Performance of Multi-Armed Bandit Algorithms in Dynamic vs. Static Environments: A Comparative Analysis

Boxi Zhao*

College of Art and Science, Santa Clara University, 95053 500 El Camino Real, California, USA

Abstract. This paper conducts a comparative analysis of Multi-Armed Bandit (MAB) algorithms, particularly the Upper Confidence Bound (UCB) and Thompson Sampling (TS) algorithms, and focuses on the performance of these algorithms in both static and dynamic environments. Multi-armed bandit algorithms are instrumental in optimizing decision-making problems. While these algorithms have been studied in a static environment where the reward distribution is constant throughout the problem, real-world issues often have an unstable reward distribution, where the reward distribution may change throughout the process. This paper simulates both static and dynamic environments to evaluate the performance of UCB and TS algorithms by using the MovieLens 1M database. The paper demonstrates that the TS algorithm consistently outperforms the UCB algorithm in both static and dynamic environments. However, both algorithm shows a significantly higher cumulative regret in a dynamic environment compared with a static environment, which is due to the challenges of adapting to changing reward distribution over time. These results provide valuable insight into the application of Multi-Armed Bandit algorithms in real-world environments and highlight the need for further advancement in dynamic adaption for algorithms.

1 Introduction

MAB algorithms have emerged as a critical tool for decision-making problems in different areas where a balance of exploration and exploitation is needed. These algorithms are implemented in healthcare, telecommunication, recommendation systems, and many other fields. In these areas, an optimal decision needs to be made during an uncertain environment. The goal of MAB algorithms is to maximize the cumulative rewards over time by choosing the best actions among a set of possible actions, which is also known as “arms”. For each arm, the possible outcomes are unknown. In recent years, the evolution of real-world applications has introduced new challenges to MAB algorithms, such as a dynamic environment where rewards for each arm change over time. For example, the recommendation system on websites such as YouTube or Amazon. Such a situation is different from a static environment, where the reward for each arm remains constant

* Corresponding author: Bzhao2@scu.edu

throughout the whole decision process. This could happen by using past data to determine future actions. Understanding the performance of MAB algorithms in different environments would improve their application of them in diverse real-world situations.

In recent years, research in MAB algorithms has improved significantly. Various algorithms have been developed to optimize decision-making in different environments in society. Early works, such as a study of Auer, which uses confidence bounds to manage the exploration-exploitation trade-off, laid the foundation for static environments where rewards remain stationary [1]. However, recent works have shifted their focus toward a dynamic environment. For example, Komiyama, Fouché, and Honda highlighted the challenge caused by the variation of reward distribution in a dynamic environment by focusing on a finite-time analysis of the MAB algorithm in society [2]. Similarly, Cavenaghi, Sottocornola, and Stella introduced concept drift-aware algorithms to improve the performance of analysis in a dynamic environment by adapting to the change in reward distribution [3]. Furthermore, Dimakopoulou, Ren, and Zhou explored the use of adaptive inference in online MAB algorithms, which allow them to adjust dynamically as new data is gathered [4]. In contrast, reward distribution in a static environment is easier to model and optimize. Henri, Vlachou, and Thiran demonstrated the effectiveness of MAB algorithms in optimizing performance in hybrid networks with relatively stable conditions in their paper [5]. There is a growing need to compare the performance of MAB algorithms in dynamic and static environments as research continues to advance.

The motivation for this paper comes from the increasing prevalence of nonstationary environments in real-world applications which challenge the traditional assumption that MAB algorithms are aimed for static environments. There remains a gap in comparing and analysing the performance of MAB algorithms in dynamic and static environments besides significant progress in adapting the algorithms to dynamic environment. This paper focus on filling this gap by conducting a comprehensive comparative analysis of MAB algorithms in dynamic and static environments. The framework of this paper is divided into three parts. First, the paper will introduce a theoretical analysis of the fundamental difference between dynamic and static environments with their influences on MAB algorithms. Second, the paper will introduce the algorithms compared in the experiments. Then, experiments with different MAB algorithms will be established in both dynamic and static environments. Last, the paper will conclude the development of recommendations for choosing MAB algorithms based on the difference in environments. By combining theoretical insights with practical evaluation, this paper aims to provide a clearer understanding of how MAB algorithms can be optimized for different environmental conditions, ultimately advancing their applicability in diverse real-world scenarios.

