Thompson Sampling Programming Report

Nicholas Chiapputo Hua Sun

February 28, 2020

1 Introduction

Reinforcement learning is a class of problems that focuses on an agent that learns behavior through trial-and-error with its environment. One of the main strategies for tackling this class of problems is to use statistical techniques to determine the value of taking an action given the state of the environment. This strategy is known as the exploration-exploitation trade-off and involves deciding when to explore less understood actions that may be optimal, or exploit actions that are the best known given prior information [1].

1.1 The Multi-Armed Bandit Problem

A fundamental problem in reinforcement learning is the multi-armed bandit problem. It is also often referred to as the K-armed bandit problem as there are K arms, each representing a separate reward distribution. A popular analogy is to relate each arm to a different slot machine in a casino (colloquially known as a one-armed bandit). When each slot machine is played (an arm is sampled), a reward is randomly chosen from the distribution for that machine (the slot machine pays out the winnings). This problem is an elegant way of viewing the exploration-exploitation trade-off. The algorithm must converge on an optimal arm quickly so as to minimize the cumulative regret. However, it must also be wary of unintentionally converging on a sub-optimal arm, resulting in even worse consequences.

The main goal of an algorithm in the multi-armed bandit problem is to minimize the regret of the actions, determined by the difference in the expected reward and the observed reward if the optimal arm is played every time. The **reward** of a system refers to the result of choosing a specific action. An action is the equivalent to pulling an arm on the bandit machine. The terms action and arm are used interchangeably in this report. In a system where the reward distributions are Bernoulli, the reward for choosing an action can be either 0 or 1. The reward distributions are given a label in $[K] \triangleq \{1, \ldots, K\}$ and its mean reward is given by μ_i where $i \in [K]$. The arm selected at time t is given by $i(t)$. The optimal arm has the highest mean reward and is given by

$$\mu^* = \max_{i \in [K]} \mu_i. \tag{1}$$

If the algorithm knew the best arm, then the best arm would be selected every time. This is the optimal scenario. To determine the performance of an algorithm, the **regret** of its actions is measured as the difference between the mean reward of an optimal action, the one with the highest mean, and the mean reward of the selected action (see [1, 4, 5]). **Per-period regret** is the amount of regret incurred in a single trial, or a single pull of an arm $i \in [K]$. The empirical per-period regret is given by:

$$R_1 = \mu^* - r(i(t)) \tag{2}$$

where $i(t)$ is the action chosen at time t, and $r(i(t))$ is the reward for choosing action i at time t. The cumulative regret over T time slots is the sum of the expected value of per-period regrets [5, p. 11]:.

$$R_T = T\mu^* - \mathbb{E}\Big[\sum_{t=1}^{T} r(i(t))\Big] \tag{3}$$

When we simulate the multi-armed bandit algorithm, we approximate the expected regret value by the average of empirical regret values over a number of Monte-Carlo simulations, S. The empirical regret value is experimentally determined so that a large number of experiments of S simulations all converge towards the expected cumulative regret value. Therefore the empirical cumulative regret is calculated as follows and is used to represent the estimate of the expected cumulative regret R_T.

$$\widehat{R}_T = T\mu^* - \frac{1}{S}\sum_{j=1}^{S}\sum_{t=1}^{T} r^j(i(t)) \tag{4}$$

where the superscript j represents the simulation index, i.e., when we run the same algorithm for the j^{th} time.

In the elementary model, the algorithm begins with no knowledge of the reward distributions for the arms. The algorithm must then explore the arms to decide on an optimal distribution by estimating μ^* and then exploit that distribution to minimize the regret and maximize the reward. In practical applications, some amount of knowledge may be known about the distributions, enabling the implementation to be tuned by adjusting the starting parameters (see e.g., [9]).

1.2 Thompson Sampling

One of the earliest algorithms for the multi-armed bandit problem is **Thompson Sampling**, introduced in [2]. While relatively unnoticed in the literature, Thompson Sampling has seen increasing popularity over the last decade due to its excellent empirical performance, first shown in [6] and the empirical performance is later verified from a theoretical point of view. Specifically, the regret achieved by Thompson Sampling in the Bernoulli bandit has been shown to be $O(\Big[(\sum_{i=2}^{K} \frac{1}{\Delta_i^2})^2\Big])$ in [3] where $\Delta_i = \mu^* - \mu_i$ is the difference between the best arm and the i-th arm and the achieved regret matches a known lower bound for any bandit algorithms [10]. Therefore, Thompson Sampling is optimal.

In Thompson Sampling, a prior distribution is created for each arm to represent the potential reward distribution. In the traditional Bernoulli bandit problem, in which each reward distribution is modeled by a Bernoulli distribution, the prior distributions for each arm are given by beta distributions with parameters (α, β). This model can be generalized to use any reward distribution, however the Bernoulli bandit is the most fundamental. For each arm i, the value W_i records the number of "wins", or the number of times a sampling of the reward distribution returns a 1, while the value D_i records the total number of times the reward distribution has been sampled.

Algorithm 1 Thompson Sampling for K-armed Bernoulli bandit

```
$\alpha = 1, \beta = 1$
$W_i = 0,\ D_i = 0 \quad \forall\ i \in [K]$
for $t = 1, \ldots, T$ do
    for $k = 1, \ldots, K$ do
        $\theta_k \leftarrow \text{Beta}(W_k + \alpha,\ D_k - W_k + \beta)$
    end for
    $i \leftarrow \text{argmax}_{i \in [K]}\, \theta_i$
    $D_i \leftarrow D_i + 1$
    Observe reward $r$ from distribution $i$
    if $r = 1$ then $W_i \leftarrow W_i + 1$
    end if
end for
```

The Thompson Sampling algorithm for the K-armed Bernoulli bandit is given in Algorithm 1. Thompson Sampling is a randomized algorithm, so the cumulative regret can vary from one simulation to another. To measure the performance of the algorithm, the cumulative regret is averaged over S simulations, i.e., the algorithm is run S times. The value of S is chosen such that the averaged empirical cumulative regret value $\widehat{R}_T$ converges towards the expected cumulative regret value R_T. The effect of the S value on the regret is explored in Section 2.1.

1.3 Motivation

In current literature, Thompson Sampling has been evaluated theoretically. In many empirical evaluations of the algorithm, the algorithm is said to be tuned for maximum performance. However, the tuned parameters are generally not defined, so that it is not fully clear how the results are obtained. To understand better how to measure the performance of Thompson Sampling, we will adjust the parameters and see the outcomes. The goals of this report include the following items.

