Analysis of Regret Function Trends of Algorithms for Multi-armed Bandit Problem

Bingjiang Xia*

Department of Mathematics and Statistics, Boston University, Boston 02215, United States of America

*Corresponding author: bx19@bu.com

Keywords: multi-armed bandit problem; Thompson sampling; regret function.

Abstract: The multi-armed bandit (MAB) problem has gained interest in reinforcement learning and other areas due to its wide applications. Many algorithms for the MAB problem have been proposed. This paper discusses the three most common algorithms for the MAB problem: ε-greedy algorithm, upper confidence bound algorithm, and Thompson sampling algorithm. Since the performance, in terms of expected returns, of these algorithms is critical when deciding which algorithm to use, this paper analyzes the trends and properties of the regret function of these algorithms to compare their performance by stimulations. It is suggested that the Thompson sampling algorithm usually shows the best algorithm among the three algorithms based on the simulation performed.

1. Introduction

The multi-armed bandit (MAB) problem [1] describes a gambler facing a row of slot machines with multiple arms, each with a different chance of winning. The gambler needs to decide how to get the most rewards by choosing other machines within a limited number of times. When there are just a few slot machines, the gambler can simply play each slot machine many times and find the machine with a relatively higher probability of winning and then play that machine in future games to win more rewards. However, when there are many slot machines or when the gambler has a small number of games, this method can take expensive costs and usually gives poor total rewards. It is also notable that, in each round of play, the probability of winning is sampled from the probability distribution of the individual arms, which is independent of the gambler's previous actions. An important factor affecting the MAB problem's success is finding a good trade-off between exploration and exploitation [2]. Due to the nature of the MAB problem, it has various applications in statistics [3], reinforcement learning [3], operation research [4], biological research [5], and business management [6], so it has attracted increasing interest in research in many areas.

As mentioned above, the standard type of MAB problem is when the gambler faces a number of slot machines in which each machine has a different but fixed chance of winning. Another notable MAB problem is the multi-armed bandit problem with a known trend [7]. The multi-armed bandit problem with a known trend is when the gambler faces several slot machines, each following a known trend reward function. The gambler must decide which machine to play in this changing environment. This type of MAB problem adds complexity to the algorithm design and analysis. In addition, the MAB problem with known trend has various applications in the real world. In recent research [7], scientists are motivated by applications in music recommendation and active learning to propose a modified algorithm to be used in the MAB problem with known trends. Another MAB problem critical in terms of application is the MAB problem with covariates [8]. Compared with the standard static MAB problem, the multi-armed bandit problem with covariates is in a dynamic setting in which each machine has a noisy reward realization depending on an observed random covariate. A multi-armed bandit problem with covariates more appropriately describes the situation with dynamically changing rewards and thus is more suitable for applications where side information can be involved.

MAB problems have many applications in real life. One notable research proposes a picture-based decision support system called DSSApple to help identify apple diseases [5]. This system is based on

the concept and algorithm of contextual MAB problems. Specifically, DSSApple uses past interactions with the system and its user-provided information and consistently updates the decision model to empower decision-making accuracy. The idea of MAB problems can also be applied to applications in quantitative finance. In recent years, sequential portfolio selection has become a popular investment tool [9]. The primary difficulty in sequential portfolio selection is to achieve a balance between risk and return under uncertainties. This idea is similar to the concept of exploration versus exploitation in the MAB problem. Thus many mathematicians in the field of quantitative finance have started to apply the algorithm of sequential decision making based on concepts of the MAB problem to address the challenge of achieving a balance between risk and return in sequential portfolio selection [10]. Risk awareness and topological structure of the financial market have been considered in the algorithm constructing portfolio, which is a modification of optimal MAB policy. In addition, the application of MAB problems in recommendation system services, such as online news, advertising, and merchandise selection, has gained attention due to its capability of leveraging contextual information. Potential implementations of MAB in robotics state estimation [11] and control [12] are also shown, such as multi-target search [13]. In the theoretical analysis, the application of MAB problems usually involves the assumption of the existence of a reward mapping function. This reward mapping function is fixed but, in practice, is typically unknown. In fact, in real-life applications, this reward mapping function has to deal with those variables that are dynamically evolving over time. Thus, research in reward mapping functions modeled as a set of random walk particles gained popularity in recent years. Those random particles are used to learn the mapping process dynamically, so this method is able to effectively capture the context change in real life and make a recommendation accordingly.

