Exploring Bandit Algorithms in User-Interactive Systems

Influence of Delay on Contextual Multi-Armed Bandits

Dragos-Cristian Arsene[1]

Supervisor(s): Julia Olkhovskaya[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 23, 2024

Name of the student: Dragos-Cristian Arsene
Final project course: CSE3000 Research Project
Thesis committee: Julia Olkhovskaya, Ranga Rao Venkatesha Prasad

Abstract

Delay is a frequently encountered phenomenon in Multi-armed bandit problems that affects the accuracy of choosing the optimal arm. One example of this phenomenon is online shopping, where there is a delay between a user being recommended a product and placing the order. This study investigates the influence of delay in contextual multi-armed bandit settings, focusing on the performance of the OTFLinUCB algorithm, an adaptation of the LinUCB algorithm designed to handle delayed feedback. The cumulative regret associated with various bandit algorithms, including UCB, Exp3, LinUCB, and OTFLinUCB, is examined under different delayed environments. Experiments conducted using artificial data generated to simulate real-world scenarios reveal that while delay cannot be entirely mitigated, its impact can be minimized through the strategic selection of hyperparameters, particularly the time window size for OTFLinUCB. The findings indicate that OTFLinUCB's performance is highly dependent on the chosen window size, which balances the trade-off between estimator accuracy and memory consumption. A method for determining the optimal window size is proposed by introducing an intermediary distribution-agnostic term, conversion rate. Empirical evidence suggests that maintaining a conversion rate of around 70% achieves a balance between minimizing cumulative regret and optimizing memory usage. As a result, this research contributes to the understanding of the delays' effect on bandit algorithms in contextual environments and offers practical guidelines for tuning OTFLinUCB's time window.

1 Introduction

This section provides some background for the subject at hand in Subsection 1.2 and then goes on to present some relevant work that can be studied to get a broader knowledge of the subject.

1.1 Background

The Multi-armed bandit(MAB) is a problem in which an agent is tasked with choosing between different actions(also called arms) over discrete time steps, each action having its reward distribution. The objective is for the agent to maximize the cumulative reward, or minimize the cumulative regret. In other words, it is a decision-making problem concerned with finding a balance between the exploration-exploitation trade-off.

It was first formulated in an article published in 1933 in *Biometrika*, getting its current name in honour of the One-armed bandit slot machine. While the problem was first proposed to improve how clinical trials are conducted, recently, with the advent of the Internet and the need for good recommendation systems, the problem has seen a resurgence in published research papers. While there is a theoretical bound for the optimal regret, no algorithm performs best in all situations.

MAB is formulated generically to simplify analyzing the problem from a theoretical angle, but multiple versions exist to adapt to real-life use cases. In practice, rewards are often not given instantaneously for actions, introducing a delay in the feedback that the algorithm receives and processes to improve its arm choice accuracy. This research paper proposes to answer the main question of how delay affects cumulative regret in contextual environments. Moreover, it also examines how the hyper-parameters of these algorithms influence the exploration-exploitation trade-off and what steps need to be taken to find a balance between these two factors.

To answer the research questions, this paper will explore four algorithms: *UCB* [1], *Exp3* [2], *LinUCB* [3], and *OTFLinUCB* [4]. A more in-depth description of the problem and the metrics used is given in the Formal Problem Description section. The Methodology section explains how artificial data is generated and what frameworks are used. The setup for running the experiments and the results of these experiments are presented in the Experimental Setup and Results section. Then, the Discussion section interprets these results and proposes a method for choosing suitable hyperparameters for *OTFLinUCB*. Finally, the Conclusion section synthesizes all findings, answers the research questions and discusses possible improvements. The pseudocodes of the discussed algorithms are also provided in the Appendix section.

1.2 Related work

The subject of delay has been studied in numerous settings, with previous works laying the groundwork for representing delay in a form that can be better studied and integrated into existing or novel algorithms. One of the earliest ones is [5], which compares *UCB* with other popular algorithms in a setting where rewards are given with a 10, 30, or 60-minute delay. That was not robust enough and does not mimic the true behaviour of, for example, online recommendation systems. In recommendation systems, the concept of conversion introduced by [6] and [7] provided a way to analyze how rewards not being observed at all can be discarded. This conversion concept was then adapted to Multi-armed bandit problems as the censored feedback variable, that can be adapted for delay. This has led to different ways of handling unknown delay, such as skipping rounds with large delay, [8] or replacing observed rewards with surrogate rewards [9]. Moreover, there are conflicting findings on whether the impact of delay increases with the time horizon: [10] says it does, while [11] concludes it does not.

2 Formal problem description

In this section, the Multi-armed Bandit problem is given a more formal description in Subsection 2.1 and then, in Subsection 2.2, explains what metrics will be used to assess the accuracy of the studied algorithms.

