[AICP 2021] Final Report

❒ Team Information

Research Topic	Development of news article recommendation system via reinforcement learning. (Parallelization of bandit algorithms to reduce computational cost of news article recommendation systems.)			
Research Period	☒ Long-term ☐ Short-term			
Professor	**Gi-Soo Kim**			
TA				
Team Members		Student No.	Name	Major
	1	20192024	MD Khalequzzaman Chowdhry Sayem	MGE
	2	20192010	Mubarrat Tajoar Chowdhury	MGE
	3	20182010	Elkhan Ismayilzada	CS

❒ Abstract

Keywords	Contextual Bandit Algorithm, News Article Recommendation. Parallelized Bandit Algorithms

Impacts on performance of bandit algorithms in recommending news article when adapted for recommending to multiple users in parallel is studied in this work. The algorithms explored in this work are sequential LinUCB, sequential Thompson Sampling, parallelized Lazy LinUCB, parallelized Lazy Thompson Sampling, parallelized non-Lazy LinUCB, parallelized non-Lazy Thompson Sampling. The algorithms belong to a class of contextual bandit algorithms which in turn is a subclass of multiarmed bandit algorithms. Aim in this work is to explore whether parallelized implementation for faster performance at the expected sacrifice of rewards is worth using in a practical scenario – the scenario being news article recommendation to users. The dataset used for experimentation is Yahoo!

Front Page Today Module dataset R6A.

❒ Research Contents

1. Introduction

The algorithms implemented in this work are a class of contextual bandit algorithms which in turn are a subclass of multiarmed bandit problems. A multiarmed bandit problem refers to a scenario where a hypothetical agent has access to a limited set of choices, where each of the choices yield a reward unknown to the agent. The only way for the agent to get an estimate of the underlying reward distribution is to first try out a few choices and to observe the reward obtained for the corresponding choices. Different bandit algorithms offer different theoretical basis on which to estimate the long-term expected reward of each of the choices or arms based on the reward/trial history. The agent then uses the current estimate to choose arms to maximize the long-term reward gain from limited tryouts/trials at hand. In case of contextual bandit problems, the reward varies depending not just on the choices themselves but also on the situation under which the choices were made.

In the analogy of news article recommendation, the news articles are the “arms” a and the reward at time t (denoted as r_t) would be “1” if the reader(user) to whom it was recommended clicks the recommended article or “0” if the user does not click on the article. The situation or “context” in this scenario is denoted by a vector, denoted as $x_{t,a}$, which is derived from information about users(user features) to whom the article Is being recommended, and from information about the news(arm features). The derivation is illustrated in **Fig1.**

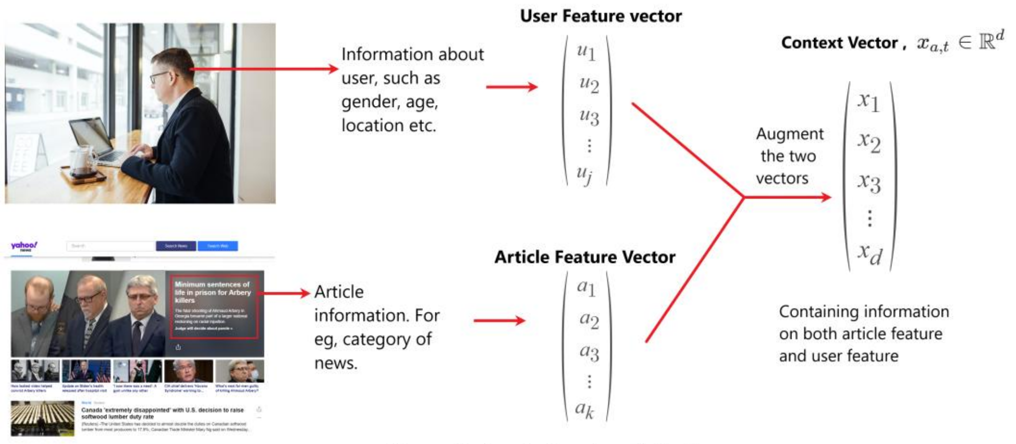
User Feature vector
Information about user, such as gender, age, location etc.
u_1 u_2 u_3 $\vdots$ u_j
Context Vector , $x_{a,t} \in \mathbb{R}^d$
Augment the two vectors
x_1 x_2 x_3 $\vdots$ x_d
Article Feature Vector
Article information. For eg, category of news.
a_1 a_2 a_3 $\vdots$ a_k
Containing information on both article feature and user feature