Due to the popularity of MAB in practical applications, there are many researchers proposed solutions and algorithms [14] to MAB problems. Upper confidence bound (UCB) algorithm, ε-greedy algorithm, and Thompson sampling (TS) will be discussed in this paper. One classic algorithm to address MAB problems is the ε-greedy algorithm. As mentioned above, the MAB problem is a problem about exploration and exploitation. The ε-greedy algorithm [15] is a method that chooses between exploration and exploitation randomly in order to achieve a balance between exploration and exploitation. ε is the probability of exploration, and usually, it is small. Therefore, through the process of exploration and exploitation, the ε-greedy algorithm selects the highest reward most of the time, but still, there is the flexibility of exploring. While the ε-greedy algorithm selects an action with a probability of exploration, another algorithm that addresses the MAB problem, the Upper Confidence Bound [16], changes the balance between exploration and exploitation through the learning process. The first stage of the UCB algorithm primarily focuses on exploration. When the algorithm gradually gathers the context and setting of the problem, it will change its strategy to exploitation. In the exploitation process, the algorithm will select the action with the highest reward based on the learning from the environment. TS algorithm [17] is also a solution to the MAB problem. TS algorithm is a sampling method that samples each machine's probability distribution to produce the highest reward. The basic idea of the TS algorithm is that in each round, it samples the parameters from the posterior distribution based on the posterior belief about the unknown parameters [8]. This sampling parameter produces a set of expected returns for each machine, and the machine with the highest expected return has been determined. Specifically, the TS algorithm considers Beta-Bernoulli Bandit such that the prior distribution is a beta distribution, the distribution of each machine reward is a Bernoulli distribution [18] with parameters, and the posterior distribution also follows a beta distribution [19]. In practice, the TS algorithm usually gives a lower total regret [20] than the ε-greedy algorithm and UCB algorithm [21]. TS algorithm also has an extensive range of applications, such as web page recommendation problems [22] or even chemical molecular screening problems [23]. Therefore, the study of the TS algorithm has always been a valuable topic in reinforcement learning.

Since the performance of MAB algorithms is crucial when deciding which algorithm to use, this paper will analyze the regret function of the ε-greedy algorithm, UCB algorithm, and TS algorithm. This paper will first introduce the mathematical background of the MAB problem and the algorithms in general. Further, this paper will analyze the regret function of the algorithms in many aspects: the formulas, properties, and comparisons among the three algorithms. In addition, this paper will run

stimulation to further analyzes the trends of regret function of the three algorithms in a general MAB problem setting.

2. Solutions to MAB Problems

2.1 MAB Formulation

Suppose the gambler is given N slot machines. The gambler must play a slot machine each round. Each machine has a probability that gives a random reward; when the gambler plays any slot machine n, the machine will give a random reward based on some unknown distribution in [0, 1]. As discussed above, each play is independent, so the random rewards are independently identically distributed. Suppose there are i = 1, 2, 3, … round. The challenge for the gambler is to decide which slot machine to play in ith round based on the rewards received in all previous rounds. Therefore, the goal for the gambler is to maximize the expected total reward in a fixed number of rounds. An equivalent measure to analyze is the expected total regret [20]. The regret is the reward that the gambler loses due to not playing the optimal machine. For the total I round, the regret function, *R(I)*, can be defined as:

$$R(I) = \sum_{i=1}^{I}(\mu_{maxi} - \mu_i), \tag{1}$$

Where μ_{maxi} is the optimal expected reward in i^{th} play, and μ_i is the actual expected reward in i^{th} play. Then the goal for the gambler becomes minimizing the expected total regret, that is, to minimize:

$$\mathrm{E}(R(I)) = \mathrm{E}(\sum_{i=1}^{I}(\mu_{maxi} - \mu_i)). \tag{2}$$

2.2 ε-greedy Algorithm

The ε-greedy algorithm is a method that sets an ε value between 0 and 1 that decides whether the gambler is to explore (try a new slot machine) or to exploit (play a machine that is previously played). For example, ε is set to a, then 100α% of the rounds are devoted to exploration, and 100(1-α)% of the rounds are used to exploit. Each gambler plays, the gambler should draw a random number from 0 to 1. If the number is greater than ε, then the gambler should play the machine with the highest probability of winning (estimated probability of winning) based on previous rewards. If it is less than ε, select another machine at random, then update the estimated winning probability of this machine at the same time for future reference. The problem with the ε-greedy algorithm is that the estimated probability of winning for a slot machine varies with the number of times the drawbar is played. If a machine is not played at all, the estimated probability of winning for that machine does not change. Therefore, the accuracy of the estimated probability of a machine is based on how many times the gambler played that machine. Based on this property, the ε-greedy algorithm will still choose a non-optimal machine due to the chance of exploitation. Therefore, the number of plays of the optimal machine is not maximized. Hence the expected total reward is usually not maximized neither.

2.3 Upper Confidence Bound Algorithm

While the ε-greedy algorithm uses the estimated probability of winning as a criterion to play a machine, called blind choice, the UCB algorithm chooses a machine to play based on its potential probability of winning. The UCB algorithm dynamically changes its balance of exploration and exploitation as more round is played; that is, more knowledge of the game context is learned through the process of playing. Using this idea, the machine that should be played during i^{th} round, M_i, can be modeled as [21]:

$$M_i = argmax\left[Q_i(m) + c\sqrt{\frac{\log i}{N_{i-1}(m)}}\right], \tag{3}$$

Where $Q_i(m)$ denotes the the machine with highest estimated reward m, $N_{i-1}(m)$ denotes the number of plays of machine m during the 1st play to the (i-1)th play, and c is the confidence level. The first part (term) of equation (3), $Q_i(m)$ presents "exploitation". The UCB algorithm will choose a

machine with highest estimated reward during previous plays if machine with highest reward is unsure. The second part of equation (3) represents "exploration". The confidence level c controls the level of exploration. Based on this equation, if the number of plays of a certain machine is relatively small, then this machine is more likely to be selected. The reason is that if the machine is not relatively frequently selected, the $N_{i-1}(m)$, the number of plays that the machine is selected during previous rounds, is small. The smaller $N_{i-1}(m)$ will make a larger value of the fraction. Therefore, the uncertainty is large so this machine is more likely to be selected. As more round is played, the selection of machine is more based on the exploitation since the exploration term will gradually decreases and converges to 0 as the number of rounds approaches to infinity.

2.4 Thompson Sampling Algorithm

While the ε-greedy and UCB algorithm selects the machine based on the reward during previous rounds, the TS algorithm builds a probabilistic model from the reward obtained that improves the average reward. Therefore, the estimated reward is more accurate as more round is played. TS algorithm samples from the probabilistic model in order to make a decision. The simplest context is when the TS algorithm deal with Bernoulli distribution. Assume that there are two outcomes of a machine being played:

$$\begin{cases} 1, if\ the\ machine\ has\ reward, \\ 0, if\ the\ machine\ doesn't\ has\ reward. \end{cases} \tag{4}$$

This means the reward only has two values: 1 if the machine has a reward or 0 if the machine has no reward. Based on this Bernoulli model, the TS algorithm is able to find the machine with the highest probability of giving a reward instead of finding the machine with the highest reward. However, in practice, the context of problems is not as obvious as Bernoulli distribution. Therefore, the TS algorithm needs to consider the probabilistic model of the reward for each machine using Beta distribution since the reward is binary. Beta distribution is ideal for the TS algorithm:

$$f_X(x) = \frac{1}{B(\alpha,\beta)} x^{\alpha-1}(1-x)^{\beta-1}, 0 < x < 1, \tag{5}$$

Where $B(\alpha,\beta)$ is the beta function:

$$B(s,t) = \int_0^1 x^{s-1}(1-x)^{t-1} = \frac{\Gamma(s)\Gamma(t)}{\Gamma(s+t)}, s,t > 0. \tag{6}$$