2.1 Problem overview

While the slot machine example given before is intuitive enough for an introduction to the problem, a more rigorous formulation has to be done to analyze it properly. One agent has the choice between *K* actions. Each action has its reward distribution and mean reward hidden from the player. The player chooses an action at each time step and plays for *T* discrete time steps, where *T* is a time horizon not known in advance.

The problem can be divided into two settings:

- non-contextual setting: actions sample rewards from probability distributions or functions. Estimating each action's mean is enough to choose the best action.
- contextual setting: actions provide a different context vector with each time step as input for a fixed reward function. In the scope of this paper, only linear combinations are used as a reward function. Estimating the parameters of the reward function is enough to estimate each action's reward

2.2 Metrics used

Two metrics are used: cumulative reward and cumulative regret. These are not used by the algorithms at runtime but are calculated by the evaluators responsible for running them and present a useful tool for comparing the performance of different algorithms. The cumulative reward at time step *t* is the sum of all rewards up to that point. The objective of the player is to maximize reward. More interest presents the cumulative regret metric, which is preferred over the reward because it provides a more objective way of comparing different arm choices.

In non-contextual environments, regret is expressed as:

$$R_t = \mu^* - \mu_{a_t} \tag{1}$$

, where μ^* is the highest true mean reward among all arms and μ_{a_t} is the true mean reward of the chosen arm.

In a contextual setting, regret is defined as:

$$R_t = r_t^* - r_{a_t} \tag{2}$$

, where r_t^* is the highest reward among all arms at time t and r_{a_t} is the reward given by the chosen arm at time t.

The cumulative regret up to time T, denoted as CR_T, is given by:

$$\text{CR}_T = \sum_{t=1}^{T} R_t \tag{3}$$

There are two important observations to make when analyzing regret:

- Regret measures an algorithm's accuracy in estimating the best arm (the arm with the highest true mean in non-contextual settings, or the highest reward in contextual settings) at each time step. It quantifies how far the algorithm's choice is from the optimal choice and provides a better sense of scale than reward.
- Visualising cumulative regret reveals insights into the learning process of an algorithm. Analyzing the growth rate of cumulative regret is an effective way to compare the accuracy of an algorithm's estimators at different time steps. An algorithm that reaches convergence or near-convergence translates to a good enough accuracy for its estimators to effectively choose the best arm frequently(still not always because of exploration).

3 Methodology

To simulate a wide range of use cases, the data used for running the algorithms is artificial and generated so it simulates real data. As it does not depend on time, it is generated and stored for later use at the start of a run.

Here is how all variables are sampled:

- θ is the vector containing the parameters of the linear reward function. It is set at the beginning of a run and has to be estimated by *LinUCB* and *OTFLinUCB*.
- delay is given by:

$$d_{a,t} = X_a(t), \quad X_a \sim \mathcal{D}_a, \quad \text{support}(\mathcal{D}_a) = \mathbb{N}_0 \tag{4}$$

- context is given by:

$$\begin{aligned} c_{a,t} &= X_a(t), \quad X_a \sim \mathcal{N}(\bar{\mu}_a, \Sigma), \\ \|\mu_a\|_2^2 &\leq 1, \quad \Sigma = \lambda \cdot I \end{aligned} \tag{5}$$

- observed reward is given by:

$$r_{a,t} = \langle c_{a,t}, \theta \rangle + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, 0.1) \tag{6}$$

SMPyBandits [12] is a well-established Python module for *MAB* problems. It provides extensive analysis tools and configuration options for arms and allows parallelization, which helps with averaging a suitable number of repetitions, thus eliminating statistical anomalies. While *UCB* and *Exp3* were already implemented in the module mentioned above, the other two algorithms were implemented from the ground up and integrated into the existing code base [13].

Each algorithm is run for 50 repetitions and the results are averaged and plotted. To illustrate the variability and provide a 95% confidence interval, the mean cumulative regret with shaded regions representing two standard deviations above and below the mean is plotted. These results are studied and compared, focusing on how delay impacts cumulative regret. This approach provides an effective strategy for tuning the hyper-parameters of the *OTFLinUCB* algorithm.

4 Experimental Setup and Results

This section focuses on the setup used to run experiments and the results used as a reference point for Section 5. It first explains how the discussed algorithms are adapted for being run in the same environment in Subsection 4.1, after which Subsection 4.2 provides an insight into the hyperparameters that change the behaviour of said algorithms. Then, Subsection 4.3 explains how the delay is generated and how the algorithms handle it. Finally, plots with different environments and hyperparameters are provided in Subsection 4.4.