2 Literature review

The MAB algorithm is essential for addressing problems involving exploration and exploitation in different environments especially in static environments where the reward distribution remains constant. Auer's work on confidence bounds significantly advanced this by balancing exploration and exploitation to "maximize cumulative rewards [1]. Henri et al. demonstrated the potential of the MAB problem in static hybrid networks, where it can efficiently allocate resources without adapting to dynamic changes [5]. Kuleshov and Precup further analysed MAB algorithms, contributing foundational techniques still relevant for static environments [6]. Han's recent review emphasized the wide range of MAB applications and the unique challenges in both static and dynamic environments. Han's work highlights the algorithms' extensive utility in areas like resource management and network optimization [7].

In a dynamic environment, MAB algorithms encounter significant challenges since these algorithms are initially used in a static environment. Researchers have developed adaptations to address this. Komiyama et al. provided a finite-time analysis of globally nonstationary MAB problems. They focused on environments with periodically changing reward distribution. The study provides important theoretical insights into how MAB algorithms can adapt to these environments with high performance over finite time horizons [2]. Cavenaghi et al. introduced a concept drift-aware algorithm specifically for nonstationary environments. Their approach uses techniques to detect and adapt to changes in reward distributions. This study ensures the algorithms remain competitive even when the environment shifts unpredictably [3]. Manzoni and Calafate further explored this by integrating LLM-informed strategies. They enhanced MAB algorithm performance in a nonstationary environment by dynamically incorporating large language model insights [8]. Additionally, Gutowski et al. presented a portfolio approach to MAB selection which dynamically chooses the best MAB strategy for real-time recommendation environments [9]. Chen et al. highlight how MAB algorithms can adapt and excel in adversarial dynamic conditions by bridging adversarial and nonstationary frameworks [10].

3 Methodology

This study compares the performance of two MAB algorithms using the MovieLens 1M dataset. The two algorithms are the Upper Confidence Bound algorithm and the Thompson Sampling algorithm. The objective of the comparison is to observe how each algorithm adapts to different environments.

3.1 Dataset

The Movie Lens 1M dataset contains 1000209 anonymous ratings of 3883 movies provided by 6040 MovieLens users who joined MovieLens in 2000. Each user has rated at least 20 movies with a rating scale from 1 to 5 where 1 is the lowest and 5 is the highest [11].

For this study, movie lens and movie IDs are chosen as the “arm” of the bandit, and the rating is chosen as the reward. Specifically, the ratings of users for a movie represent the rewards of pulling the arm of the bandit.

3.2 Evaluation

The performance of the algorithm used in the experiment is evaluated by cumulative regret, which measures the difference between the actual reward accumulated by the algorithm and the best reward the algorithm can get through choosing the best arm consistently. The lower the cumulative regret is, the better the performance of the algorithm is.

3.3 Algorithms

3.3.1 Upper Confidence Bound algorithm

The Upper Confidence Bound algorithm, also called as UCB algorithm, will first select each arm once, which will give each arm’s potential reward an initial value. Then, the algorithm will choose the arm with the highest UCB value, which is calculated in Formula 1:

$$UCB_i(t-1) = \mu_i(t-1) + \frac{B}{2}\sqrt{\frac{4\log n}{T_i(t-1)}} \quad (1)$$