1. Implement Thompson Sampling in the C programming language (see [11])

2. Empirically examine the effect of altering the simulation parameters

3. Create a rule of thumb for which algorithm to select given environmental parameters

4. Adjust the number of samples Thompson Sampling takes from the prior distribution to lower cumulative regret

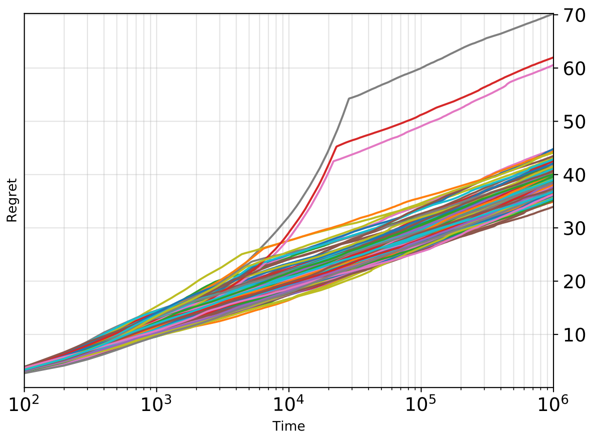
In Section 2, we measure the effect of changing a parameter on the cumulative regret by fixing all remaining parameters. Since there are multiple algorithms that solve the multi-armed bandit problem, and none of them are the best for all situations, it is important to also understand when to use which algorithm. In this report, we will focus on three classic bandit algorithms (many more variants are available in the literature and more to be invented): Thompson Sampling, Upper Confidence Bound, and Δ-Greedy. In Section 3, we summarize a simple rule of thumb for when to pick each algorithm based on the system parameters. In Section 4, we aim to improve the performance of Thompson Sampling by taking multiple samples (instead of one) from each prior distribution and the performance of such multi-sampled version of Thompson Sampling is plotted.

2 Effect of Parameters on Regret

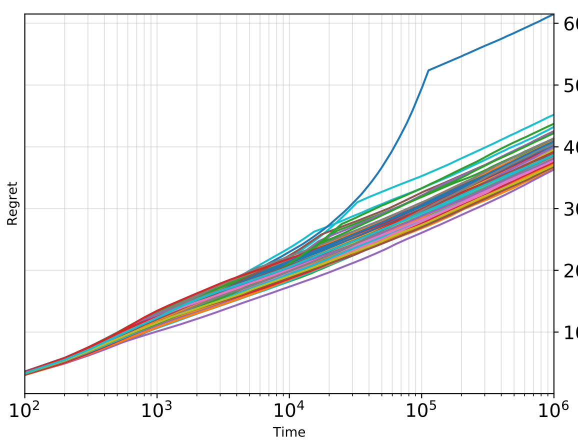
2.1 Changing Number of Simulations

The number of simulations in an experiment is the number of times the algorithm is executed and for each execution, the algorithm runs from time slot 1 to time slot T. This parameter of the number of simulations, naturally, does not come into play in a practical application where there is typically only one simulation. That is, the algorithm must choose to explore or exploit only once without the added benefit of averaging results over multiple runs of the algorithm. However, this variable can be adjusted when we design and test a bandit algorithm.

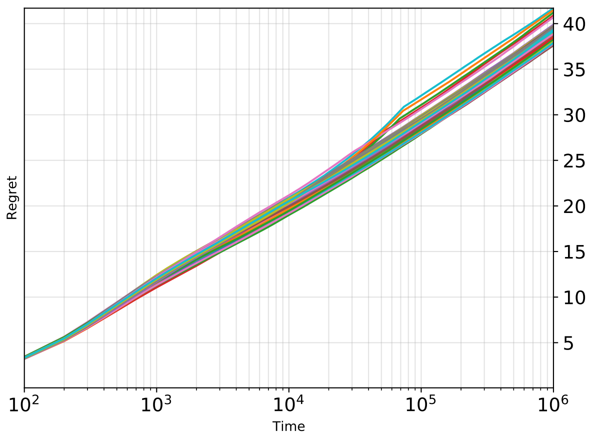
In our implementation of the Thompson Sampling algorithm, we set the number of simulations as S. By averaging the results of each of these S simulations, the averaged regret becomes a fairly accurate representation of the expected regret of the Thompson Sampling algorithm (i.e., we use $\widehat{R}_T$ in (4) as the approximation of R_T in (3)). This allows the results of an experiment to be consistent and reduce the probability of outliers so that the analysis of the plots is consistent. By increasing the number of simulations, the plots of average regret over time become smoother and less prone to sharp changes in slope as the regret tends to converge. Conversely, decreasing the number of simulations can lead to a more jagged, less clear plot where the randomness of the algorithm causes uncertainty.



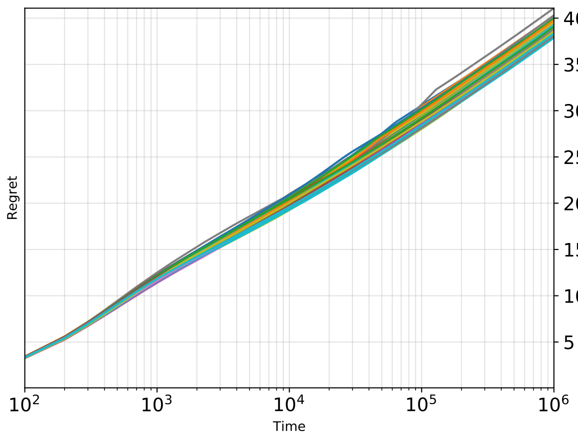
(a)



(b)



(c)



(d)

Figure 1: Regret over time of 200 runs of the Thompson Sampling algorithm. Each run consists of $S =$ (a) 100, (b) 500, (c) 2,500, and (d) 5,000 total simulations that are averaged to produce one curve of a different color in the plot. The number of arms K is 2 and the difference between the highest mean reward and the remaining mean rewards ϵ is 0.1.

Figure 1 shows the simulation results when the number of simulations S is set as 100, 500, 2,500, and 5,000. For each plot, the algorithm is run 200 times to show better how often outliers occur. It can be seen from Figure 1a and Figure 1b that when S is small, the regret curves do not always converge. Out of 200 runs, there are three deviating results with 100 simulations and one deviating result with 500 simulations. Comparing with Figure 1c and Figure 1d, it can be seen that increasing the number of simulations decreases the final regret spread, leading to a stronger convergence.