At the beginning, the gambler has no prior knowledge about the machine, so the initial setting of the two paramters α and β is 1. This gives a flat uniform distribution model. This initial setting is also called prior probability [20]. The prior probability defines the probability of an event occurring before the algorithm gathers knowledge from it, which is the initial setting. Once the algorithm have gathered knowledge about the winning probability of a certain machine, the algorithm is able to update paramters α and β to obtain a new probability. This is called posterior probability [24]. In beta distributions, the parameter α represents the number of "success" (the number of times of the slot machine gives rewards) and the parameter β represents the number of "failures" (the number of times of the slot machine does not give rewards). If a machine gives a reward, then α will increase by 1; if a machine doesn not give a reward, β will increase by 1. Therefore, as number of plays increases, the beta distribution model will become more accurate to estimate the expected total reward. Thus the TS algorithm is able to describe the estimated expected total reward based on the prior probability model and posterior probability model by taking samples from the beta distribution of each machine and choosing the machine with the highest reward among discovered machines and locating the near-optimal slot machine.

3. Regret Analysis of MAB Algorithm

The performance of a MAB algorithm can be evaluated by the total expected reward. The higher total expected reward means the algorithm is closer to the optimal solution. As discussed above, an

equivalent measure of the total expected reward is the total expected regret. As the term regret suggests, when the gambler makes a choice about which machine to play, there is no way to know in advance whether the specific machine gives the most reward or not. However, the total expected regret can be used to compare the selection policy given by the algorithm against the optimal situation. To assess the performance of a multi-armed algorithm in a practical setting, if the regret function is constant or increases slowly, it can be concluded that the algorithm has relatively optimal performance in terms of maximizing total reward.

To maximize the reward, the algorithm needs to minimize the total expected regret, as noted in equation (2). The arm gives the most reward is denoted as

$$a_* = \max \mathrm{E}[W_i | A_i = a], \tag{7}$$

Where W_t is the reward given at round *i* and the arm a_* maximizes the the expected reward at step *i*. As noted above, the TS algorithm involves posterior probability distribution. The posterior probability distribution of machine that the gambler plays at round *i* can be defined as

$$f(a_i) = P(a_* = a_i | H_i). \tag{8}$$

This is a conditional distribution where the machine played at round *i* is conditioning on H_i, the historical information distribution [22] on rewards from round 1 to $i-1$. For each round the gambler plays a machine, the TS algorithm will sample from this distribution. Based on the context of MAB problem, the machines are independent from each other. It is further notable that the optimal machine a_* and the machine that the gambler played a_i are independently identically distributed since each machine is indepdent from each based on the context of MAB problem.

Since the posterior probability distribution of a machine follows a conditional distribution. Specifically, the probability that a machine gives a reward (assume the machine is in Bernoulli setting) depends on the historical information collected during previous rounds, on the probability that machines played before round *i.* If the number of machines is small, the algorithm is able to quickly identify the machine with the highest probability that gives a reward and continues to choose that machine. Therefore, the regret is nearly zero. However, if the number of machines is fairly large or if the gambler has a limited number of rounds, the TS algorithm faces the trade-off between choosing the optimal machine among the historical information and continue exploring more untested machines. Therefore, the regret varies each round and continues to increase until the algorithm finds the optimal machine. Once the algorithm finds the optimal machine, the gambler will choose this machine for the rest rounds. Therefore, the regret becomes zero as the the optimal expected reward μ_{maxi} is equivalent to the actual expected reward μ_{maxi}. Secifically, the regret function of TS algorithm follows logarithmic regret [17]. The logarithmic regret function starts to increase as the algorithm has no prior or little information about the probability of machines giving a reward. Once enough historical information about the probability distributions is collected, the regret function becomes more stable. The regret function starts to swing, either increasing or decreasing in a small range, depending on the difference between optimal expected reward and actual expected reward. During this period, the regret function will reach its maximum. After that, the regret function starts to decrease since, as more machines are explored, the algorithm chooses the most optimal machine among a big portion of the machines. This results in the actual reward increasing, so the regret decreases. Once the algorithm explores all the slot machines and finds the machine that gives the most rewards, the difference between optimal expected reward and actual expected reward becomes zero; the regret function falls to zero. This is consistent with the property of the logarithmic regret function which the values of regret falls to zero as more rounds are played. However, this property assumes that the gambler has enough rounds to explore all the machines. In real-world applications, the number of trials is usually limited. The next section will show the performance of the TS algorithm with a different number of trials to play, including the situation in which there are not enough trials to explore all the machines. If the number of machines is small, the algorithm is able to quickly identify the machine with the highest probability that gives a reward and continues to choose that machine. Therefore, the regret is nearly zero. However, if the number of machines is fairly large or if the gambler has a limited number