4.1 Discussed algorithms

UCB and *Exp3* are designed for non-contextual environments, *LinUCB* is the state-of-the-art algorithm for contextual environments, and *OTFLinUCB* is an adaptation of *LinUCB* that takes into account delay. These algorithms perform three operations at each time step: choose an arm, observe the arm's reward, and update the estimators based on the observed reward. In the case of delayed feedback, the first and second steps do not happen concurrently.

Since only *OTFLinUCB* can handle delay, the evaluator was modified such that all other discussed algorithms observe rewards immediately. This is done to assess the effect of delay on cumulative regret and does not exhibit the behaviour of *UCB*, *Exp3*, and *LinUCB* in a true delayed-feedback setting. This decision results in two loose bounds that can be used to assess the effects of delay on cumulative regret:

- Upper bound: *UCB* and *Exp3* cannot estimate the reward function parameters, both exhibiting similarly poor accuracy when choosing an arm
- Lower bound: *LinUCB* can estimate the reward function parameters, while not having to handle the uncertainty introduced by the delay. As such, it should perform at least as well as *OTFLinUCB*

The amplitude of delay and the window size of *OTFLinUCB* influence how close *OTFLinUCB* gets to either its lower or upper cumulative regret bound.

4.2 Hyperparameters

Hyperparameters control how the algorithms behave and are set at the beginning of a run. Here are the most relevant:

- α and γ: exploration parameters that control the exploration-exploitation trade-off
- *m*: time window of *OTFLinUCB*. If a reward is not observed after m time steps, it is ignored
- $\lambda - regularizer$: used in *LinUCB* and *OTFLinUCB*, it prevents overfitting and ensures numerical stability in the case of matrix inversion

While all *hyperparameters* can affect an algorithm's accuracy, α, γ, and $\lambda - regularizer$ have tried and tested values suitable for most settings: $\lambda = 1$ and $\gamma = \alpha = 0.01$. Different values were tried for these hyper-parameters, but the default values always performed better. As such, the paper will use them from now on. The focus shifts towards the hyperparameter that significantly influences cumulative regret and is environment-dependent, *m*. One last thing to note is that, in the original implementation of *OTFLinUCB*, α is not set at the beginning of a run, but updated dynamically using the contexts saved in the time window, as so:

$$f(t,\delta) = \sqrt{\lambda} + \sqrt{2\log\left(\frac{1}{\delta}\right) + d\log\left(\frac{d\lambda + t}{d\lambda}\right)} \tag{7}$$

$$\alpha_{t,\delta} = 2f(t,\delta) + \sum_{s=t-m}^{t-1} \|A_s\|_{V_t(\lambda)^{-1}} \tag{8}$$

Regardless of the values provided for λ or δ, $f(t,\delta)$ and consequently $\alpha_{t,\delta}$ become excessively large, causing exploration to dominate over exploitation. That results in a significant increase in cumulative regret, so the constant value of *0.01* was used instead of formulas 7 and 8.

4.3 Delays

As noted by [14], delays often have heavy tails, meaning that using a small window size may result in too many rewards being ignored. Furthermore, [15] observes that another complicating factor is the unobserved negative feedback (i.e., the delay can be infinite, in which case rewards are never observed, but the algorithm cannot determine if this is the case). This leads to the conclusion that a time window is essential to prevent an algorithm from storing contexts that will never convert to a reward. Additionally, the time window should be large enough to counteract the heavy right tails often found in the probability distributions of delays in practice.

For each time step and each action, a delay, context, and reward are simultaneously drawn using equations 4, 5, and 6. Then, they are added to a buffer that reveals contexts immediately and rewards to *OTFLinUCB* with delay. This paper only studies reward-independent delays. According to [16], reward-dependent delays impact the confidence bounds of algorithms, but this case is outside the scope of this study.

The environment reveals a reward and its associated delay, but not its associated context, using *censored feedback*. Censored feedback improves exploration and accuracy, as noted by [17]. This approach is implemented in *OTFLinUCB* using a window for storing past actions' contexts, which can thus link rewards with the context that generated them. A naive implementation would store all past contexts at each time step, which is inefficient. Instead, a parameter limits how far in the past a reward can be observed before being discarded. A time window of suitable size maintains regret minimization while improving memory usage.