However, there are some diminishing returns. As the number of simulations increases, the overall execution time increases linearly, so some trade-off between convergence and execution time must be taken into account when testing an algorithm.

2.2 Changing Number of Arms

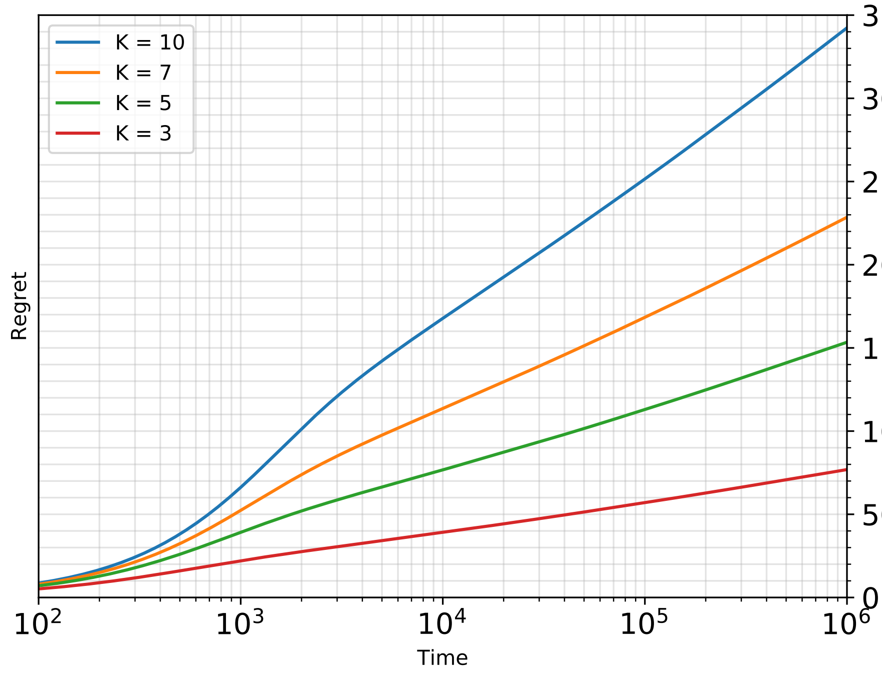


Figure 2: Regret over time when we change the number of arms K. For each case, the regret over time is averaged over $S = 5,000$ simulations and the difference between the highest mean reward and the remaining mean rewards ϵ is 0.1.

By varying the number of arms K, Thompson Sampling must explore more options to find the best arm. The total regret increases as the number of arms increases because the algorithm explores non-optimal arms more often. Figure 2 shows the regret over time plots where the number of arms K is 3, 5, 7, and 10.

To keep a consistent simulation environment, all parameters, besides the number of arms, are kept constant. 5,000 simulations are used for each experiment to reduce the spread of the final regret values. The difference between the best reward distribution mean and the others, ϵ, is set at 0.1. This value is chosen so that the algorithm is able to converge on a best arm within $T = 10^6$ time slots.

K	Final Regret	Standard Deviation
3	76.88	1.68
5	153.41	1.80
7	228.53	2.15
10	342.36	3.07

Table 1: Final regret (i.e., regret at time slot $T = 10^6$) and standard deviation of different simulations for different numbers of arms K.

From Figure 2, it can be seen that, as K increases, the final regret increases. Table 1 shows how the average final regret and its deviation changes as K increases. As expected, the deviation in the final regret does increase with K. This is primarily due to the algorithm dedicating more time to exploration before confidence is high enough to choose a best arm a large portion of the time.

2.3 Changing Epsilon

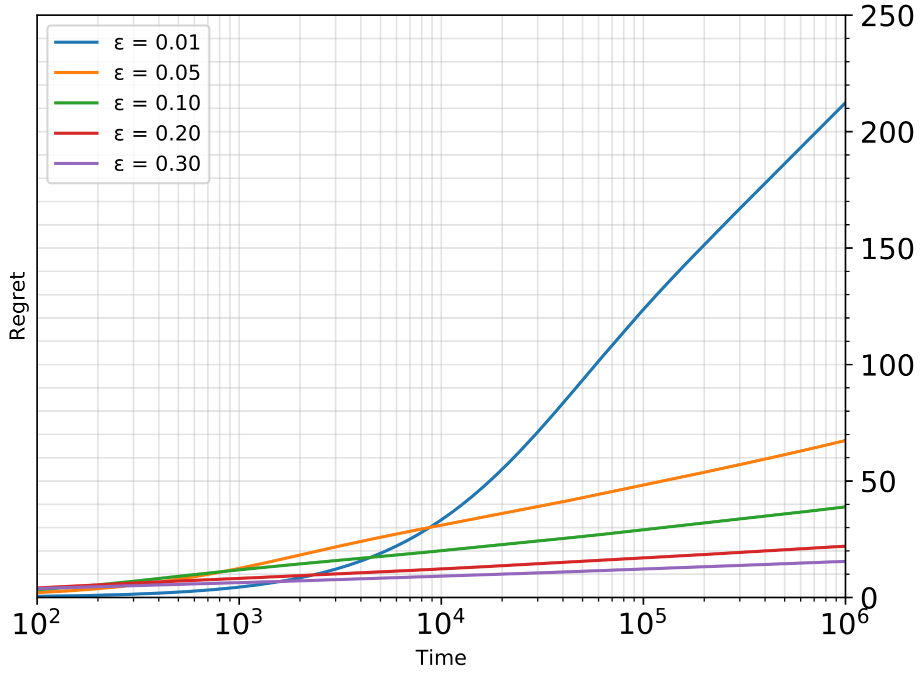


Figure 3: Regret over time when we change ϵ, the difference between the highest mean reward and the remaining mean reward. For each case, the number of arms K is 2 and the regret is averaged over $S = 100,000$ simulations.

The value ϵ describes the difference between the mean of the best arm μ^* and the means of the remaining arms. By decreasing μ^*, the best arm becomes harder to detect and it requires more samples to determine the true mean of each arm. To experimentally show this, $S = 100,000$ simulations are performed for each ϵ value. The number of arms is chosen as $K = 2$, reducing the final regret value and spread due to exploration costs. Figure 3 shows the difference in regret over time with varying values of ϵ. These plots show that, when ϵ is smaller, the final regret value and spread are increased significantly. This is due to the algorithm needing to explore the bad option more to increase the confidence of the estimated reward distribution mean.