of rounds, the TS algorithm faces the trade-off between choosing the optimal machine among the historical information and continue exploring more untested machines. Therefore, the regret varies each round and continues to increase until the algorithm finds the optimal machine. Once the algorithm finds the optimal machine, the gambler will choose this machine for the rest rounds.

Although all three algorithms solve the MAB problem discussed above share one thing in common: they need to balance exploration and exploitation and use different approaches to achieve this balance. Therefore, regret is different among these three algorithms. In the ε-greedy algorithm, the action that the gambler takes is random. Due to the random exploration-exploitation property of the ε-greedy algorithm and the probability of exploration being constant, the regret will start to accumulate in the first few rounds and increase linearly. Unlike there is a fixed probability of exploration in the ε-greedy algorithm, the UCB algorithm updates its probability of exploration and exploration as it gathers more information about the problem's context. In the early stage of the UCB algorithm, the algorithm has no or little prior knowledge about the probability distributions that the machines give reward, so the probability of exploring other machines is relatively higher. Therefore, the regret will vary each round, but the overall trend increases. As the algorithms get more historical information about the machine's setting and better estimates for the rewards given by each machine are gathered, it will adjust the probability of exploration to a lower level. Thus, the chances of playing an optimal machine among the discovered machine are higher, which results in a higher expected optimal reward. Hence the regret function will gradually decrease. Consequently, the UCB algorithm has a lower level of regret than the ε-greedy algorithm has. Further, it is noticed that although the UCB algorithm and TS algorithm use different approaches to solve a MAB problem, their trends and regret functions are similar. This comparison will be further examined in the next section.

4. Results and Discussions

4.1 Simulation

The performance and trend of the regret functions of the three algorithms for MAB problem have been examined by simulation. Let the N denotes the number of slot machines and T denotes the number of rounds that the gambler played. Suppose the gambler is facing 500 ($N = 500$) slot machines. Each machine gives a random fixed reward from $[0,0.1]$. The rewards are uniformaly distributed between $[0,0.1]$. The goal for the gambler is to collect as much as reward possible in 1000 rounds ($T = 1000$). To achieve this goal, the gambler uses the three algorithms (ε-greedy, UCB, and Tompson sampling algorithm) discussed previously as stradegies. The common ε value [25] in practice is 0.1, so the stimulation also uses 0.1 as the probability that choosing a unvisited slot machine each round. In addition, the simulation is repeated for ε = 0.2 and ε = 0.4 for further analysis. All other settings remain the same as discussed in previous sections for UCB algorithm and TS algorithm. In the stimulations, the regret functions

$$R(t) = r_{maxt} - r_{realt} \tag{9}$$

For the three algorithms are captured, where $R(t)$ denotes the regret function and $t\epsilon T$, r_{maxt} denotes the maximum reward at t[th] round, and r_{realt} denotes the actual reward that the gambler gets at t[th] round. For analyzing the trend of regret function of TS algorithm, its stimulation runs again for $T = 3000$. The performance of the three algorithms can be campared using the regret functions.

4.2 Results

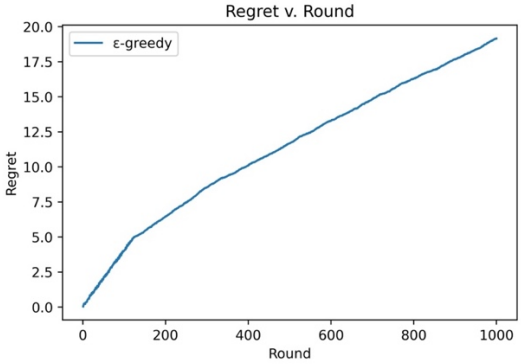

Fig.1 Regret function for ε-greedy algorithm when $\varepsilon = 0.1$, $N = 500$, and $T = 1000$.