4.4 Experiments

The following plots show each algorithm's performance, focusing on cumulative regret and the effects of various delay models and window sizes. These visualizations highlight the effect of outside factors and hyper-parameters on the regret of *OTFLinUCB*:

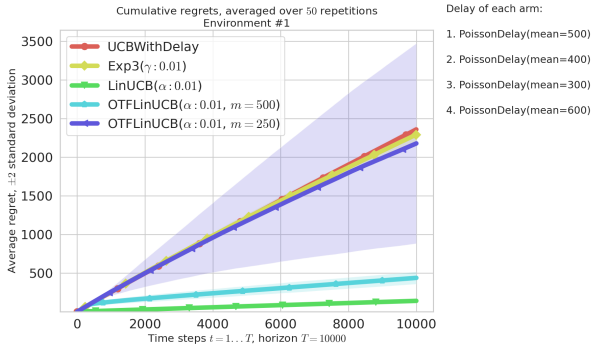

Figure 1: One OTFLinUCB approaching the higher bound and one approaching the lower bound

In fig. 1, it can be observed how the *OTFLinUCB* algorithm with two different time window sizes can have vastly differing cumulative regrets. While *OTFLinUCB(m = 250)* is not able to observe enough rewards to update its estimators successfully and its accuracy is comparable to that of *UCB* and *Exp3*, *OTFLinUCB(m = 500)* does so and its accuracy gets much closer to that of *LinUCB*.

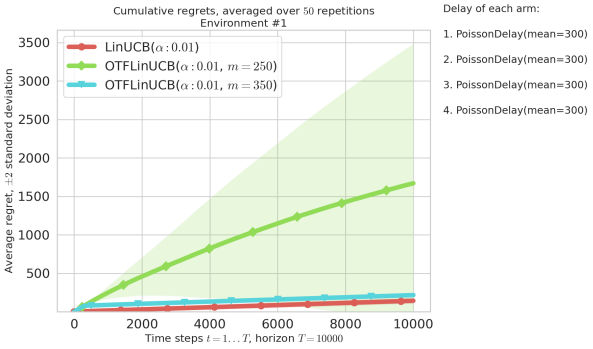

Figure 2: Poisson delay

Fig. 2 demonstrates how two seemingly similar time window sizes (250 and 350) lead to significantly different cumulative regrets. Most values are concentrated around the mean for Poisson delays with a large λ. Thus, a time window size slightly smaller than the mean results in most rewards being ignored, while one marginally larger than the mean ensures most rewards are observed.

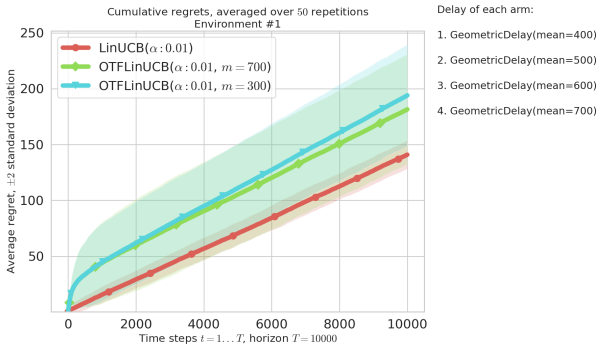

Figure 3: Geometric delays with lower means

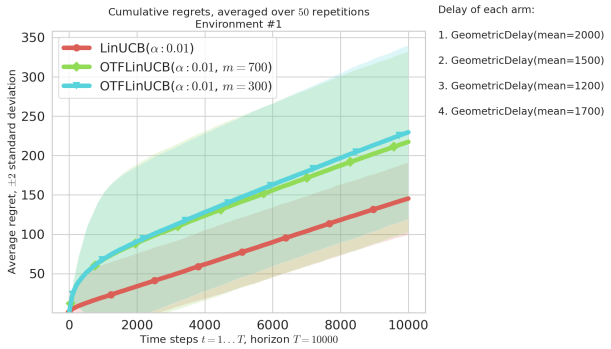

Figure 4: Geometric delays with higher means

Figures 3 and 4 reveal how a sudden increase in the mean of Geometric distributions does not translate to a significant increase in the cumulative regret. The reason is that Geometric delays are heavily skewed to the right, so higher means do not significantly increase the number of ignored rewards.

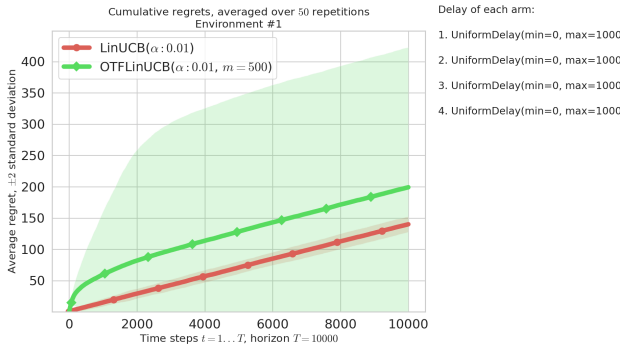

Figure 5: Uniform delays

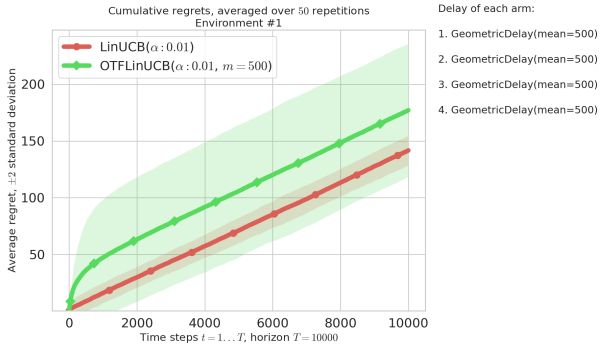

Figure 6: Geometric delays

Figures 5 and 6 show how the type of probability distribution modelling rewards influences cumulative regret. While both plots show probability distributions with equal means, the Geometric distribution concentrates more values to the left, which results in an increased number of observed de-

lays, and thus decreased cumulative regret compared to the Uniform distribution.