From [3], it is known that regret must become logarithmic after some time. Figure 3 shows this bound. The plots of the results with different values of ϵ show that the time it takes to reach this logarithmic behavior is inversely proportional to ϵ. The slope of the logarithmic result is also inversely proportional to ϵ as it significantly increases with lower values of ϵ.

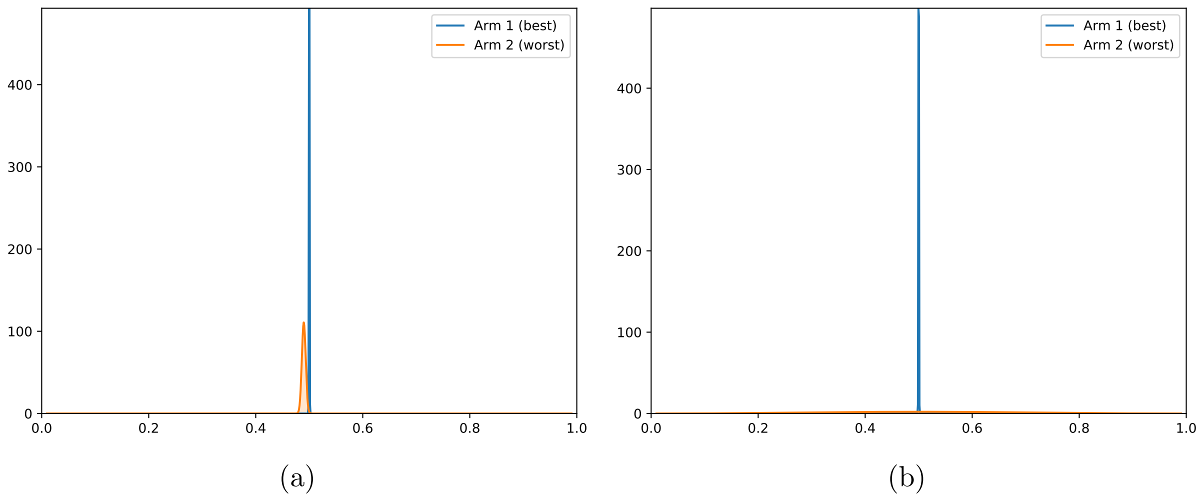


Figure 4: Posterior beta distribution at $T = 10^6$ when ϵ is (a) 0.01 and (b) 0.4.

Arm	ϵ	α_1	β_2	Estimated Mean	Actual Mean
1	0.01	499,988.92	499,983.47	0.500	0.50
2		3.95	27.65	0.125	0.10
1	0.40	489,378.70	489,380.95	0.500	0.50
2		10,409.88	10,834.47	0.490	0.49

Table 2: Final α and β values for both arms when $\epsilon = 0.01$ and $\epsilon = 0.40$.

Figure 4 shows the posterior beta distribution probability density function for the two extreme cases of ϵ. Figure 4a shows the posterior beta distributions for both arms after 100,000 simulations with $\epsilon = 0.01$. Arm 2, the bad arm with $\mu = 0.49$, is explored very often, as given by the narrow distribution, leading to a confident estimate of the reward distribution mean. Figure 4b shows a very wide beta distribution for arm 2, the bad arm with $\mu = 0.1$, indicating that it was explored very little. The average final α and β values for these two extreme cases are recorded in Table 2 where α_i and β_i refer to the α and β values of the i-th arm.

3 Rule of Thumb

Along with Thompson Sampling, there are many classic multi-armed bandit algorithms. Two of the most well-known are Upper Confidence Bound (UCB) (see [7,10]) and Δ-Greedy. UCB is a deterministic algorithm that tracks empirical mean rewards for each arm and calculates an upper confidence bound on the mean reward to select an arm at each time slot. Δ-Greedy is a greedy algorithm that selects with probability $1 - \Delta$ to choose the arm with the highest empirical mean reward and selects an arm uniformly at random with probability Δ where Δ is a value over [0, 1].

3.1 Upper Confidence Bound

Algorithm 2 Upper Confidence Bound

```
Input $K$
$\bar{x}_j = 0,\ n_j = 0 \quad \forall\, j \in [K]$
for $t = 1, \ldots, T$ do
    if $t < K$ then
        $j \leftarrow t$
    else
        for $i = 1, \ldots, K$ do
            $\text{bound}_i = \bar{x}_i + \sqrt{2 \ln t / n_i}$
        end for
     $j \leftarrow \text{argmax}_{j \in [K]}\, \text{bound}_j$
    end if
    $n_j \leftarrow n_j + 1$
    Observe reward $r$ from distribution $i$
    $\bar{x}_j \leftarrow \bar{x}_j * (n_j - 1)/n_j + r/n_j$
end for
```

Algorithm 2 shows the execution of UCB. For the first $t \in [K]$ time slots, the algorithm selects arm t. This allows the mean reward x_j for arm j to be instantiated with some value so that the reward distribution for each arm can be estimated. For the remaining time slots, UCB calculates an expected upper bound for each arm's reward distribution. This is analogous to Thompson Sampling drawing from a posterior beta distribution to estimate the expected reward distribution for each arm. The arm with the highest bound value is then chosen as the action for that time slot.

$$\text{bound}_j = \bar{x}_j + \sqrt{\frac{2 \ln t}{n_j}} \tag{5}$$

To calculate the bound, (5) from [7] is used where $\bar{x}_j$ is the empirical mean reward obtained from arm j, t is the current time slot (the same as the number of arms played in total), and n_j is the number of times arm j has been played so far. The bound is then calculated for all K arms. The arm with the highest bound is chosen to be played in that round.

