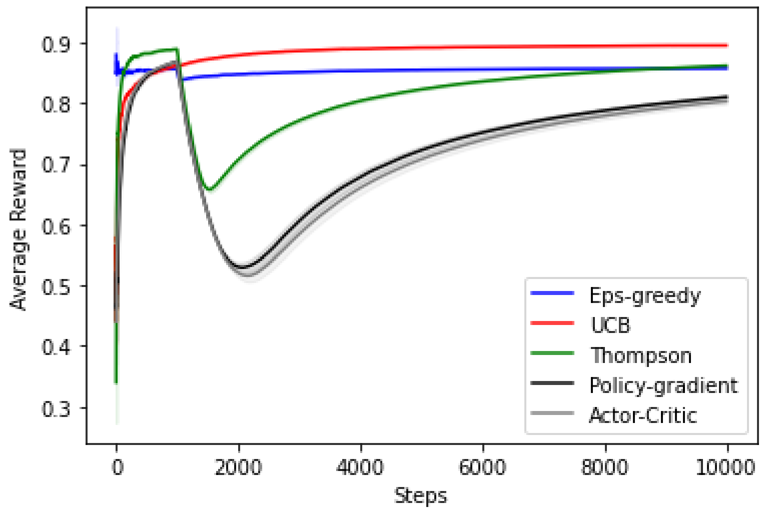
For the easy problem, we consider the mean of Bernoulli distributions,i.e. probability of selecting 1, through the learning as follows:



We use the Epsilon-greedy, Upper-Confidence Bound(UCB), Policy-gradient, Thompson Sampling, and Actor-Critic algorithms. Furthermore, we plot the percent of selecting optimal action, regret, and Average reward for all algorithms in one plot. We considered the learning rate to be 0.1 wherever it was needed. For the epsilon-greedy, we consider epsilon to be 0.1 and non-decaying again due to the non-stationariness of the environment. Furthermore, we use c = 2 in the UCB algorithm. The result is as follows:



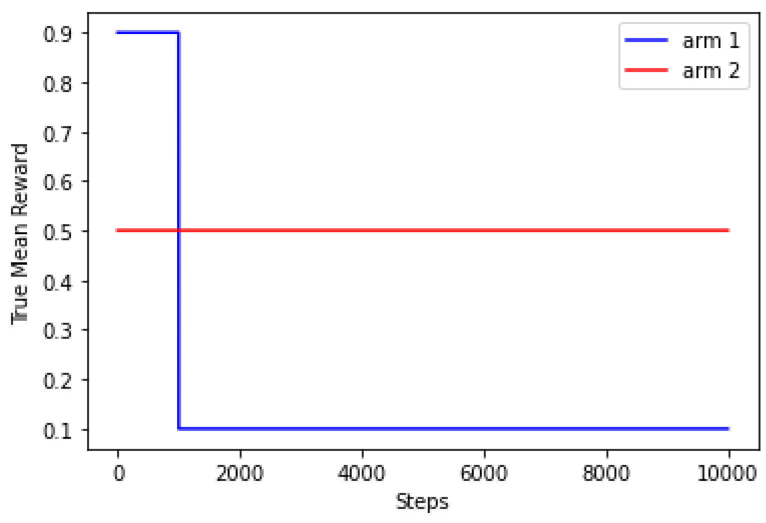




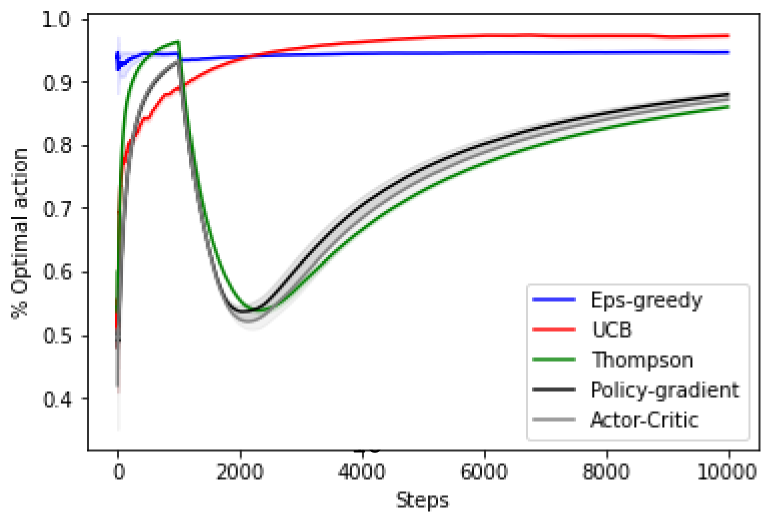
From the above plots, we can see that the UCB algorithm performs best in this easy problem. In the second place, according to the regret measure, is the Thompson sampling. The epsilon-greedy is in third place, and the policy-gradient and actor-critic are in the following places.

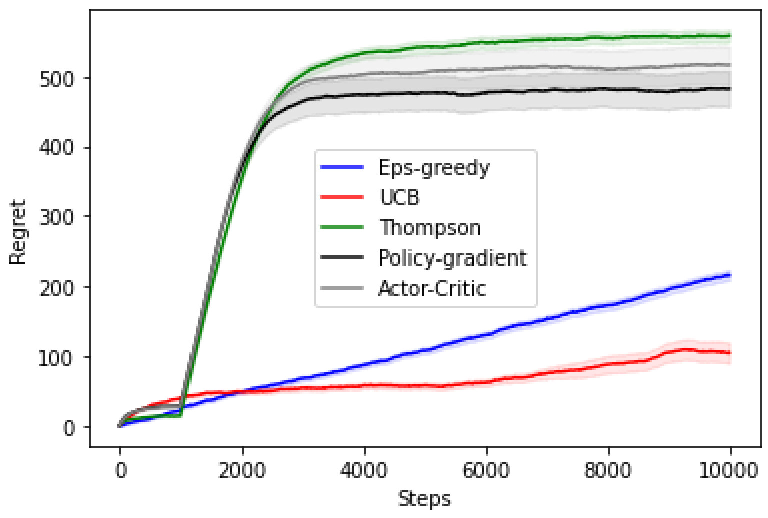
2.1.2. Medium Problem

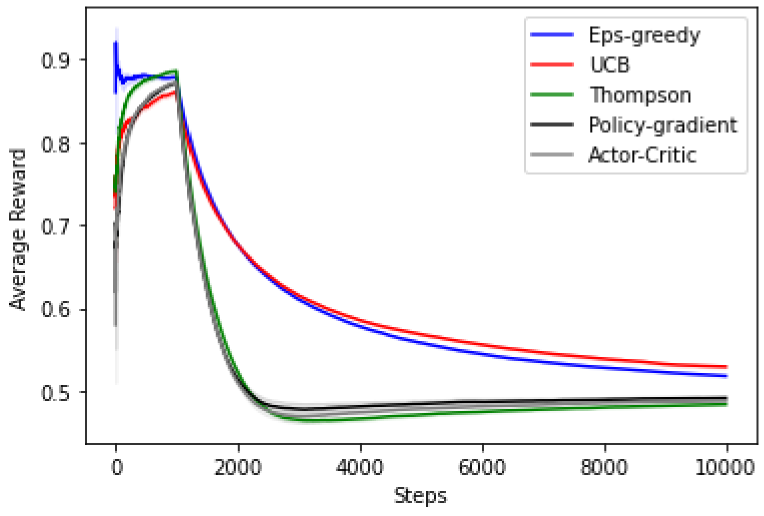
For the medium problem, we consider the mean of Bernoulli distributions,i.e. probability of selecting 1, through the learning as follows:



We use the Epsilon-greedy, Upper-Confidence Bound(UCB), Policy-gradient, Thompson Sampling, and Actor-Critic algorithms. Furthermore, we plot the percent of selecting optimal action, regret, and Average reward for all algorithms in one plot. We considered the learning rate to be 0.1 wherever it was needed. For the epsilon-greedy, we consider epsilon to be 0.1 and non-decaying again due to the non-stationariness of the environment. Furthermore, we use c = 2 in the UCB algorithm. The result is as follows:



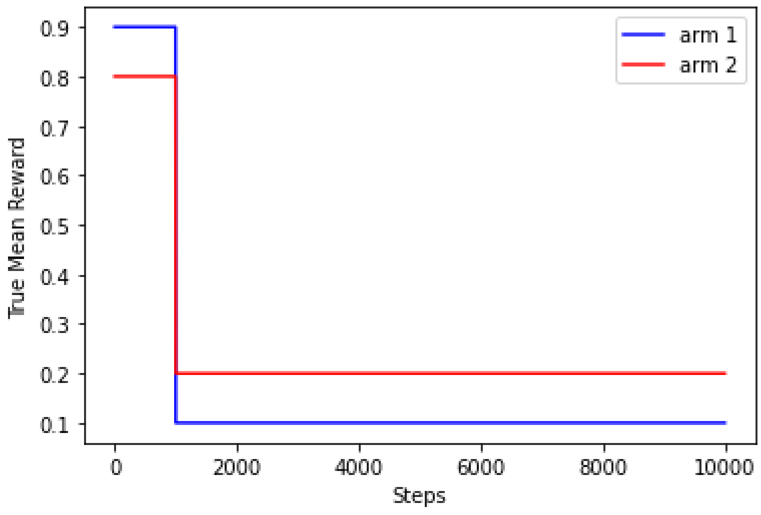




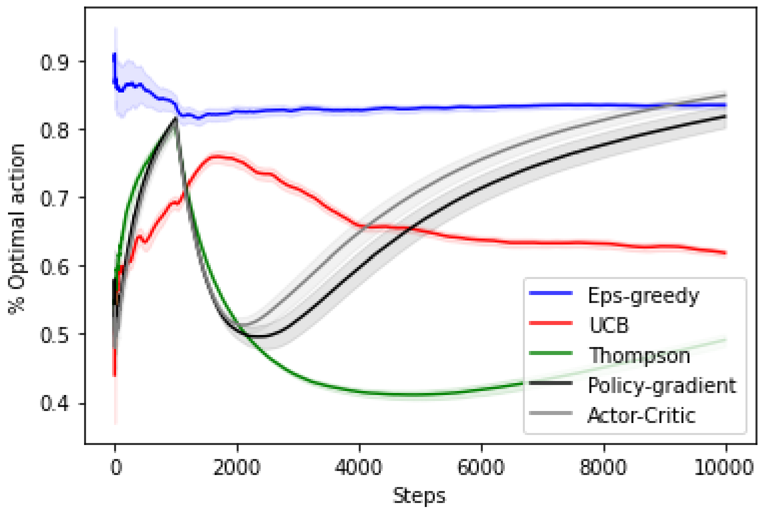
From the above plots, we can see that the UCB algorithm performs best in this medium problem. In the second place, according to the regret measure, is the epsilon-greedy. The Thompson algorithm, policy-gradient, and actor-critic are in third place and have very similar behavior. Notably, in this problem, after trial 1000, the mean of rewards of the best arm is 0.5, as seen in the figure.

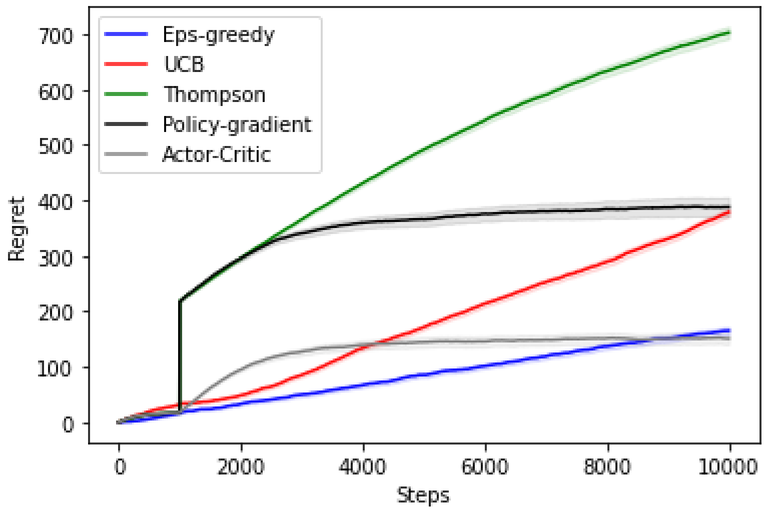
2.1.3. Hard Problem

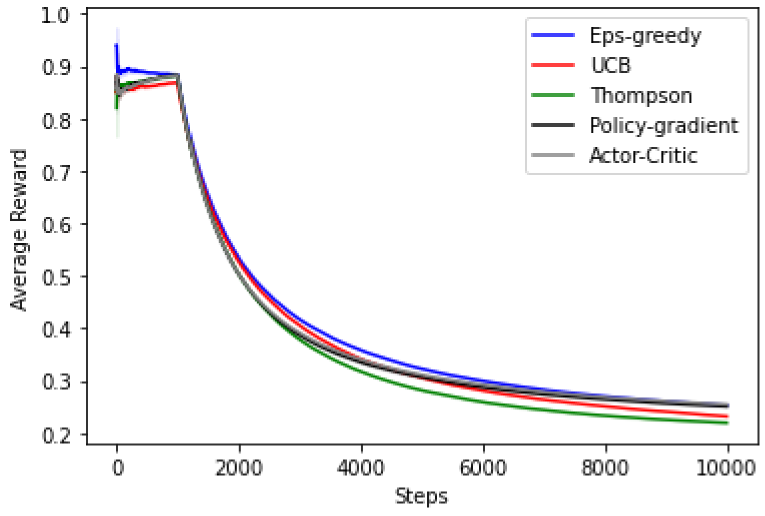
For the hard problem, we consider the mean of Bernoulli distributions,i.e. probability of selecting 1, through the learning as follows:



We use the Epsilon-greedy, Upper-Confidence Bound(UCB), Policy-gradient, Thompson Sampling, and Actor-Critic algorithms. Furthermore, we plot the percent of selecting optimal action, regret, and Average reward for all algorithms in one plot. We considered the learning rate to be 0.1 wherever it was needed. For the epsilon-greedy, we consider epsilon to be 0.1 and non-decaying again due to the non-stationariness of the environment. Furthermore, we use c = 2 in the UCB algorithm. The result is as follows:







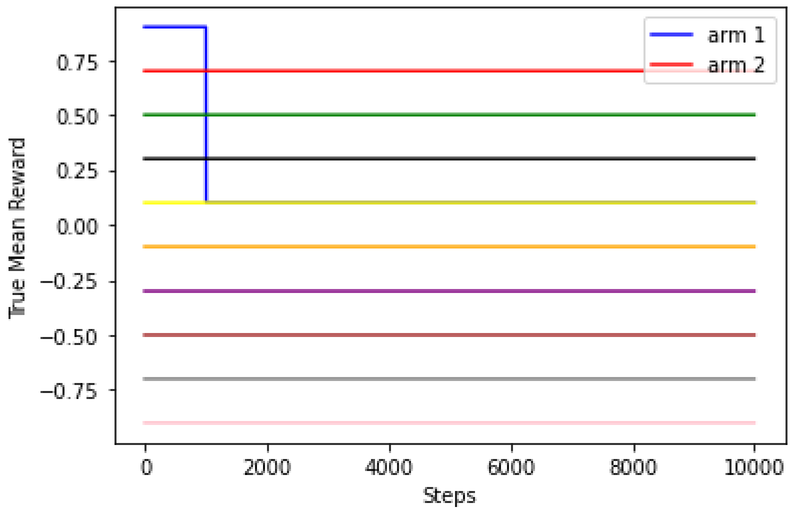
From the above plots, we can see that the Actor-Critic algorithm performs best in this hard problem. In the second place, according to the regret measure, is the epsilon-greedy. The policy gradient and UCB are in third place. Furthermore, the Thompson Algorithm is the last one. It is notable that in this problem, after trial 1000, the mean of rewards of the best arm is 0.2, as seen in the figure.

2.2. 10-Armed Bandit

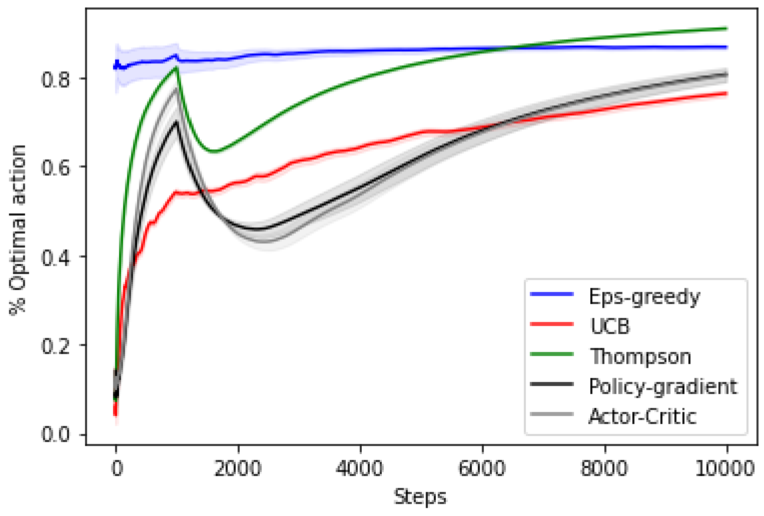
For this part, we consider a single-agent 10-armed bandit with Bernoulli reward functions, in two different levels: medium and hard. The specific information about these reward functions will be stated in the following parts.