5 Discussion

In this section, a parameter will be introduced in Subsection 5.1 that can then be used to reliably calculate a suitable time window size, as seen in Subsection 5.2.

5.1 Conversion rate

In Subsection 4.4, it has been shown how the probability distribution modelling delay and the window size m influence regret. There are two possible reasons for why cumulative regret increases when delay is introduced:

- A reward is observed: estimator updates in *OTFLinUCB* happen later than in *LinUCB*, particularly affecting the arm choice accuracy of earlier time steps(illustrated by the steep growth rate of cumulative regret of *OTFLinUCB* at the beginning of the round in figures 3, 4, 6, and 5, regardless of the time window size). This is only influenced by the underlying probability distribution of delay.
- A reward is not observed: rewards delayed beyond *m* are not considered in *OTFLinUCB*, leading to estimators being updated less often throughout the round. This is influenced by the value of *m* in relation to the probability distribution of delay. In Figure 4, the gap between the cumulative regrets of *OTFLinUCB* algorithms with different time window sizes widens as the time horizon increases, which confirms the observation made in [10] that the impact of delay increases with the time horizon.

A probability distribution modelling delay is predetermined and cannot be altered to enhance the algorithm's accuracy. The only adjustable parameter is the window size *m*. The challenge lies in determining an optimal *m* that effectively balances a high conversion rate with low memory usage. Thus, what can be controlled by the algorithm is the number of observed rewards relative to the total number of rewards.

To make the comparison of window size delay distribution-agnostic, the conversion rate per arm term is introduced:

$$\rho_a = \frac{\text{number of observed rewards for arm } a}{\text{number of total rewards for arm } a} \tag{9}$$

For a big enough time horizon, (9) can also be expressed as:

$$\rho_a = F_{X_a}(m) = \Pr(X_a < m) \tag{10}$$

where X_a represents the delay distribution for an arm a.

Even with an infinite *m*, greater regret is observed for *OTFLinUCB* than for *LinUCB* due to delayed estimator updates. Consequently, *LinUCB* is not suitable as a baseline for comparing different values of *m* in *OTFLinUCB*. Instead, a version of *OTFLinUCB* designed to achieve a 100% ρ_a represents the baseline. Discrete Uniform distributions will model delay because they are bounded to the right and there is a linear relation between window size and conversion rate.

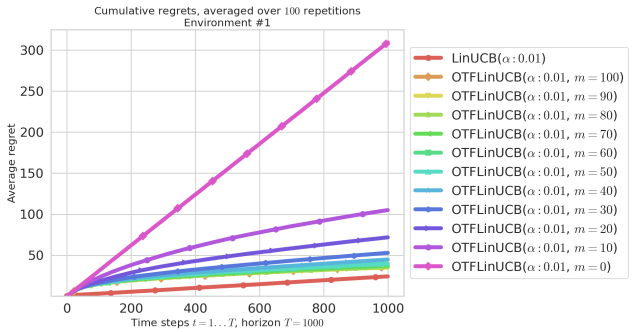

Figure 7: $\mathcal{U}_{[0,100]}$ with corresponding conversion rates from 0% to 100%

Table 1: Conversion rates and their corresponding regret(Part 1)

ρ	**100%**	**90%**	**80%**	**70%**	**60%**
CR	35.836	36.057	36.29	37.472	43.264

Table 2: Conversion rates and their corresponding regret(Part 2)

ρ	**50%**	**40%**	**30%**	**20%**	**10%**
CR	45.904	51.06	55.091	66.128	105.59

Figure 7 and tables 1 and 2 show how cumulative regret increases significantly when ρ_a falls below 70%, and the threshold is around 20%-30%. Thus, the optimal *m* should ensure no more than 70% of rewards are ignored for any arm.