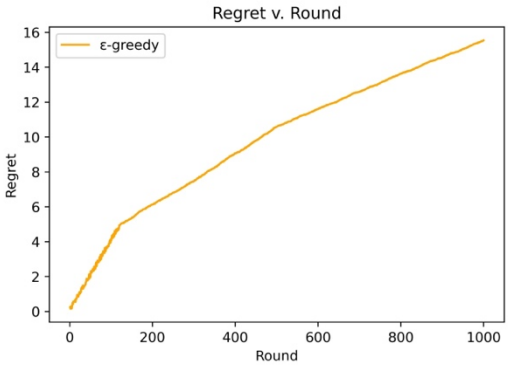

Fig.2 Regret function for ε-greedy algorithm when $\varepsilon = 0.2$, $N = 500$, and $T = 1000$.

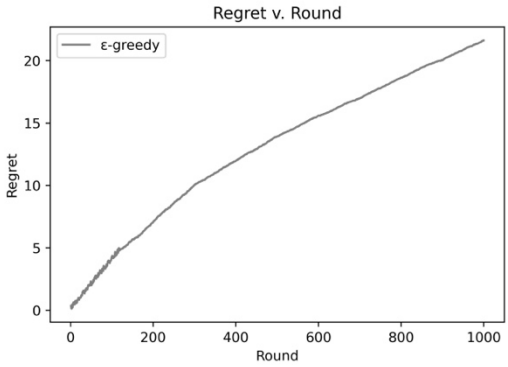

Fig.3 Regret function for ε-greedy algorithm when $\varepsilon = 0.4$, $N = 500$, and $T = 1000$.

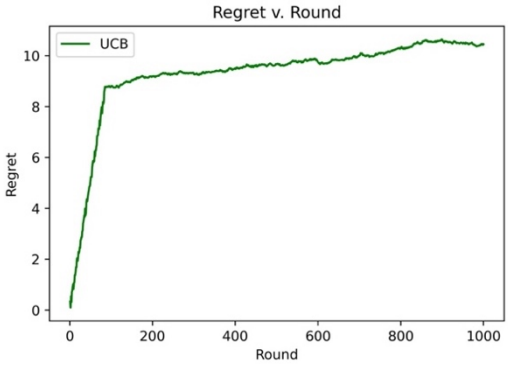

Fig.4 Regret function for Upper Confidence Bound algorithm when $N = 500$ and $T = 1000$.

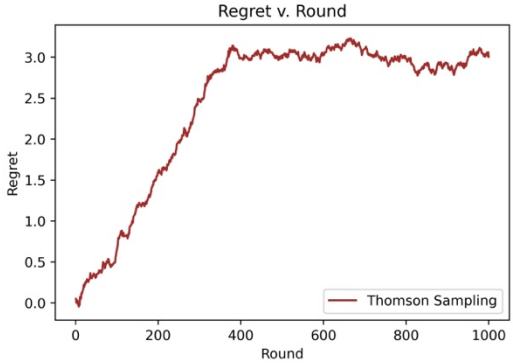

Fig.5 Regret function for Thomson sampling algorithm when $N = 500$ and $T = 1000$.

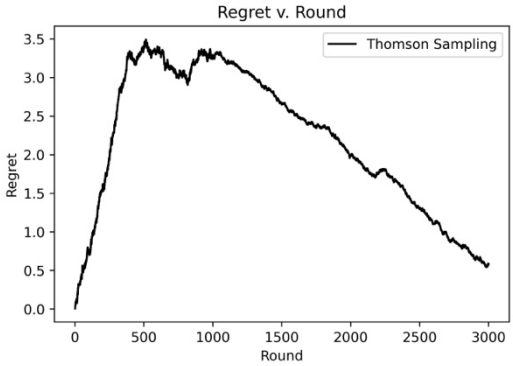

Fig.6 Regret function for Thomson sampling algorithm when $N = 500$ and $T = 3000$.