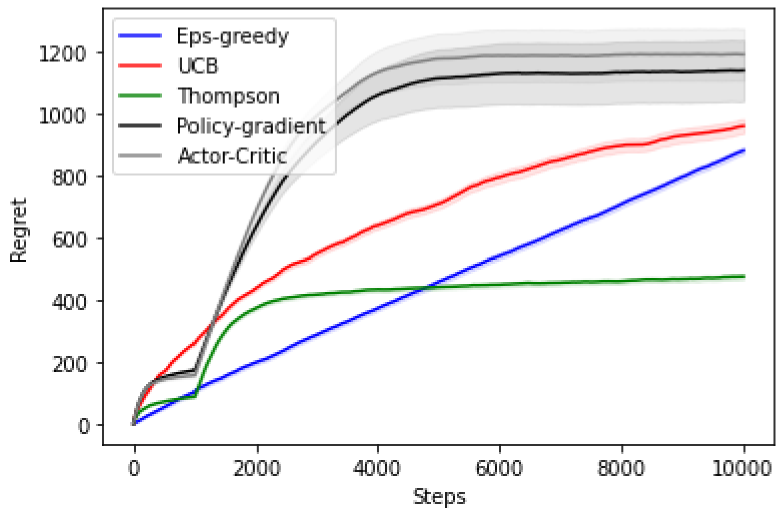
2.2.1. Medium Problem

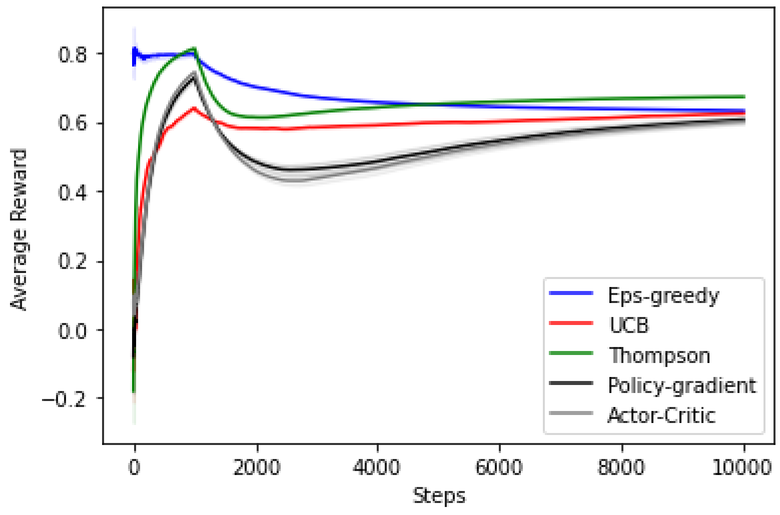
For the medium problem, we consider the mean of Bernoulli distributions,i.e. probability of selecting 1, through the learning as follows:



We use the Epsilon-greedy, Upper-Confidence Bound(UCB), Policy-gradient, Thompson Sampling, and Actor-Critic algorithms. Furthermore, we plot the percent of selecting optimal action, regret, and Average reward for all algorithms in one plot. We considered the learning rate to be 0.1 wherever it was needed. For the epsilon-greedy, we consider epsilon to be 0.1 and non-decaying again due to the non-stationariness of the environment. Furthermore, we use c = 2 in the UCB algorithm. The result is as follows:



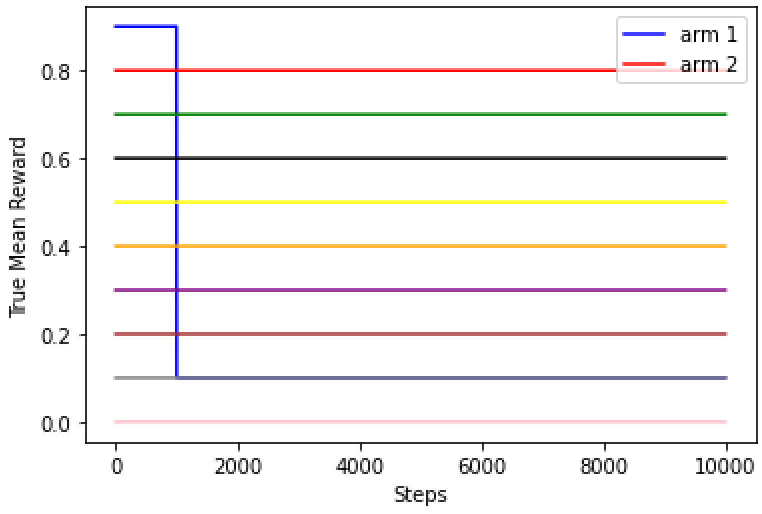




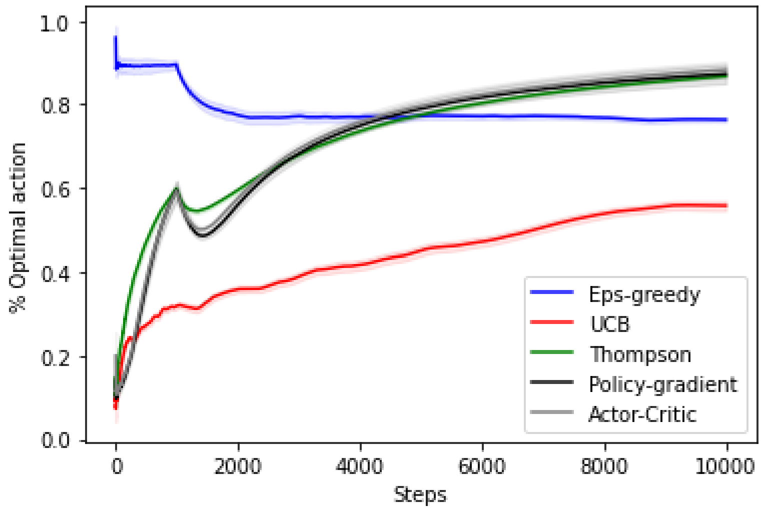
From the above plots, we can see that the Thompson algorithm performs best in this medium problem. In the second place, according to the regret measure, is the epsilon-greedy. Furthermore, UCB is in third place. The policy-gradient and actor-critic are in fourth place and have very similar behavior. It is notable that in this problem, after trial 1000, the mean reward of the best arm is 0.7, as seen in the figure.

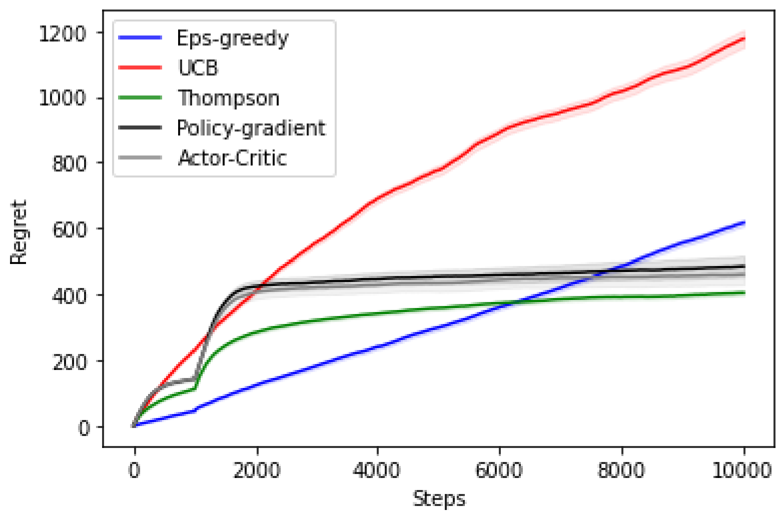
2.2.2. Hard Problem

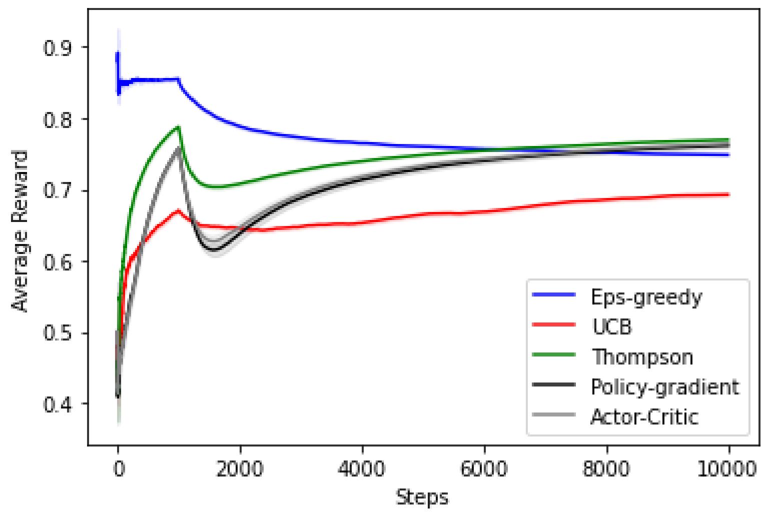
For the hard problem, we consider the mean of Bernoulli distributions,i.e. probability of selecting 1, through the learning as follows:



We use the Epsilon-greedy, Upper-Confidence Bound(UCB), Policy-gradient, Thompson Sampling, and Actor-Critic algorithms. Furthermore, we plot the percent of selecting optimal action, regret, and Average reward for all algorithms in one plot. We considered the learning rate to be 0.1 wherever it was needed. For the epsilon-greedy, we consider epsilon to be 0.1 and non-decaying again due to the non-stationariness of the environment. Furthermore, we use c = 2 in the UCB algorithm. The result is as follows:





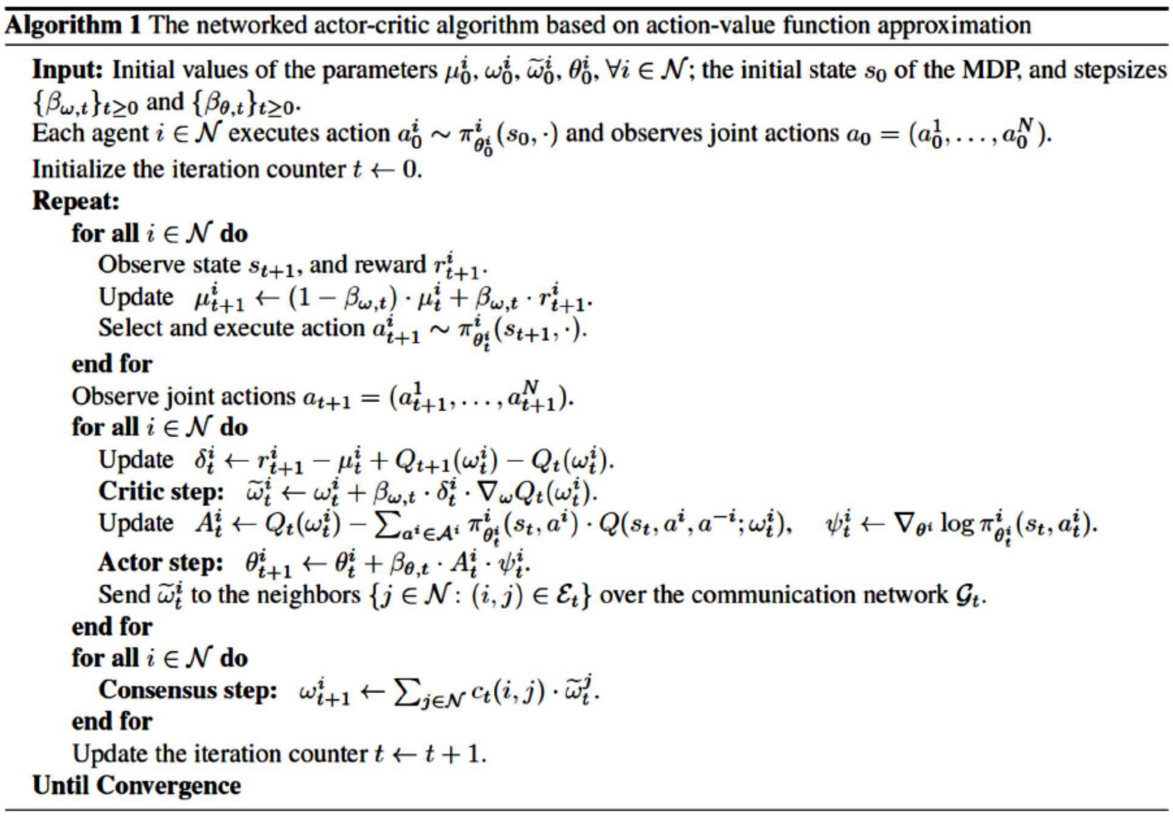


From the above plots, we can see that the Actor-Critic, policy-gradient, and Thompson sampling is in the first place and have a very similar performance. The epsilon-greedy and UCB are in the following ranks. It is notable that in this problem, after trial 1000, the mean of rewards of the best arm is 0.8, as seen in the figure.

Section 3) Multi-Agent Multi-Armed Bandit

This section considers different single-agent multi-armed bandits with 2 and 10 arms. In both cases, we design environments with different difficulty levels, i.e. discriminability of rewards. In this part, we use Joint Action Learners(JAL), Free Maximum Q-value(FMQ), Distributed Q-learning, and Multi-agent Actor-Critic Algorithm. We trained each algorithm in each environment for 10,000 steps in all parts. We repeat this procedure 50 times and plot the mean and standard deviation. Also, it is notable, that we consider moving averages in all algorithms whenever possible due to the non-stationariness of our environment.

For the Multi-agent Actor-Critic Algorithm, we use the following algorithm:



Algorithm 1 The networked actor-critic algorithm based on action-value function approximation

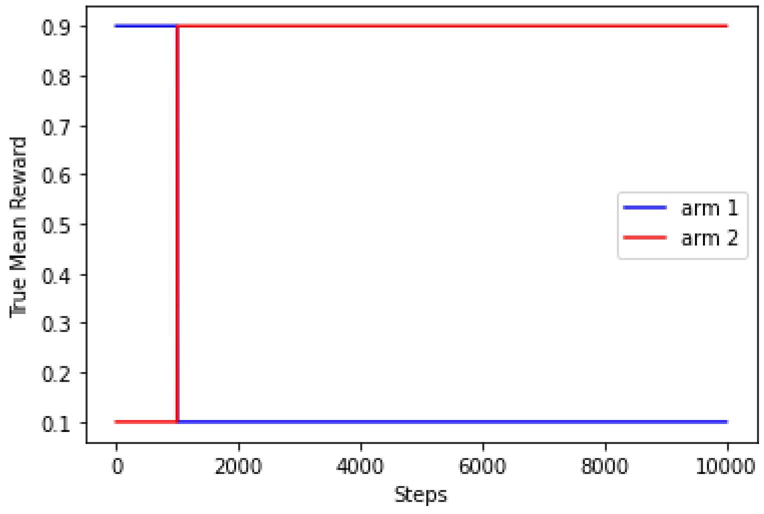
```
Input: Initial values of the parameters $\mu_0^i, \omega_0^i, \widetilde{\omega}_0^i, \theta_0^i, \forall i \in \mathcal{N}$; the initial state $s_0$ of the MDP, and stepsizes $\{\beta_{\omega,t}\}_{t\geq 0}$ and $\{\beta_{\theta,t}\}_{t\geq 0}$.
Each agent $i \in \mathcal{N}$ executes action $a_0^i \sim \pi_{\theta_0^i}^i(s_0, \cdot)$ and observes joint actions $a_0 = (a_0^1, \ldots, a_0^N)$.
Initialize the iteration counter $t \leftarrow 0$.
Repeat:
    for all $i \in \mathcal{N}$ do
        Observe state $s_{t+1}$, and reward $r_{t+1}^i$.
        Update  $\mu_{t+1}^i \leftarrow (1 - \beta_{\omega,t}) \cdot \mu_t^i + \beta_{\omega,t} \cdot r_{t+1}^i$.
        Select and execute action $a_{t+1}^i \sim \pi_{\theta_t^i}^i(s_{t+1}, \cdot)$.
    end for
    Observe joint actions $a_{t+1} = (a_{t+1}^1, \ldots, a_{t+1}^N)$.
    for all $i \in \mathcal{N}$ do
        Update  $\delta_t^i \leftarrow r_{t+1}^i - \mu_t^i + Q_{t+1}(\omega_t^i) - Q_t(\omega_t^i)$.
        Critic step:  $\widetilde{\omega}_t^i \leftarrow \omega_t^i + \beta_{\omega,t} \cdot \delta_t^i \cdot \nabla_\omega Q_t(\omega_t^i)$.
        Update  $A_t^i \leftarrow Q_t(\omega_t^i) - \sum_{a^i \in \mathcal{A}^i} \pi_{\theta_t^i}^i(s_t, a^i) \cdot Q(s_t, a^i, a^{-i}; \omega_t^i)$,    $\psi_t^i \leftarrow \nabla_{\theta^i} \log \pi_{\theta_t^i}^i(s_t, a_t^i)$.
        Actor step:  $\theta_{t+1}^i \leftarrow \theta_t^i + \beta_{\theta,t} \cdot A_t^i \cdot \psi_t^i$.
        Send $\widetilde{\omega}_t^i$ to the neighbors $\{j \in \mathcal{N} : (i,j) \in \mathcal{E}_t\}$ over the communication network $\mathcal{G}_t$.
    end for
    for all $i \in \mathcal{N}$ do
        Consensus step:  $\omega_{t+1}^i \leftarrow \sum_{j \in \mathcal{N}} c_t(i,j) \cdot \widetilde{\omega}_t^j$.
    end for
    Update the iteration counter $t \leftarrow t + 1$.
Until Convergence
```