5.2 Time window size(m)

Using (10), *m* can be calculated using the quantile function:

$$m = Q(\rho_a) = F^{-1}_{X_a}(\rho_a) \tag{11}$$

With equation 11, *m* can be calculated for various distributions and conversion rates:

Table 3: Values of *m* for different Conversion Rates and Discrete Distributions (Part 1)

	Conversion Rate				
Probability Distribution of Delay	**100%**	**90%**	**80%**	**70%**	**60%**
Poisson($\lambda = 500$)	∞	529	519	512	506
Poisson($\lambda = 1000$)	∞	1041	1027	1016	1008
Geometric($p = 0.002$)	∞	1151	804	602	458
Geometric($p = 0.001$)	∞	2302	1609	1204	916
NegBin($r = 5, p = 0.01$)	∞	793	666	584	519
Uniform($0, 1000$)	1000	900	800	700	600

Table 4: Values of *m* for different Conversion Rates and Discrete Distributions (Part 2)

	Conversion Rate				
Probability Distribution of Delay	**50%**	**40%**	**30%**	**20%**	**10%**
Poisson($\lambda = 500$)	500	494	488	481	471
Poisson($\lambda = 1000$)	1000	992	983	973	960
Geometric($p = 0.002$)	347	256	179	112	53
Geometric($p = 0.001$)	693	511	357	224	106
NegBin($r = 5, p = 0.01$)	462	410	359	305	240
Uniform($0, 1000$)	500	400	300	200	100

Tables 3 and 4 show the corresponding time window sizes for different conversion rates and some example probability distributions. A conversion rate can result in vastly different time window sizes, depending on the shape of the probability distribution modelling delay. Since these distributions can differ for each arm, it is necessary to determine a time window size which ensures all arms' rewards are observed an adequate number of times.

Setting the global window size as the maximum of all arms' window sizes is a simple method to ensure that none of the arms have conversion rates lower than 70%. While delays are independent of rewards, a potentially interesting approach would be to dynamically adjust the time window size based on the rewards given by the arms and their delays. Specifically, an arm that consistently gives low rewards should not influence the time window size as much as one that gives high rewards.

6 Responsible Research

This section covers the ethical considerations of this study. Subsection 6.1 focuses on the exclusive use of synthetic data to avoid privacy concerns. Subsection 6.2 discusses the need for algorithm validation in clinical trials to ensure patient safety. Finally, Subsection 6.3 addresses the importance of bias mitigation in user-specific content recommendations.

6.1 Data sourcing

This study exclusively employs artificial data generated to evaluate the *OTFLinUCB* algorithm. No real-world data, particularly personal or sensitive information, was utilized at any stage. Consequently, no risks are associated with privacy infringement or ethical concerns related to data misuse. Synthetic datasets ensure that the findings are generalizable and that strict adherence to ethical research standards is maintained.

6.2 Impact on Clinical Trials

Using *OTFLinUCB* in clinical trials introduces important ethical considerations. The proposed method of calculating parameters for this algorithm can directly influence decision-making in this particular setting, thus influencing patient well-being. It is essential to thoroughly validate the algorithm's accuracy to minimize risks and ensure reliable trial outcomes.

6.3 Bias in Content Recommendation

The algorithms' use of user-specific data when recommending content to users can reinforce already-existing biases. While this paper focuses on a method for calculating an essential *hyperparameter* of *OTFLinUCB* in a purely theoretical setting, it is important to include bias mitigation strategies when adapting this algorithm for content recommendation.

7 Conclusion

Taking *OTFLinUCB* as a starting point, this paper studies the effect of delay on cumulative regret and what strategies can be employed to minimize this effect. It compares the algorithm mentioned above with *LinUCB*, *Exp3* and *UCB* and then finally reasons how the probability distributions that model delays can be used to estimate an appropriate window size that balances cumulative regret minimisation with memory usage minimisation.

Moreover, it introduces the metric ρ that can be used to calculate the window size and provides empirical evidence to support its use. Results presented in this paper suggest that choosing a conversion rate of 70% does not translate to a significant increase in cumulative regret compared to a conversion rate equal to or close to 100%, while also decreasing memory use. Using the *quantile function* of the probability distribution of delay, the window size can be directly calculated instead of estimated by trial and error.

Future work and considerations

- This paper assumes the probability distributions used to model delay are accurate. If that assumption is incorrect, window size calculation is also inaccurate and that can result in an unexpected increase of cumulative regret. It would be valuable to see how the uncertainty introduced by delay distribution estimation can be considered when calculating the time window size. Using real data could potentially provide new insights in this regard.
- An implementation decision was made to use a fixed α for estimating rewards instead of the more complicated formula from the standard implementation of *OTFLinUCB*. Because equation 8 depends on the window size, it is unclear how this design choice impacts window size calculation when the original calculation method for α is used instead of the alternative default value of 0.01.

References

[1] T. Lattimore and C. Szepesvári, "Bandit algorithms," in Cambridge University Press, 2020, pp. 102–116.

[2] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire, "The nonstochastic multiarmed bandit problem," *SIAM Journal on Computing*, vol. 32, no. 1, pp. 48–77, 2002. DOI: 10.1137/S0097539701398375.