$UCB_i(t-1)$ represents the Upper Confidence Bound, or the UCB value, for arm i at time step $t-1$. The UCB value is used to decide which arm to select in the next round. $\mu_i(t-1)$ is the estimated mean of arm i at round $t-1$. It captures the exploitation part of the algorithm. $T_i(t-1)$ is the number of times arm i has been chosen at round $t-1$. B is a predefined scale parameter which will affect how much exploration is encouraged. In this experiment, B equals the difference between the highest and lowest reward, which is 4. n represent the total number of rounds in the experiment. The UCB value represent a combination of current estimated reward of an arm, which is the estimated mean, and an additional bonus of uncertainty of other arms, which is the rest part of the equation. The bonus decreases as the arm is pulled more often. This encouraged the exploration of less-selected arms. The UCB algorithm chooses the arm with the highest UCB value, which ensure a balance between exploration and exploitation.

3.3.2 Thompson Sampling algorithm

TS algorithm samples a potential reward from the distribution for each arm at each round. This will reflect both the observed average reward and the uncertainty. Then, the algorithm selects the arm with the highest sampled reward and updates the distribution for that arm based on the observed outcome. The normal distribution allows straightforward updates of mean and variance, which makes it easier to update the belief about the reward distribution efficiently. If the reward distribution is skewed, the normal distribution may not capture the full range of variability, which will make the algorithm choose suboptimal exploration-exploitation trade-offs.

$$F_i(t) \sim N\left(\mu_i(t), \frac{B^2}{4T_i(t)}\right) \quad (2)$$

In Formula 2, $F_i(t)$ is a random sample drawn from a distribution to represent the estimated reward for arm i at time t. It reflects both the mean estimate of the reward and the uncertainty in that estimate. $\mu_i(t)$ represents the empirical mean reward for arm i at time t. It is the best estimate for the mean reward of arm i based on the observed data so far. $\frac{B^2}{4T_i(t)}$ is the variance of the normal distribution. It captures the uncertainty about the expected reward for arm i. B represents the range of possible rewards, 4 in this experiment. $T_i(t)$ represents the number of times arm i has been pulled up to time t. As $T_i(t)$ increases, the variance decreases, meaning the algorithm becomes more certain about the value of that arm.

3.4 Experiment setup

UCB and TS algorithms will be tested in two different environments, which are static and dynamic, through three different environment setups. Each environment is designed to stimulate real-world scenario where the user preference and reward distribution may either remain constant or changes over time. This will help to illustrate how well the algorithm can achieve under different situations.

For each setup, the algorithm will be tested through 100 independent experiments, each with 400000 rounds. The result of each experiment is measured by the cumulative regret. The final result of the algorithm under the environment is the average of the 100 experiments to ensure robustness and reduce the randomness in each run. The large number of rounds can

reveal the performance of each algorithm by showing a logarithmic curve instead of a linear one. The repetition of the experiment reduces the randomness of the result. The difference between each setup is the arm. First setup pick 7 movie genres out of 15 as the arm through the experiment. The second and third picked movie IDs as the arm of the experiment, where the second set up chose 7 different movies and the third chose 70 different movies from 3883 movies.

In a static environment, the average reward for each arm is different from each other. For each setup, a group of arms is chosen as the initial setup and is fixed throughout the experiment. Both UCB and TS algorithms are executed for 400000 rounds in a static environment. The primary goal for each algorithm is to efficiently learn which genre has the best reward distribution and focus on exploiting the best arm.

In a dynamic environment, the initial reward distribution is the same as in a static environment, but at around 100000, 200000, and 300000, the reward distribution will be randomly changed into another group of different reward distributions with the same number of arms as in the initial setup. The round is changed based on the performance of the algorithms, where both algorithms show a clear logarithmic behavior after 100000 rounds. When the interval between changing rounds is small, the outcome will indicate a linear form, which does not reflect the traits of the algorithms. At each time the reward distribution is modified, the algorithms need to explore each arm again. The variation in reward distribution is used to stimulate a situation of a changing environment and show how quickly each algorithm reacts to adapt to the dynamic environment.