3.2 Δ-Greedy

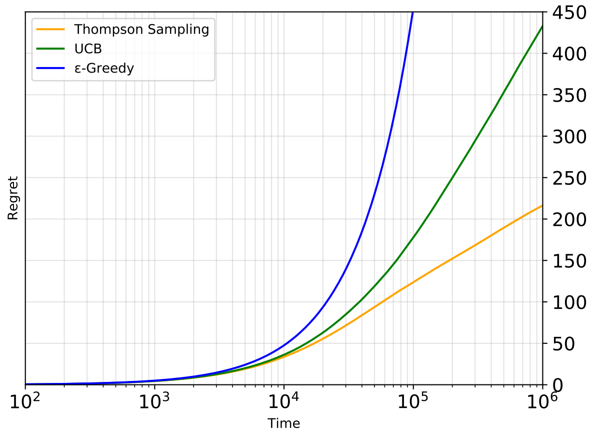
Algorithm 3 Δ-Greedy

```
Input  E, $\Delta$, $K$
$x_i = 0,\ n_i = 0 \quad \forall\ i \in [K]$
for $t = 1, \dots, T$ do
    if $t < E$ then
        Choose arm $i$ uniformly at random
    else
        $u \leftarrow$ Random value uniformly over [0, 1]
        if $u > \Delta$ then
            Choose arm $i$ uniformly at random
        else
            $i \leftarrow \text{argmax}_{i \in [K]}\, x_i$
        end if
    end if
    $n_i \leftarrow n_i + 1$
    Observe reward $r$ from distribution $j$
    $x_i \leftarrow x_i * (n_i - 1)/n_i + r/n_i$
end for
```

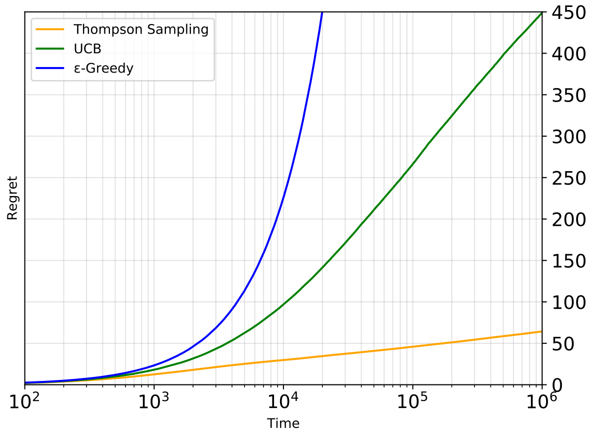
Algorithm 3 shows how the Δ-Greedy algorithm is executed. Δ-Greedy is the naive method of solving the multi-armed bandit problem. The first parameter in Δ-Greedy is the exploration length. This value determines the number of rounds the algorithm is purely using exploration. During these rounds, the chosen arm is selected uniformly at random and explored. After the reward is obtained at the end of each round, the empirical mean reward for the selected arm is updated.

After the exploration rounds, the second parameter is used to determine whether the algorithm will explore the arms, or exploit the best known arm. This parameter Δ is a value over [0, 1] and represents the exploration probability. That is, with probability Δ, the algorithm will randomly select an arm for that round. Otherwise, with probability $1 - \Delta$, the algorithm will exploit the arm with the highest empirical mean reward.

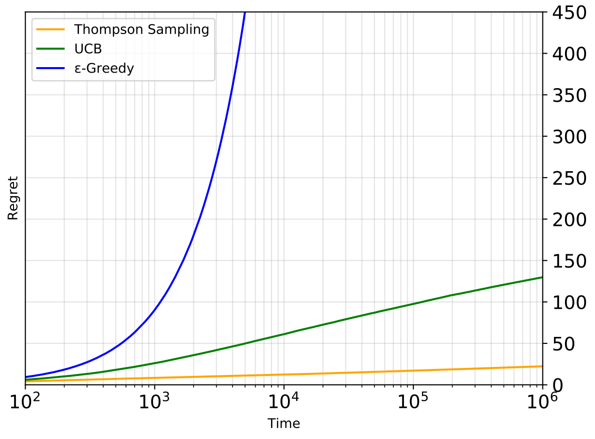
3.3 Results



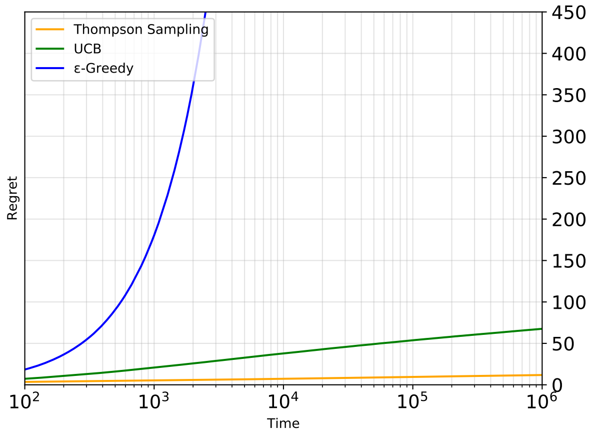
(a) $\epsilon = 0.01$



(b) $\epsilon = 0.05$



(c) $\epsilon = 0.20$

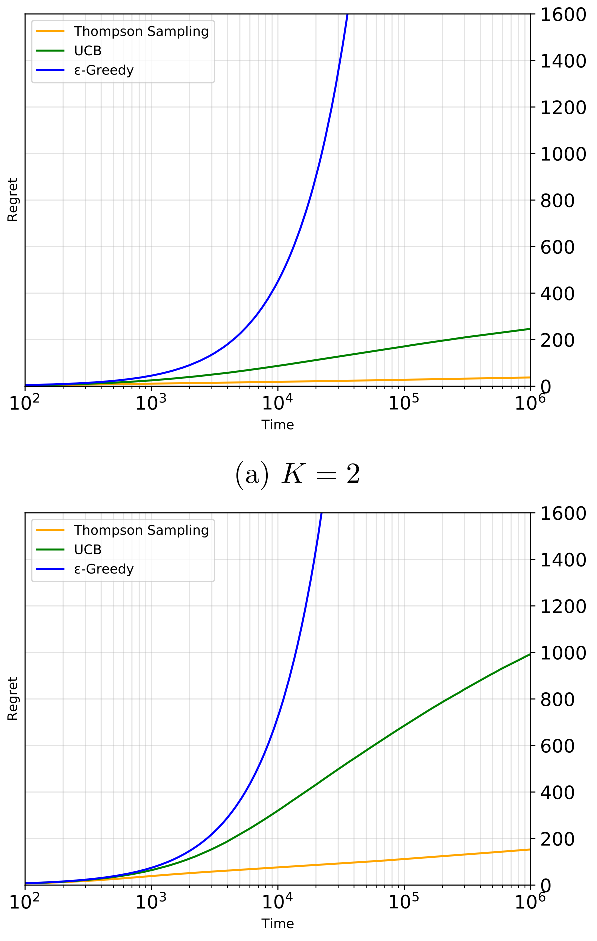


(d) $\epsilon = 0.40$

Figure 5: Thompson Sampling, UCB, and Δ-Greedy when ϵ is (a) 0.01, (b) 0.05, (c) 0.20, and (d) 0.40. For each execution, the number of arms K is 2 and the exploration parameter of Δ-Greedy is 10,000.