3.1. 2-Armed Bandit

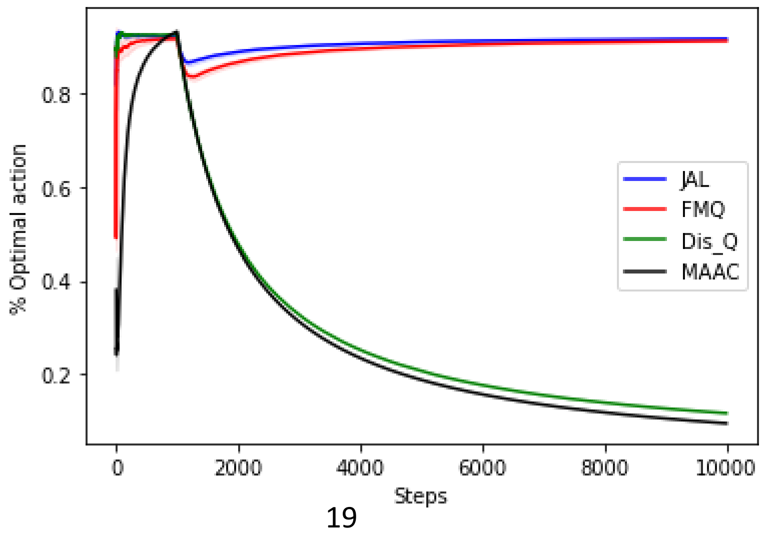
For this part, we consider a 2-agents 2-armed bandit with Bernoulli reward functions in three levels: easy, medium, and hard. The specific information about these reward functions will be stated in the following parts. We consider these agents to cooperate with others to solve the bandit problem. Therefore, they received the same reward signal through learning.

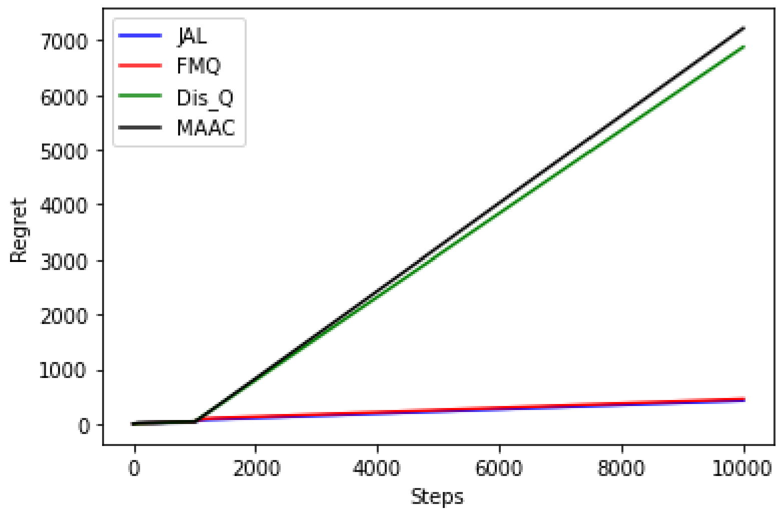
3.1.1. Easy Problem

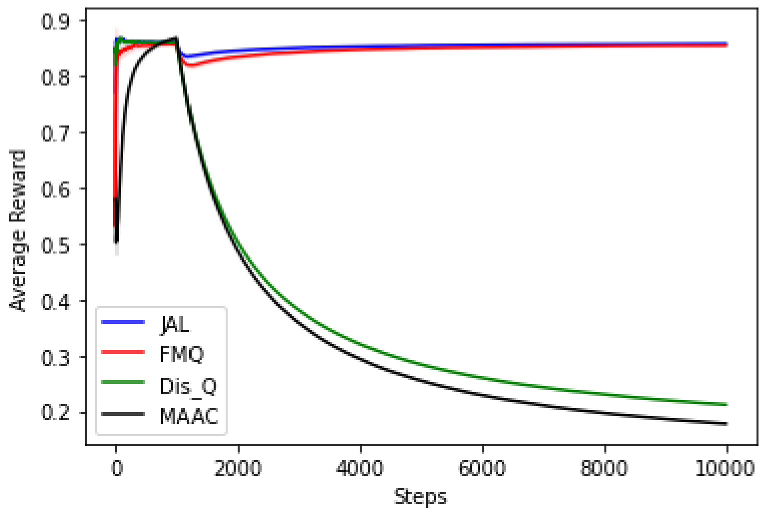
For the easy problem, we consider the mean of Bernoulli distributions,i.e. probability of selecting 1, through the learning as follows:



We use Joint Action Learners(JAL), Free Maximum Q-value(FMQ), Distributed Q-learning, and Multi-agent Actor-Critic Algorithm. Furthermore, we plot the percent of selecting optimal action, regret, and Average reward for all algorithms in one plot. We considered the learning rate to be 0.1 wherever it was needed. For the JAL and FMQ, we consider epsilon to be 0.1 and non-decaying again due to the non-stationariness of the environment. The result is as follows:



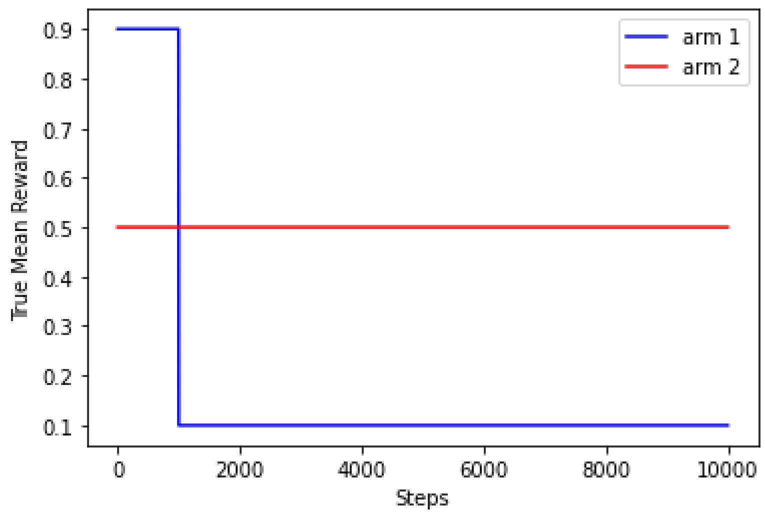




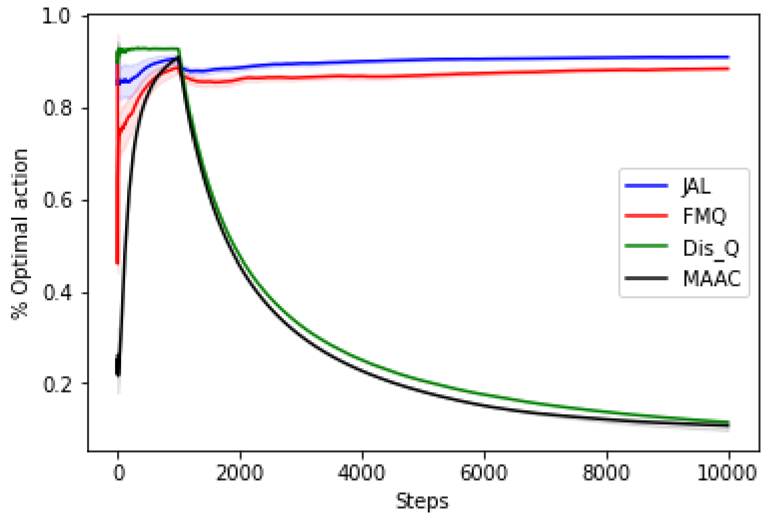
The above plots show that the Distributed Q-learning and Multi-Agent Actor-Critic(MAAC) algorithms cannot cope with non-stationariness, and their performances drop catastrophically. However, the JAL and FQM algorithms behave very well and adapt to the change.