[3] W. Chu, L. Li, L. Reyzin, and R. Schapire, "Contextual bandits with linear payoff functions," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, JMLR Workshop and Conference Proceedings, 2011, pp. 208–214.

[4] C. Vernade, A. Carpentier, T. Lattimore, G. Zappella, B. Ermis, and M. Brueckner, "Linear bandits with stochastic delayed feedback," in *International Conference on Machine Learning*, PMLR, 2020, pp. 9712–9721.

[5] O. Chapelle and L. Li, "An empirical evaluation of thompson sampling," in *Advances in Neural Information Processing Systems*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, Eds., vol. 24, Curran Associates, Inc., 2011. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2011/file/e53a0a2978c28872a4505bdb51db06dc-Paper.pdf.

[6] X. Shao and L. Li, "Data-driven multi-touch attribution models," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '11, San Diego, California, USA: Association for Computing Machinery, 2011, pp. 258–264, ISBN: 9781450308137. DOI: 10.1145/2020408.2020453. [Online]. Available: https://doi.org/10.1145/2020408.2020453.

[7] B. Dalessandro, C. Perlich, O. Stitelman, and F. Provost, "Causally motivated attribution for online advertising," in *Proceedings of the Sixth International Workshop on Data Mining for Online Advertising and Internet Economy*, ser. ADKDD '12, Beijing, China: Association for Computing Machinery, 2012, ISBN: 9781450315456. DOI: 10.1145/2351356.2351363. [Online]. Available: https://doi.org/10.1145/2351356.2351363.

[8] A. Gyorgy and P. Joulani, "Adapting to delays and data in adversarial multi-armed bandits," in *Proceedings of the 38th International Conference on Machine Learning*, M. Meila and T. Zhang, Eds., ser. Proceedings of Machine Learning Research, vol. 139, PMLR, 18–24 Jul 2021, pp. 3988–3997. [Online]. Available: https://proceedings.mlr.press/v139/gyorgy21a.html.

[9] B. Han and C. Arndt, "Budget allocation as a multi-agent system of contextual & continuous bandits," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, ser. KDD '21, Virtual Event, Singapore: Association for Computing Machinery, 2021, pp. 2937–2945, ISBN: 9781450383325. DOI: 10.1145/3447548.3467124. [Online]. Available: https://doi.org/10.1145/3447548.3467124.

[10] B. Howson, C. Pike-Burke, and S. Filippi, "Delayed feedback in generalised linear bandits revisited," in *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, F. Ruiz, J. Dy, and J.-W. van de Meent, Eds., ser. Proceedings of Machine Learning Research, vol. 206, PMLR, 25–27 Apr 2023, pp. 6095–6119. [Online]. Available: https://proceedings.mlr.press/v206/howson23b.html.

[11] P. Joulani, A. Gyorgy, and C. Szepesvari, "Online learning under delayed feedback," in *Proceedings of the 30th International Conference on Machine Learning*, S. Dasgupta and D. McAllester, Eds., ser. Proceedings of Machine Learning Research, vol. 28, Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 1453–1461. [Online]. Available: https://proceedings.mlr.press/v28/joulani13.html.

[12] L. Besson, *SMPyBandits: an Open-Source Research Framework for Single and Multi-Players Multi-Arms Bandits (MAB) Algorithms in Python*, Online at: GitHub.com/SMPyBandits/SMPyBandits, Code at https://github.com/SMPyBandits/SMPyBandits/, documentation at https://smpybandits.github.io/, 2018. [Online]. Available: https://github.com/SMPyBandits/SMPyBandits/.

[13] D. Arsene, C. M. Boon, M. Herrebout, W. Hu, and R. Owczarski, *Exploring Bandit Algorithms in User-Interactive Systems*, Online at: GitHub.com/thatCbean/SMPyBandits/tree/dragos-delayed-feedback, 2024. [Online]. Available: https://github.com/thatCbean/SMPyBandits/tree/dragos-delayed-feedback.

[14] O. Chapelle, "Modeling delayed feedback in display advertising," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '14, New York, New York, USA: Association for Computing Machinery, 2014, pp. 1097–1105, ISBN: 9781450329569. DOI: 10.1145/2623330.2623634. [Online]. Available: https://doi.org/10.1145/2623330.2623634.

[15] Z. Wang, C. Carrion, X. Lin, F. Ji, Y. Bao, and W. Yan, "Adaptive experimentation with delayed binary feedback," in *Proceedings of the ACM Web Conference 2022*, ser. WWW '22, ¡conf-loc¿, ¡city¿Virtual Event, Lyon¡/city¿, ¡country¿France¡/country¿, ¡/conf-loc¿: Association for Computing Machinery, 2022, pp. 2247–2255, ISBN: 9781450390965. DOI: 10.1145/3485447.3512097. [Online]. Available: https://doi.org/10.1145/3485447.3512097.