When solving a real-world multi-armed bandit problem, we cannot choose the parameters of the system. It is then important to determine which algorithm best suits each situation. To do this, the three algorithms (Thompson Sampling, Upper Confidence Bound, and Δ-Greedy) are examined with varying values of ϵ and K. The exploration parameter of Δ-Greedy is selected to be 10,000 so that, for the first 10,000 time slots, the arm selected is chosen uniformly at random to allow for the algorithm to attain a reasonable understanding of the reward distributions. Figure 5 shows the outcome of all three algorithms when ϵ is 0.01, 0.05, 0.20, and 0.40. The Δ-Greedy plot increases beyond what the plot can show as it increases exponentially and would force the UCB and Thompson plots to be barely visible if shown in its entirety.

Figure 5 uses $K = 2$ arms and a best arm mean $\mu^* = 0.5$ in order to accurately compare the performance of each algorithm with different ϵ values. As the value of ϵ increases, the range within which the three algorithms perform similarly decreases. In Figure 5a where $\epsilon = 0.01$, this range extends to $t = 10^3$ while in Figure 5d where $\epsilon = 0.40$, this range is less than 10^2. In fact, the algorithms begin to diverge around $t = 50$.



(a) $K = 2$

(b) $K = 3$

(c) $K = 5$

(d) $K = 7$

Figure 6: Thompson Sampling, Upper Confidence Bound, and Δ-Greedy when K is (a) 2, (b) 3, (c) 5, and (d) 7. The difference between the highest mean reward and the remaining mean rewards ϵ is 0.1 and the exploration parameter for Δ-Greedy is 10,000.

The first algorithm to diverge is Δ-Greedy. Thus, it should only be used when the number of rounds is sufficiently small with respect to the value of ϵ. UCB and Thompson Sampling stay close together for a short amount of time afterwards. However, UCB does eventually begin to increase faster than Thompson, making it a worse choice than Thompson Sampling. When we compare the response to a changing ϵ value, it is clear from Figure 5 that Thompson is generally the best choice while Δ-Greedy and UCB can be chosen if the number of rounds is small enough. Figure 6 then shows the response of each of the three algorithms when the number of arms K is 2, 3, 5, and 7. As with Figure 5, the y-axis is limited so that showing the entirety of Δ-Greedy does not hide UCB and Thompson Sampling.

The three algorithms have a similar performance for a period of time at the beginning of the experiment, before diverging and increasing at different rates. Following the same reasoning, Δ-Greedy is only a viable choice for smaller values of T. However, there is not a very large correlation between increasing K and the number of rounds the three algorithms perform similarly. In all four plots, the divergence begins between round 200 and 400 and does not change as K increases.

After evaluating both cases of testing the effect of changing ϵ and K, a rule of thumb can be created to determine which algorithm is best to choose in different scenarios. When the number of rounds is small, all three perform similarly. If computation costs are an issue, Δ-Greedy is likely the best choice as the calculations simply include running empirical means and a uniform random choice with probability of Δ. UCB is the next most computationally heavy as it requires running empirical means along with a slightly more difficult calculation for the bound on each arm. Thompson Sampling is then the heaviest computationally as it requires the use of a prior beta distribution from which samples are taken at each round to determine the arm to select.

As the number of rounds increases, Thompson Sampling emerges as the clear best choice for an algorithm. Δ-Greedy begins to increase very rapidly over time and is clearly not a good choice for situations with a high number of rounds. Both UCB and Thompson are bounded logarithmically, however Thompson Sampling has a lower scale and is thus the better choice for higher values of T.

3.4 Summary

After we compare Upper Confidence Bound, Δ-Greedy, and Thompson Sampling with varying simulation parameters, Thompson Sampling emerges as the best algorithm for simulations with a higher number of time slots. Thompson Sampling is also the best choice for simulations with a high degree of difficulty in selecting a best arm. That is, when the value of ϵ is low or the number of arms K is high.

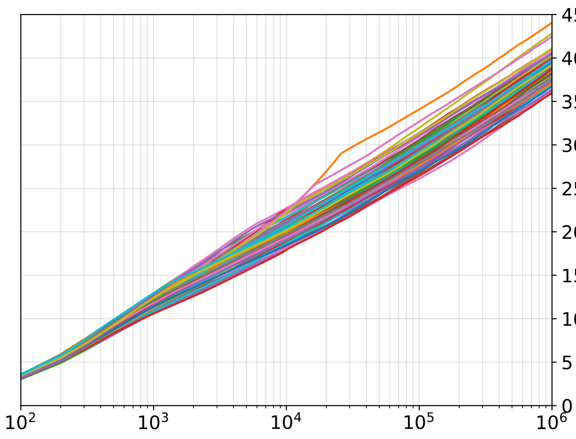
For computationally constrained systems, Δ-Greedy is the best choice due to the simplicity of its calculations followed by Upper Confidence Bound. Thompson Sampling simulations are very computationally intensive due to the overhead of sampling from a beta distribution. For small values of T, all three algorithms perform similarly. The range in which they perform similarly increases as the value of ϵ decreases.

4 Proposed Modifications

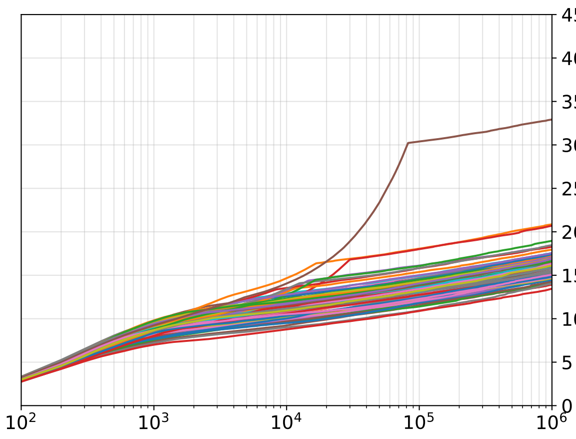
At each time slot in standard Thompson Sampling, a sample θ_i from each arm i's posterior distribution is taken. The arm with the highest sample is then chosen as the arm to be selected for that time slot. A potential modification of this algorithm is to sample multiple times from each arm to form a more confident understanding of the current posterior distribution for each arm. Then, the arm with the highest average sample is chosen to be sampled at that time slot. This proposed improvement can be implemented in two ways. First, a static sampling can be chosen where, at each time slot, every arm is sampled A times. Second, the number of samples A is progressively tuned based on the current time slot t. For example, for the first 10^3 time slots, the normal one sample, $A = 1$, can be taken, and for the remaining time slots, two samples, $A = 2$, can be taken. This can be progressively tuned to increase the number of samples further as the number of time slots increases.