3.5 Implementation of algorithms

At the start of each experiment, all arms are initialized with no reward information. Each arm is selected once to gather an initial reward for the UCB algorithm. For the TS algorithm, the belief about the reward distribution for each arm starts with a prior distribution based on a mean, which is used as the initial value in the calculation, and a variance, which reflects uncertainty.

The algorithm selects one arm based on the calculated values each round. UCB algorithm selects the arm that has the highest UCB value, which is calculated using the equation that is listed previously. The reward obtained from this action will be updated to the UCB value of the arm that is chosen. TS algorithm selects the arm with the reward that is highest in the rewards that are sampled from the posterior distribution for each arm. The posterior distribution is updated after each round based on the value obtained.

At the beginning of the experiment and once the reward distribution is modified, the mean for each arm is calculated to get the optimal reward of the environment. This value is only used for calculating regret and is not used as a factor in the algorithm to decide on actions. After the arm is selected for each round, a reward will be obtained randomly from the arm. The regret is calculated based on this value and the optimal reward for each round. The regret will then be updated to the cumulative regret and stored.

4 Experiment

4.1 Experiment results

The primary goal of the experiment is to evaluate and compare the performance of two algorithms, the UCB algorithm and the TS algorithm, in a static and dynamic environment, using cumulative regrets. First, the UCB algorithm will be compared with the TS algorithm in a static environment, where the lower the cumulative regret is, the better the performance

is. Second, both algorithms will be compared with themselves in static and dynamic environments. The difference between the cumulative regret will show the adaptability of each algorithm in a dynamic environment. Also, the study seeks to understand the long-term behaviour of the UCB and TS algorithm over 400000 rounds through the cumulative regret by determining how well each algorithm balanced the exploration and exploitation phase.

For the UCB and TS algorithms, each algorithm will be implemented in both static and dynamic environments with three different setups. In total, there will be twelve different cases. In each case, the algorithm will be implemented 100 times with 400000 rounds. The cumulative regret will be calculated each round. The average cumulative regret over 100 experiments with an error bar of 1 standard deviation will be presented. The result is listed below.