4.3 Discussion

As Fig.1 suggested, the regret function of the ε-greedy algorithm increases relatively faster during the first approximately 200 rounds. The reason for this relatively more rapid growth occurring at the beginning of the graph is that the algorithm has not gained enough information about the rewards given by the slot machines. As the algorithms get more information about the slot machine, the rate of increase slows down. Then its rate of increasing decreases as more rounds is played. The overall trend of the regret function of the ε-greedy algorithm is roughly linear and has a positive slope. In Fig.2, when the ε value becomes 0.2, the regret function has a similar trend when ε = 0.1 but reaches a lower maximum value at the 1000th round. Therefore, the ε-greedy algorithm with ε = 0.2 shows a greater performance than the ε-greedy algorithm with ε = 0.1. However, when ε = 0.4, Fig.3 suggests that the regret function of ε-greedy algorithm increases more rapidly and reaches a greater maximum value at the 1000th round. A greater ε value adds randomness to the algorithm so choosing a sutible ε value is crucial for ε-greedy algorithm.

In Fig.4, the regret function of the UCB algorithm starts to increase rapidly, and then the rate of increasing suddenly slows. This trend changes around the 100th round. Therefore, it can be inferred that the algorithm gets enough knowledge about the reward distribution starting around the 100th round, so the function begins to change slowly.

In Fig.5, the regret function of the TS algorithm starts increasing until around the 400th round, and then the graph has an overall decreasing rate. Fig.6 further shows this property because the regret function further declines as more rounds are played. These two graphs also support the claim of the logarithmic property of the TS regret function, where the function initially increases and eventually falls to 0.

Comparing the regret functions for the three algorithms, the regret function of the ε-greedy algorithm has the highest range. This means the ε-greedy algorithm accumulates the total regret during the 1000 rounds. UCB algorithm also accumulates more regret than the TS algorithm, but the difference is not as large as the difference between the ε-greedy algorithm and the UCB algorithm.

Further, the regret function of the TS algorithm declines property after certain rounds. The regret functions for the other two algorithms don't show a decreasing trend. In addition, in both of UCB algorithm and TS algorithm, the regret functions have an explicit point where the rate of increasing becomes slower (or the rate of increasing becomes negative for the TS algorithm). The point where the function changes its rate is the round that the algorithm collects enough information about the reward distribution. Although the TS algorithm accumulates less regret than the UCB does, the UCB achieves this point earlier than the TS algorithm does.

5. Conclusion and Future Work

Based on the results and discussions presented above, the conclusions are obtained as below.

(1) The overall performance for this simulation-based regret analysis can be ranked as: ε-greedy algorithm < UCB algorithm < TS algorithm. Therefore, the TS algorithm usually gives the least total regret in situations similar to the stimulation above.

(2) The TS algorithm has a logarithmic regret function. As the number of rounds, the regret function of the TS algorithm will eventually decline to 0. This property is useful when the number of rounds is fairly large.

(3) The trend of the regret function of the ε-greedy algorithm continues to increase at the 1000th round. Similarly, the trend of regret function of the UCB algorithm continues to grow at the 1000th round, but with a significantly lower rate compared to the ε-greedy algorithm. However, the regret function of the UCB algorithm is greater than regret function of ε-greedy algorithm before the 500^{th} round (roughly).

It is suggested that further research on regret analysis of MAB problem can be expanded to other variants of MAB problems, such as multi-armed Bandit problem with known trends and multi-armed bandit problem with covariates as mentioned above. In addition, the mathematical derivation on analysis of regret functions is desired.

References

[1] Shivaswamy, P. & Joachims, T.. (2012). Multi-armed Bandit Problems with History. Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics, in Proceedings of Machine Learning Research 22:1046-1054

[2] Črepinšek, M., Liu, S. H., & Mernik, M. (2013). Exploration and exploitation in evolutionary Algorithms: A survey. ACM computing surveys (CSUR), 45(3), 1-33.

[3] Bouneffouf, D., Rish, I., & Aggarwal, C. (2020, July). Survey on applications of multi-armed and contextual bandits. In 2020 IEEE Congress on Evolutionary Computation (CEC) (pp. 1-8). IEEE.