4.1 Static Change For All Time Slots

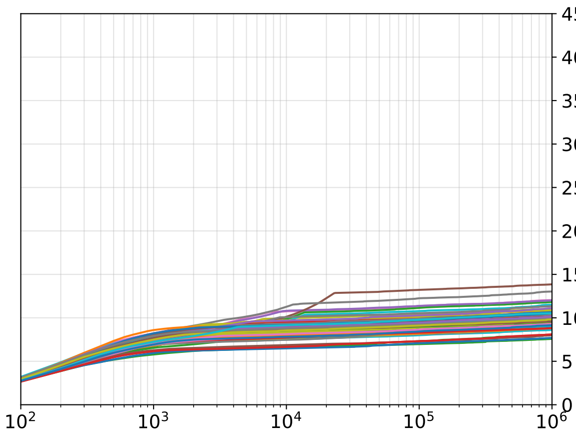
When we are implement a static change over all time slots, A is given as a constant value. The purpose behind this adaptation is to achieve a more accurate estimate of the beta distribution. Simply taking one sample gives the potential for a good arm to repeatedly return a poor result while a bad arm will return a good result. This problem comes at even greater probability when the time slot is small (i.e., the beta distribution has not changed significantly from the uniform distribution) and when the difference of mean reward is small. Figure 7 shows the improved results when using 1, 3, 7, and 10 samples with $K = 10$ arms and $\epsilon = 0.1$. For each scenario, 200 runs of 500 simulations are performed so that the effects of any potential outliers are reduced.



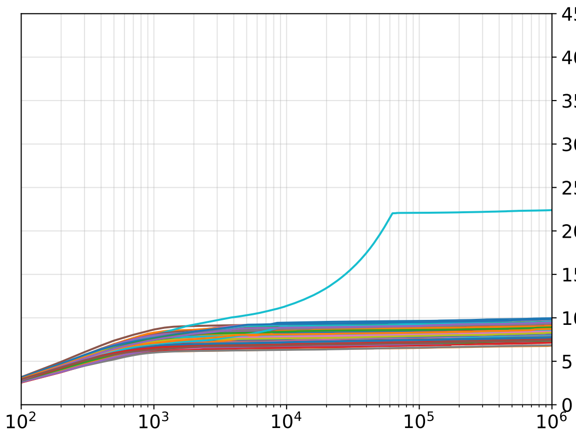
(a) $A = 1$.



(b) $A = 3$.



(c) $A = 7$.



(d) $A = 10$.

Figure 7: Regular Thompson Sampling (a) with (b) 54.5%, (c) 50.5%, and (d) 43.0% of results being improved over (a) using multiple samples with $K = 2$ arms, the difference between the highest mean reward and the remaining mean reward ϵ is 0.1, and each experiment plotted is averaged over 500 simulations of the algorithm. The remaining experiments that are not improved are not shown due to the large difference in cumulative regret between improved and worsened cases.

For reference, Figure 7a is included to show the result of a regular Thompson Sampling performance. The average final regret for this case is 38.734 with a standard deviation of 1.102. To judge final regret improvement over the reference, the percentage of experiments with a final regret below 45 is calculated as this shows the amount of experiments that did not get worse. Additionally, the mean and standard deviation for each experiment is calculated to show consistent improvements over the reference.

An interesting result of this proposed improvement is that a portion of the experiments, instead of following the theoretical logarithmic bound for Thompson Sampling, instead increase linearly. This results in two groups of results where one is better than the reference and one is significantly worse. As the value of A increases from 1 to 10 when changed statically, the percentage of improved cases over 200 experiments moves from over 50% down to 43%.

This behavior is most likely explained by the algorithm accidentally choosing the bad arm over the good arm early on in the execution, leading to a poorly informed result by the end of the experiment. Viewing the α and β values after execution, however, shows that the algorithm does in fact exploit the best arm more than it explores the bad arm, though it does explore the bad arm significantly more than in the classic algorithm.

A	Improved Mean	Improved Standard Deviation	Worsened Mean	Worsened Standard Deviation	Improvement (%)
1	38.734	1.102	–	–	–
3	16.282	1.990	276.224	172.121	54.5
7	9.806	0.977	2783.26	688.579	50.5
10	8.638	1.678	5073.133	996.306	43.0

Table 3: Thompson Sampling performance for improved and worsened cases when $A \in \{1, 3, 7, 10\}$.

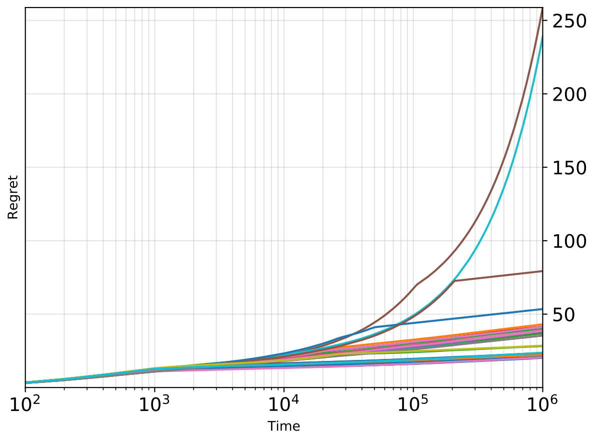
Table 3 shows the average final regret after $T = 10^6$ trials over 200 runs with $S = 500$ simulations each for $K = 2$ arms with a difference of means of $\epsilon = 0.1$ (i.e., $\mu_1 = 0.50$ and $\mu_2 = 0.40$). The number of samples A used ranges from 1, the reference figure, to 10. The table includes the mean and standard deviation for improved and worsened cases as well as the percentage of improved cases. Smaller increases of A above 1 showed a significant improvement in final regret, while higher values of A showed less significant improvements when considering the improved cases. When the outcome was not improved, it was generally very bad with final regret values averaging over 5,000 with $A = 10$ samples. Additionally, the outcome was not consistent with a standard deviation of 996.