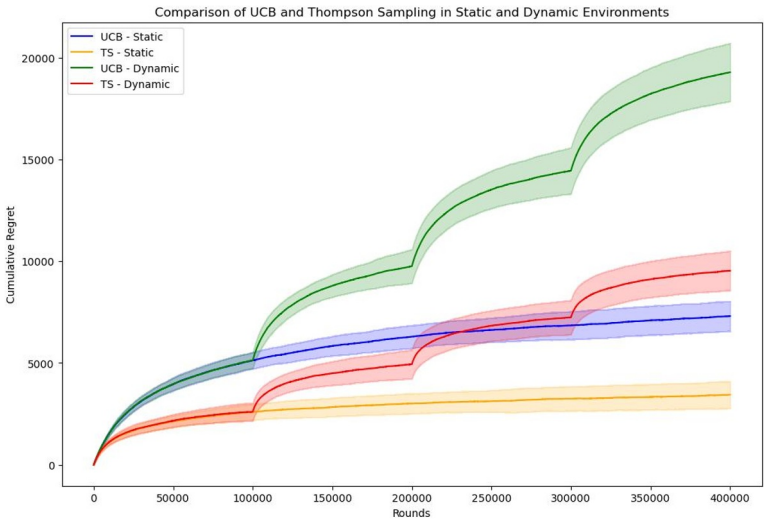

Fig.1. Cumulative regret with 7 movie genres as the arm

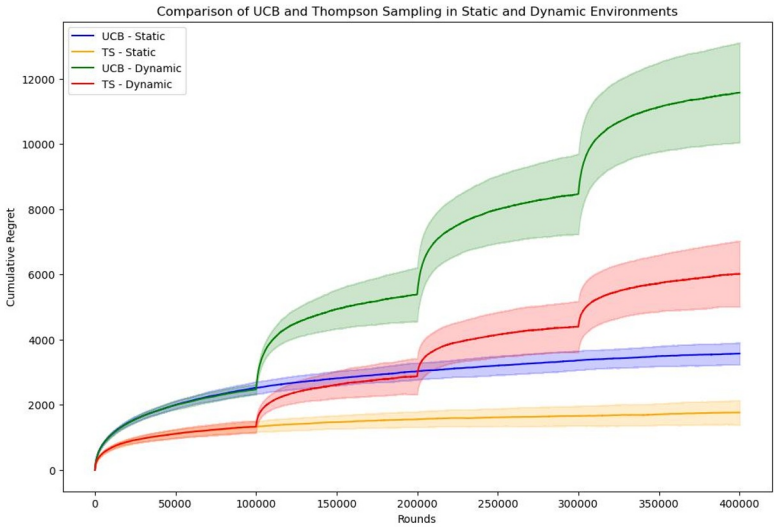

Fig.2. Cumulative regret with 7 movie IDs as the arm

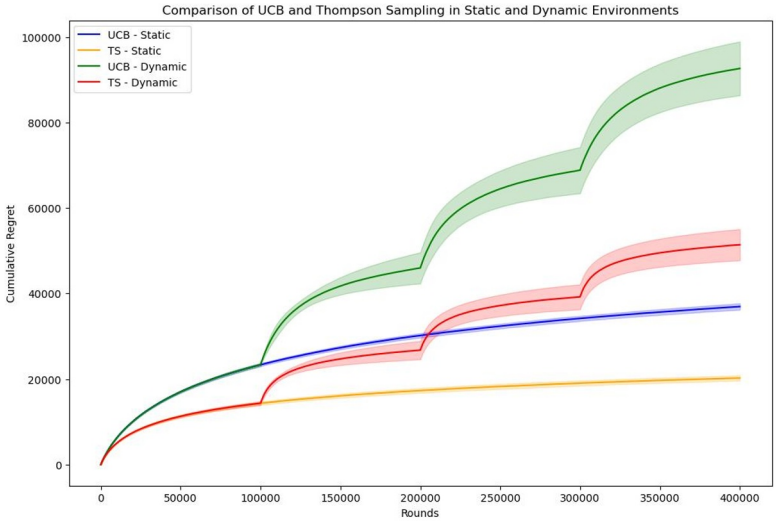

Fig.3. Cumulative regret with 70 movie IDs as the arm

For all three figures, the blue line with shadow represents the cumulative regret with error bar of one standard deviation of UCB algorithm's performance in static environment. The yellow one represents TS algorithm in static environment, green line represents UCB algorithm in dynamic environment, and red line for TS algorithm in dynamic environment. The x axis of the graph represents the rounds of the experiment, and the y axis represent the cumulative regrets of algorithms.

For the Figure 1, the graph compares the cumulative regret of UCB and TS algorithms in both static and dynamic environments when the movie genre is the arm. In both environments, the TS algorithm has better performance, which indicates its adaptability to the environment, particularly in static environments since it has the lowest cumulative regret.

For the Figure 2, the comparison between different algorithms in different environments is the same as the previous one. The cumulative regret for all situations is lower than in the first experiment due to the difference in reward distribution in the data set.

For the Figure 3, the comparison for both algorithms in both environments is the same as the previous two. The cumulative regret is higher than in the second experiment. This is caused by the increase in number of arms in the third experiment.

4.2 Comparison

When there are seven arms, the algorithms have smaller space to explore since there are fewer arms. However, since the mean reward for each genre is close to each other and only a small amount of reward distribution is selected, the algorithm may switch between arms more frequently and lead to higher standard deviation. The algorithm converges faster since there are fewer arms to explore.

When the number of arms increases to 70, the algorithms have a longer exploration phase due to the larger number of arms. This also leads to a higher cumulative regret in the exploration phase. However, once the algorithms explore enough, the cumulative regret converges with a smaller standard deviation.