[16] T. Lancewicki, S. Segal, T. Koren, and Y. Mansour, "Stochastic multi-armed bandits with unrestricted delay distributions," in *Proceedings of the 38th International Conference on Machine Learning*, M. Meila and T. Zhang, Eds., ser. Proceedings of Machine Learning Research, vol. 139, PMLR, 18–24 Jul 2021, pp. 5969–5978. [Online]. Available: https://proceedings.mlr.press/v139/lancewicki21a.html.

[17] A. Verma, Z. Dai, and B. K. H. Low, "Bayesian optimization under stochastic delayed feedback," in *Proceedings of the 39th International Conference on Machine Learning*, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, Eds., ser. Proceedings of Machine Learning Research, vol. 162, PMLR, 17–23 Jul 2022, pp. 22 145–22 167. [Online]. Available: https://proceedings.mlr.press/v162/verma22a.html.

A Pseudocode of discussed algorithms

[H]

Input k and δ
for $t \in 1, \ldots, n$ **do**
  Choose action $A_t = \arg\max_i \mathrm{UCB}_i(t-1, \delta)$
  Observe reward X_t and update upper confidence bounds
end for

Algorithm 1: UCB

Parameters: Real $\gamma \in (0, 1]$.
Initialization: $w_i(1) = 1$ for $i = 1, \ldots, K$.
for each $t = 1, 2, \ldots$ **do**
1. Set

$$p_i(t) = (1 - \gamma)\frac{w_i(t)}{\sum_{j=1}^{K} w_j(t)} + \frac{\gamma}{K} \quad \text{for } i = 1, \ldots, K.$$

2. Draw i_t randomly according to the probabilities $p_1(t), \ldots, p_K(t)$.
3. Receive reward $x_{i_t}(t) \in [0, 1]$.
4. For $j = 1, \ldots, K$ set

$$\hat{x}_j(t) = \begin{cases} \frac{x_j(t)}{p_j(t)} & \text{if } j = i_t, \\ 0 & \text{otherwise,} \end{cases}$$

$$w_j(t + 1) = w_j(t) \exp\left(\frac{\gamma \hat{x}_j(t)}{K}\right).$$

end for

Algorithm 2: Exp3

Inputs: $\alpha \in \mathbb{R}^+$, K, $d \in \mathbb{N}$
$A \leftarrow I_d$ {The d-by-d identity matrix}
$b \leftarrow 0_d$
for $t = 1, 2, 3, \ldots, T$ **do**
$\theta_t \leftarrow A^{-1}b$
Observe K features, $x_{t,1}, x_{t,2}, \ldots, x_{t,K} \in \mathbb{R}^d$
for $a = 1, 2, \ldots, K$ **do**
$p_{t,a} \leftarrow \theta_t^\top x_{t,a} + \alpha\sqrt{x_{t,a}^\top A^{-1} x_{t,a}}$ {Computes upper confidence bound}
end for
Choose action $a_t = \arg\max_a p_{t,a}$ with ties broken arbitrarily
Observe payoff $r_t \in \{0, 1\}$
$A \leftarrow A + x_{t,a_t} x_{t,a_t}^\top$
$b \leftarrow b + x_{t,a_t} r_t$
end for

Algorithm 3: LinUCB: UCB with Linear Hypotheses

Input: Window parameter $m > 0$, confidence level $\delta > 0$ and $\lambda > 0$.
for $t = 2, \ldots, T$ **do**
Receive action set A_t
Compute width of confidence interval:

$$\alpha_{t,\delta} = 2f_{t,\delta} + \sum_{s=t-m}^{t-1} \|A_s\|_{V_t(\lambda)^{-1}}$$

Compute the least squares estimate $\hat{\theta}_W^t$ using (2)
Compute the optimistic action:

$$A_t = \arg\max_{a \in A_t} \left\langle a, \hat{\theta}_W^t \right\rangle + \alpha_{t,\delta} \|a\|_{V_t(\lambda)^{-1}}$$

Play A_t and receive observations
end for

Algorithm 4: OTFLinUCB

B LLM use

Most of the LLM use in this paper revolved around making paragraphs more readable and concise. One notable exception is the use of LLMs for the Responsible Research Section, where it was a starting point for the structure of each paragraph. Prompts such as "Write me a paragraph about privacy concerns when using artificial data", "Write me a paragraph about ethical concerns of using ML in clinical trials", or "Write me a paragraph about bias in recommendation systems" were used, and the results adapted to the subject at hand. Another notable exception is the abstract, which used a reworded summary of the paper provided by an LLM. Other than that, no particular paragraph or original ideas discussed here can be attributed to LLMs.