The results of this experiment show that there is the possibility of improvement using a multi-sample technique. However, it is almost equally as likely to produce a significant deterioration in final regret. Using a sample value above 3 will likely result in frequently poor results, while using a sample value of 2 or 3 will show some improvement with the possibility of degradation. The following section covers an improvement over a constant A value in which the value is dynamically adjusted.

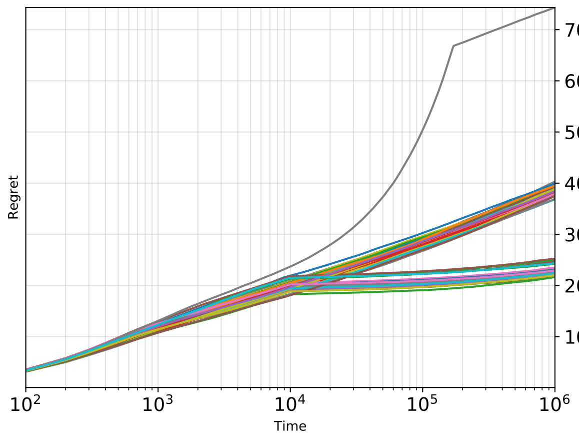
4.2 Time-Varying Change

From the previous section, it is seen that a constant change in the number of samples A can result in the algorithm accidentally choosing the bad arm early on, leading to a potential increase in final regret over the classic algorithm. To remedy this, A can be chosen to increment gradually over time.

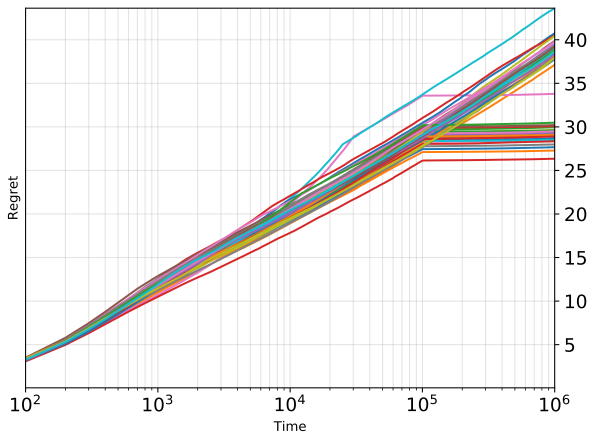
Figure 8 shows the results of changing A from 1 at the beginning to 2 at 10^3, 10^4, and 10^5. As with the previous test, these figures use $K = 2$ arms with a difference of means of $\epsilon = 0.1$ for 50 runs with $S = 500$ samples each. Since adding additional figures would clutter the paper, the repository at [11] contains all simulated data referenced in this paper in addition to the data omitted in order to keep the figures concise.



(a)



(b)



(c)

Figure 8: Gradual increment of A from 1 to 2 at $t =$ (a) 10^3, (b) 10^4, and (c) 10^5. The number of arms K is 2, and the difference between the highest mean reward and the remaining mean reward ϵ is 0.1. Each experiment plotted is averaged over 500 simulations.

Figure 8a shows the results of incrementing A at $t = 10^3$. While the majority of the resulting runs are equal to or better than the classic algorithm, there are four noticeable runs that are much worse. Additionally, two groups can be seen to split into final regret values of around 40 and around 25. These groups become even more distinct in Figure 8b in which A is incremented at $t = 10^4$. In this, only one plot goes above the classic reference final regret of 45. This shows that by simply waiting until $t = 10^4$ to increase the number of samples, a significant increase can be observed nearly half the time with a much smaller chance of error than with a static change in A.

To further reduce the chance of error, A can be increased at $t = 10^5$ as in Figure 8c. In this plot, the improved cases are slightly higher than previous plots, however there is no longer any divergent runs with a final regret above 45. This plot shows that by waiting even further until $t = 10^5$, an improvement can be observed half of the time with a minimal chance for error.

It is important to note, however, that these improvements are observed with specific algorithm parameters ($K = 2$, $\epsilon = 0.1$). Therefore, it is necessary to further adjust the time slot where A is incremented in the event these parameters are different. If K increases or ϵ increases, then the time slot will likely need to be increased to accommodate for the increase in difficulty the algorithm faces in determining a best arm.

References

[1] L. P. Kaelbling, M. L. Littman, A. W. Moore. "Reinforcement learning: a survey," In *Journal of Artificial Intelligence Research*, May 1996.[Online].Available: https://www.jair.org/index.php/jair/article/download/10166/24110/

[2] W. R. Thompson. "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples," In *Biometrika*, vol. 25, no. 3/4, Dec., pp. 285-294, 1933.

[3] S. Agrawal and N. Goyal, "Analysis of thompson sampling for the multi-armed bandit problem," In Proc. *JMLR 25th Annual Conference on Learning Theory*, 2012, pp. 39.1-39.26.

[4] D. J. Russo, B. V. Roy, A. Kazerouni, I. Osband, and Z. Wen, "A tutorial on thompson sampling", Nov 2017, e-print Arxiv: 1707.02038.

[5] T. Lattimore and C. Szepesvari, *Bandit algorithms*, Dec 2019, Available: https://tor-lattimore.com/downloads/book/book.pdf.

[6] O. Chapelle and L. Li. "An empirical evaluation of thompson sampling," [Online].Available: https://papers.nips.cc/paper/4321-an-empirical-evaluation-of-thompson-sampling.pdf.

[7] P. Auer, N. Cesa-Bianchi, and P. Fischer. "Finite-time analysis of the multiarmed bandit problem," In *Machine Learning*, vol. 47, pp. 235-256, 2002.

[8] S. Agrawal and N. Goyal, "Near-optimal regret bounds for thompson sampling," In *J. ACM*, vol. 64, no. 5, Sept., article 30, 2017.

[9] A. Slivkins. "Introduction to multi-armed bandits," Sep. 2019, e-print Arxiv: 1904.07272.

[10] T. L. Lai and H. Robbins. "Asymptotically efficient adaptive allocation rules," In *Advances in applied mathematics*, 6:4-22, 1985.

[11] [Online].Available: https://github.com/NickChiapputo/MultiArmedBandit