Both algorithms have a significantly smaller cumulative regret in a static environment. This is caused by the stability of the reward distribution, where once the algorithms explore and find the best arm, they can exploit the arm in the rest of the experiment. In a dynamic

environment, the algorithms need to explore the reward distribution once it has changed, which leads to a higher cumulative regret.

In all six different setups with the environment, the TS algorithm has a better performance compared with the UCB algorithm. This is a result of the probabilistic approach to the exploration and exploitation of the TS algorithm, which allows the TS algorithm to balance exploration and exploitation more efficiently and adapt to changing environments quickly.

5 Conclusion

In the experiments, both UCB and TS algorithms tend to have a decreased standard deviation with increased cumulative regrets as the number of arms increases. Other than that, in a static environment, both UCB and TS algorithms demonstrated efficiency in optimizing the selection of actions. TS algorithm results in better performance since it has lower overall cumulative regrets in all three setups in a static environment. Also, the TS algorithm still has a better performance in a dynamic environment compared with the UCB algorithm, but both algorithms result in significantly higher cumulative regrets compared with the performance in a static environment. The difference in number of arms will also influence the cumulative regrets of both algorithms. With more arms, both UCB and TS algorithms will spend more rounds on exploring arms, which leads to a significant increase in cumulative regrets. But still, the comparison remains the same that both algorithm has better performance in static environment, and TS algorithm has the best performance. The difference in outcomes in dynamic and static environments suggests that both the UCB and TS algorithms have better performance in static environments compared with dynamic environments. This is caused by the variation of reward distribution in a dynamic environment, which requires the algorithm to explore the rewards again and results in significantly increased cumulative regrets.

References

1. P. Auer, Using confidence bounds for exploitation-exploration trade-offs. J. Mach. Learn. Res. 3(Nov), 397–422 (2002)
2. J. Komiyama, E. Fouché, and J. Honda, Finite-time analysis of globally nonstationary multi-armed bandits. J. Mach. Learn. Res. 25(112), 1–56 (2024)
3. E. Cavenaghi, G. Sottocornola, F. Stella, et al., Nonstationary multi-armed bandit: Empirical evaluation of a new concept drift-aware algorithm. Entropy 23(3), 380 (2021)
4. M. Dimakopoulou, Z. Ren, and Z. Zhou, Online multi-armed bandits with adaptive inference. Adv. Neural Inf. Process. Syst. 34, 1939–1951 (2021)
5. S. Henri, C. Vlachou, and P. Thiran, Multi-armed bandit in action: Optimizing performance in dynamic hybrid networks. IEEE/ACM Trans. Netw. 26(4), 1879–1892 (Aug. 2018) DOI: 10.1109/TNET.2018.2856302.
6. V. Kuleshov and D. Precup, Algorithms for multi-armed bandit problems. arXiv preprint arXiv:1402.6028 (2014)
7. Y. Han, Comparative evaluation, challenges, and diverse applications of multi-armed bandit algorithms. Highlights Sci. Eng. Technol. 94, 206–210 (2024) DOI: 10.54097/jdcjkj94.
8. P. Manzoni and C. T. Calafate, LLM-informed multi-armed bandit strategies for non-stationary environments. Electronics 12(13), 2814 (2023) DOI: 10.3390/electronics12132814.

9. N. Gutowski, T. Amghar, O. Camp, et al., Gorthaur: A portfolio approach for dynamic selection of multi-armed bandit algorithms for recommendation. In Proc. 2019 IEEE 31st Int. Conf. Tools Artif. Intell. (ICTAI), 1164–1171 (2019)
10. N. Chen, S. Yang, and H. Zhang, Bridging adversarial and nonstationary multi-armed bandit. arXiv preprint (2022) DOI: 10.48550/arXiv.2201.01628.
11. F. M. Harper and J. A. Konstan, The MovieLens datasets: History and context. ACM Trans. Interact. Intell. Syst. 5(4), Article 19, 19 pages (Dec. 2015) DOI: http://dx.doi.org/10.1145/2827872.