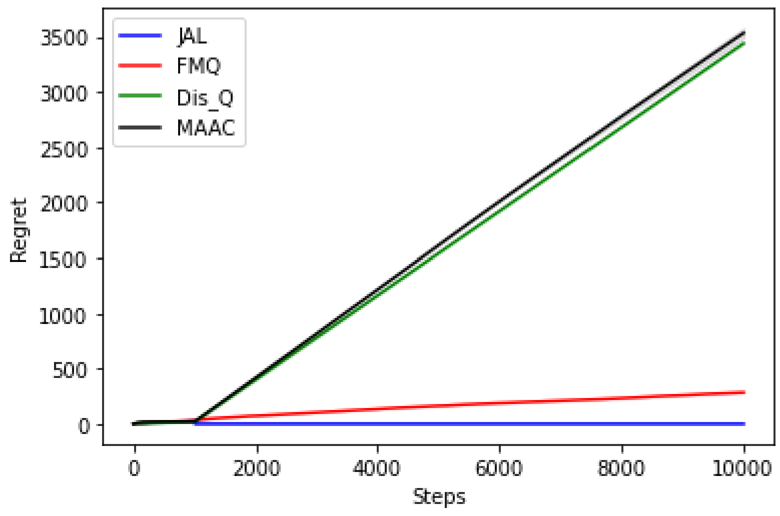
3.1.2. Medium Problem

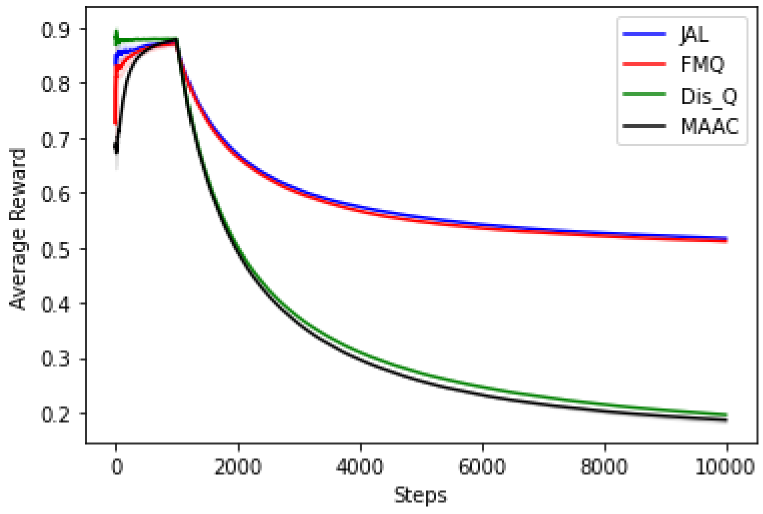
For the medium problem, we consider the mean of Bernoulli distributions,i.e. probability of selecting 1, through the learning as follows:



We use Joint Action Learners(JAL), Free Maximum Q-value(FMQ), Distributed Q-learning, and Multi-agent Actor-Critic Algorithm. Furthermore, we plot the percent of selecting optimal action, regret, and Average reward for all algorithms in one plot. We considered the learning rate to be 0.1 wherever it was needed. For the JAL and FMQ, we consider epsilon to be 0.1 and non-decaying again due to the non-stationariness of the environment. The result is as follows:



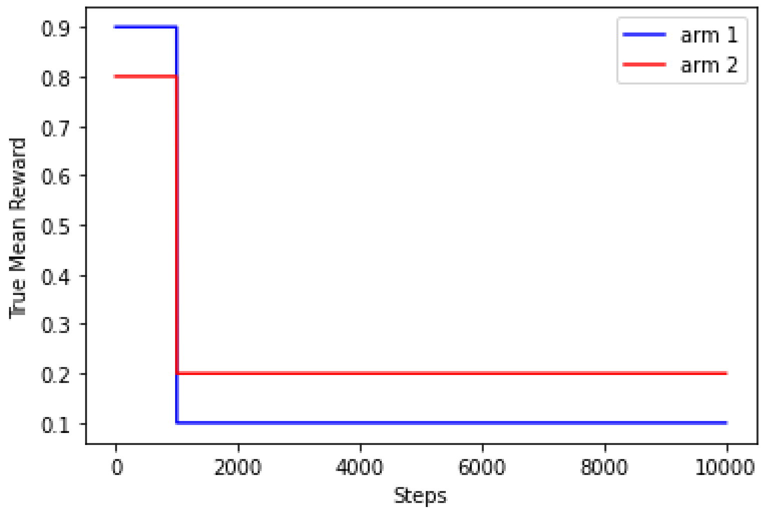




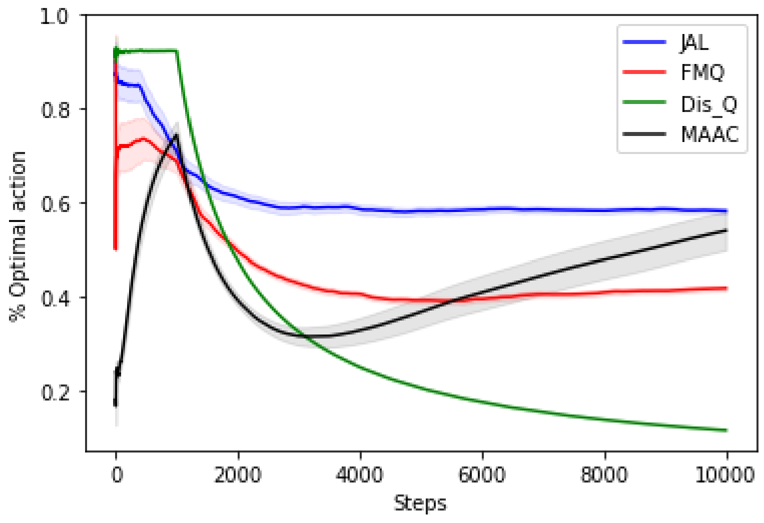
The above plots show that the Distributed Q-learning and Multi-Agent Actor-Critic(MAAC) algorithms cannot cope with non-stationariness, and their performances drop catastrophically. However, the JAL and FQM algorithms behave very well and adapt to the change. It is notable that in this problem, after trial 1000, the mean of rewards of the best arm is 0.5, as seen in the figure.