[4] Wang, Shatian, et al. (2020) Fast Thompson Sampling Algorithm with Cumulative Oversampling: Application to Budgeted Influence Maximization. arXiv: 2004.11963.

[5] Sottocornola G, Stella F, Zanker M. (2021). Counterfactual Contextual Multi-Armed Bandit: a Real-World Application to Diagnose Apple Diseases. arXiv preprint arXiv:2102.04214.

[6] Bouneffouf D, Rish I. (2019) A survey on practical applications of multi-armed and contextual bandits. arXiv preprint arXiv:1904.10040.

[7] Djallel Bouneffouf, Raphael Féraud. Multi-armed bandit problem with known. trend, Neurocomputing. Volume 205, 2016, Pages 16-21, ISSN 0925-2312.

[8] Vianney Perchet. Philippe Rigollet. (April 2013) The multi-armed bandit problem with covariates. Ann. Statist. 41 (2) 693 - 721 .

[9] Huo X, Fu F. (2017). Risk-aware multi-armed bandit problem with application to portfolio.selection.R. Soc. open sci. 4: 171377.

[10] Chunqiu Zeng, Qing Wang, Shekoofeh Mokhtari, and Tao Li. (2016). Online Context-Aware. “Recommendation with Time Varying Multi-Armed Bandit.” In Proceedings of the 22nd

[11] ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16). Association for Computing Machinery, New York, NY, USA, 2025–2034.DOI:https://doi.org/10.1145/2939672.2939878

[12] Chen, J., Xie, Z., & Dames, P. (2022). The semantic PHD filter for multi-class target tracking: From theory to practice. Robotics and Autonomous Systems, 149, 103947.

[12] Chen, J., & Dames, P. (2020). Collision-free distributed multi-target tracking using teams of mobile robots with localization uncertainty. In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 6968-6974). IEEE.

[13]Chen, J., & Park, S. Bernoulli Thompson Sampling-based Target Search Algorithm for Mobile Robots.

[13] Kuleshov, V., & Precup, D. (2014). Algorithms for multi-armed bandit problems. arXiv preprint arXiv:1402.6028

[14] DeVore, R. A., & Temlyakov, V. N. (1996). Some remarks on greedy algorithms. Advances in computational Mathematics, *5*(1), 173-187.

[15] Hao, B., Abbasi Yadkori, Y., Wen, Z., & Cheng, G. (2019). Bootstrapping upper confidence bound. Advances in Neural Information Processing Systems, *32*.

[16] Russo, D. J., Van Roy, B., Kazerouni, A., Osband, I., & Wen, Z. (2018). A tutorial on thompson sampling. Foundations and Trends® in Machine Learning, 11(1), 1-96.

[17] Weisstein, E. W. (2002). “Bernoulli distribution.”Johnson, N. L., Kotz, S. I., & Balakrishnan, N. (1994). Beta distributions. Continuous univariate distributions. 2nd ed. New York, NY: John Wiley and Sons, 221-235.

[18] Kaptein, M. (2015). The use of Thompson sampling to increase estimation precision. Behav. Res 47, 409–423

[19] Chen, Chong et al. (2020) Efficient Neural Matrix Factorization Without Sampling For. Recommendation. ACM Transactions On Information Systems, vol 38, no. 2, pp. 1-28. Association For Computing Machinery (ACM).

[20] Hernández-Lobato, J.M., Requeima, J., Pyzer-Knapp, E.O. & Aspuru-Guzik, A.. (2017). Parallel and Distributed Thompson Sampling for Large-scale Accelerated Exploration of Chemical Space. Proceedings of the 34th International Conference on Machine Learning, in Proceedings of Machine Learning Research 70:1470-1479

[21] Kashyap, R. (1971). Prior probability and uncertainty. IEEE Transactions on information theory, 17(6), 641-650

[22] Monahan, J. F., & Boos, D. D. (1992). Proper likelihoods for Bayesian analysis. Biometrika, 79(2), 271-278.

[23] Agrawal, S., & Goyal, N. (2012, June). “Analysis of thompson sampling for the multi-armed

[24] Bandit problem.” In Conference on learning theory (pp. 39-1). JMLR Workshop and Conference Proceedings.