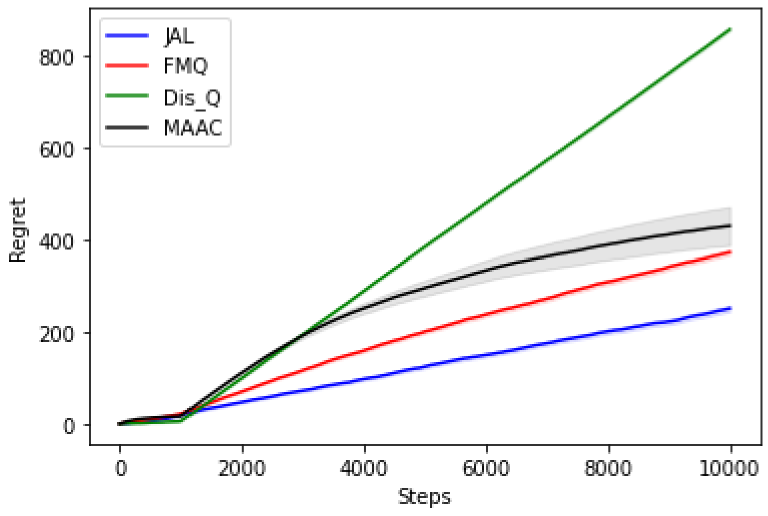
3.1.3. Hard Problem

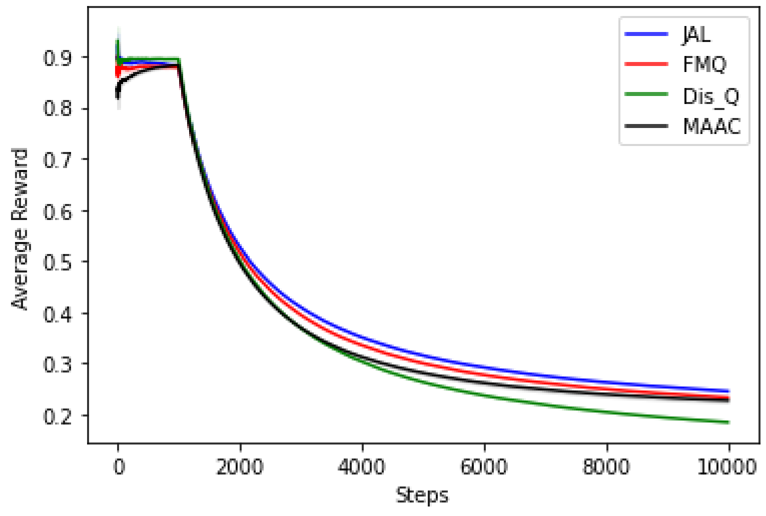
For the hard problem, we consider the mean of Bernoulli distributions,i.e. probability of selecting 1, through the learning as follows:



We use Joint Action Learners(JAL), Free Maximum Q-value(FMQ), Distributed Q-learning, and Multi-agent Actor-Critic Algorithm. Furthermore, we plot the percent of selecting optimal action, regret, and Average reward for all algorithms in one plot. We considered the learning rate to be 0.1 wherever it was needed. For the JAL and FMQ, we consider epsilon to be 0.1 and non-decaying again due to the non-stationariness of the environment. The result is as follows:







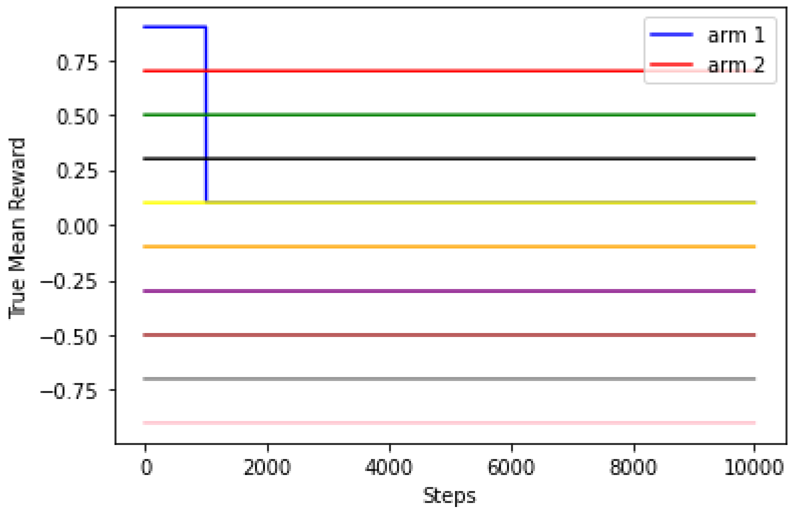
From the above plots, we can see that the Distributed Q-learning algorithm cannot cope with non-stationariness, and its performance drops catastrophically. However, the JAL, FQM, and MAAC algorithms have reasonably good performance and adapt to the change. It is notable that in this problem, after trial 1000, the mean of rewards of the best arm is 0.2, as seen in the figure.

3.2. 10-Armed Bandit

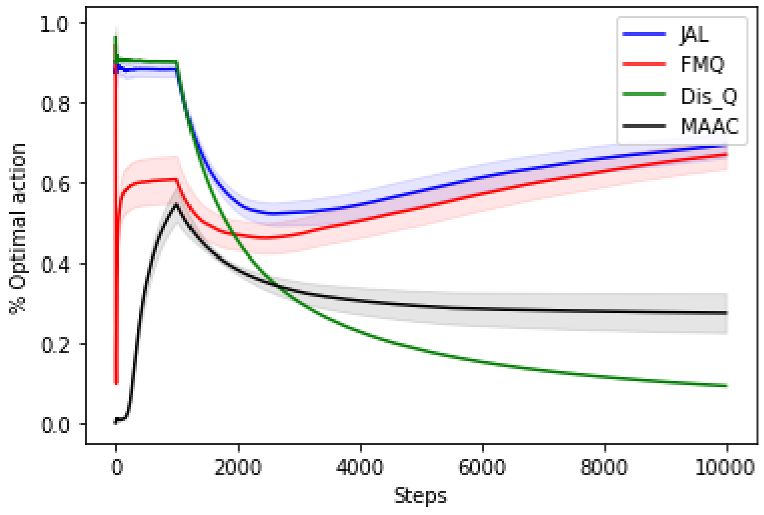
For this part, we consider a 2-agents 10-armed bandit with Bernoulli reward functions, in two different levels: medium and hard. The specific information about these reward functions will be stated in the following parts.

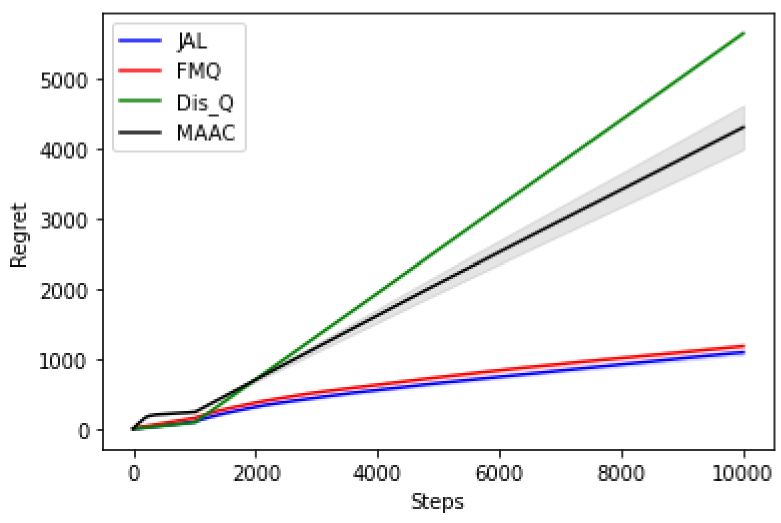
3.2.1. Medium Problem

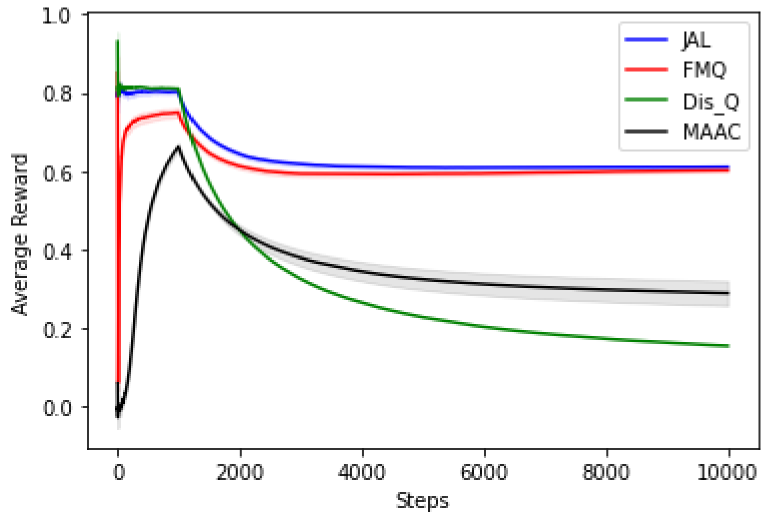
For the medium problem, we consider the mean of Bernoulli distributions,i.e. probability of selecting 1, through the learning as follows:



We use Joint Action Learners(JAL), Free Maximum Q-value(FMQ), Distributed Q-learning, and Multi-agent Actor-Critic Algorithm. Furthermore, we plot the percent of selecting optimal action, regret, and Average reward for all algorithms in one plot. We considered the learning rate to be 0.1 wherever it was needed. For the JAL and FMQ, we consider epsilon to be 0.1 and non-decaying again due to the non-stationariness of the environment. The result is as follows:



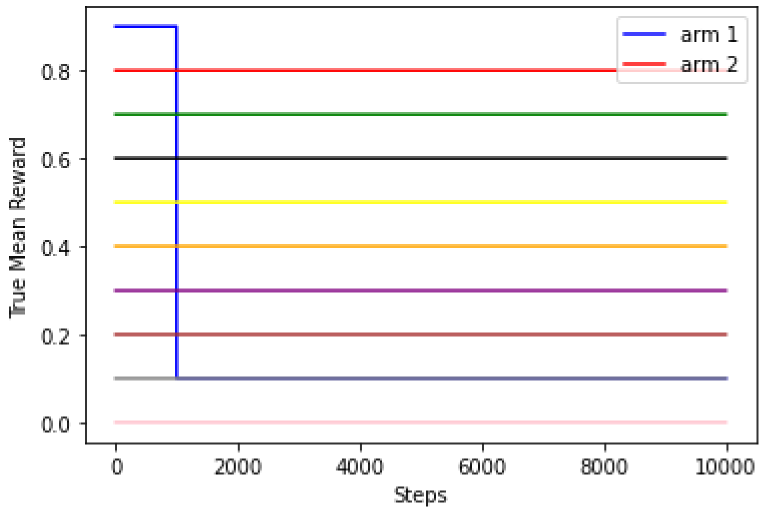




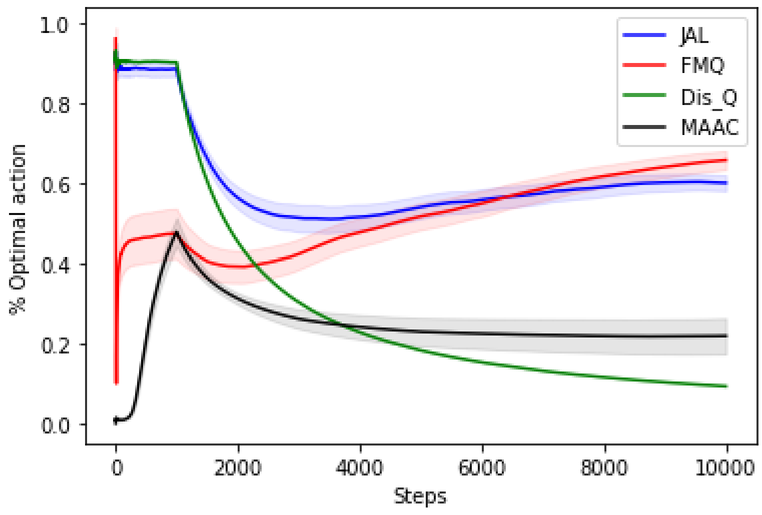
The above plots show that the Distributed Q-learning and Multi-Agent Actor-Critic(MAAC) algorithms cannot cope with non-stationariness, and their performances drop catastrophically. However, the JAL and FQM algorithms behave very well and adapt to the change. It is notable that in this problem, after trial 1000, the mean reward of the best arm is 0.7, as seen in the figure.

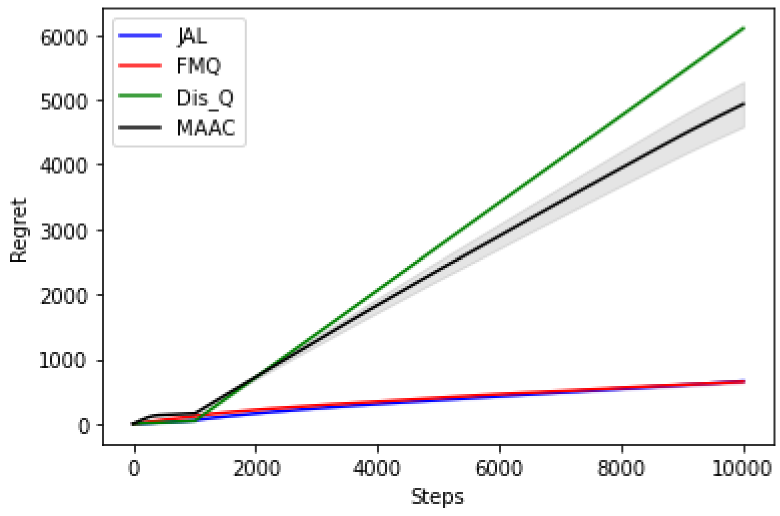
3.2.2. Hard Problem

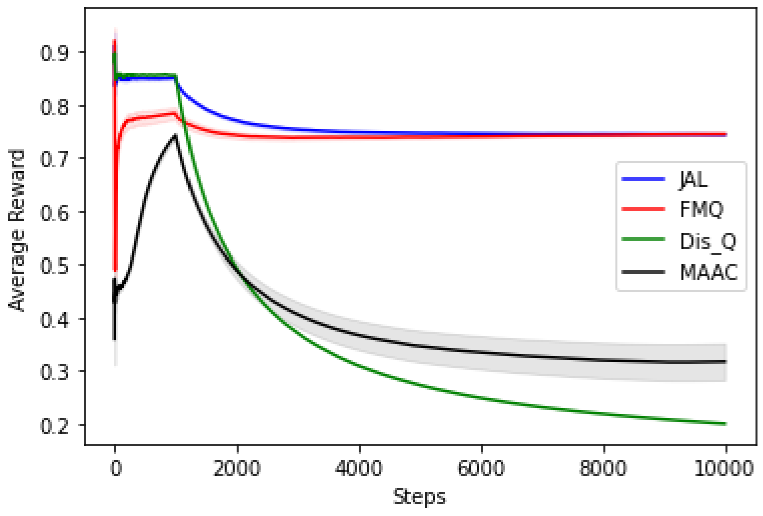
For the hard problem, we consider the mean of Bernoulli distributions,i.e. probability of selecting 1, through the learning as follows:



We use Joint Action Learners(JAL), Free Maximum Q-value(FMQ), Distributed Q-learning, and Multi-agent Actor-Critic Algorithm. Furthermore, we plot the percent of selecting optimal action, regret, and Average reward for all algorithms in one plot. We considered the learning rate to be 0.1 wherever it was needed. For the JAL and FMQ, we consider epsilon to be 0.1 and non-decaying again due to the non-stationariness of the environment. The result is as follows:







The above plots show that the Distributed Q-learning and Multi-Agent Actor-Critic(MAAC) algorithms cannot cope with non-stationariness, and their performances drop catastrophically. However, the JAL and FQM algorithms behave very well and adapt to the change. It is notable that in this problem, after trial 1000, the mean of rewards of the best arm is 0.8, as seen in the figure.

Section 4) Conclusion

In this project, We experimented with a non-stationary multi-armed bandit in 4 different scenarios:

1) Single-Agent 2-armed bandit(Easy, Medium, Hard)
2) Single-Agent 10-armed bandit(Medium, Hard)
3) Multi-Agent 2-armed bandit(Easy, Medium, Hard)
4) Multi-Agent 10-armed bandit(Medium, Hard)

In conclusion, in the first part, we saw that the Thompson sampling algorithm performs well when the reward distributions are sufficiently discriminant. However, also it performs very poorly in close distributions. Also, this behavior happened for the UCB agents but has lower sensitivity. Contrary, the policy-gradient and Actor-Critic had poor performance in easy and medium problems but were better on the complicated problem. The epsilon-greedy, since it maintains exploration has a pretty good performance in all scenarios.

In the second part, we saw that the Thompson sampling algorithm performs best in both cases. Also, the UCB agent performs well when the reward distributions are sufficiently discriminant, but it also performs poorly in close distributions. Contrary, the policy-gradient and Actor-Critic had poor performance in easy and medium problems but were better on the complicated problem. The epsilon-greedy, since it maintains exploration has a pretty good performance in all scenarios.

In the third part, The Distributed Q-learning algorithm does not work well with non-stationarity, and its performance drops catastrophically in both cases.
On the other hand, the JAL and FQM algorithms perform reasonably well and adapt to the change. In easy and medium problems, the MAAC algorithm performed poorly, whereas it performed better in hard problems.

We find the same results in the fourth part, but in this case, the MAAC functioned poorly in all cases.

References:

[1] Lattimore, Tor, and Csaba Szepesvári. Bandit algorithms. Cambridge University Press, 2020.

[2] Besbes, Omar, Yonatan Gur, and Assaf Zeevi. "Stochastic multi-armed-bandit problem with non-stationary rewards." Advances in neural information processing systems 27 (2014): 199-207.

[3] Bouneffouf, Djallel, and Irina Rish. "A survey on practical applications of multi-armed and contextual bandits." arXiv preprint arXiv:1904.10040 (2019).

[4] Zhang, Kaiqing, et al. "Fully decentralized multi-agent reinforcement learning with networked agents." International Conference on Machine Learning. PMLR, 2018.