Figure 1. *Context vector derivation.*

In the algorithms to be discussed here, The expected reward function for all the algorithms given the context vector is assumed to be linear. Expressed as: $E[r_t|x_{t,a}] = x_{t,a}^T\theta$ where θ is iteratively estimated as $\hat{\theta}$ at each time step after the reward is observed.

In this work, the algorithms explored are sequential LinUCB[1], parallel Lazy LinUCB[3], parallel Non-Lazy LinUCB[3], sequential Thompson Sampling[2], Lazy Thompson Sampling[3] and Non-Lazy Thompson sampling[3].

To help with visualization of multitude of users being recommended articles by the algorithm and the limit in spread of information with which the recommendation decisions are made, the concept of a news article "server" or a "worker" is introduced. In sequential versions of LinUCB[1] and Thompson Sampling[2], one user is served at a particular time step. In parallel versions of the algorithms, which are modified from [3], there are P workers serving news articles in parallel, attending to batches of P users from queue at a particular time step. This distinction is illustrated in **figure 2** and **figure 3.**

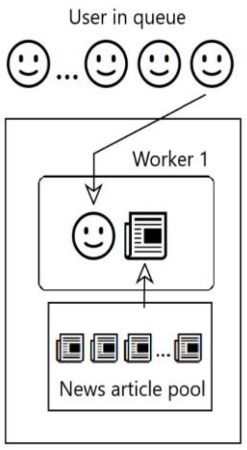

Figure 2. *Workers in sequential bandit algorithms*

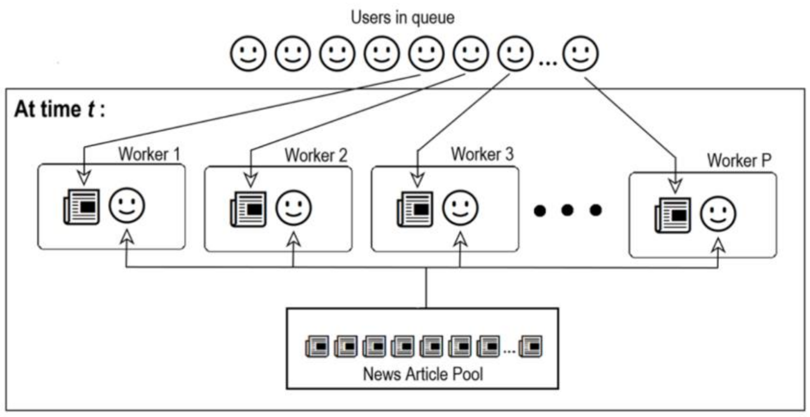

Figure 3. *Workers in parallel bandit algorithms*

The following discussion are about steps in algorithm elaborated in "Research Methods" Section.

Sequential LinUCB calculates the vector b_t by iterative update $b_t \leftarrow b_{t-1} + x_{t,a_t} r_t$. And then at time $t+1$, $\theta_{t+1} \leftarrow V_{t+1}^{-1} b_t$ is used to estimate max expected reward for each arm as : $\theta_{t+1}^T x_{t+1,a} + \alpha \sqrt{x_{t+1,a}^T V_{t+1}^{-1} x_{t+1,a}}$ where α is a hyperparameter that encourages higher selection frequency for underexplored arms. The arm/article with the highest reward estimate is then chosen to be recommended.

In case of Thompson Sampling algorithm, all parameter updates are the same except at the two following points. While $\widehat{\theta_{t+1}} \leftarrow V_{t+1}^{-1} b_t$, the estimate is then used as mean for the normal distribution $\mathrm{N}\left(\widehat{\theta_{t+1}}, \nu^2 V_{t+1}^{-1}\right)$, which is then sampled from to get an estimate for θ to be then used to compute maximum reward estimate for each arm as $\widehat{\theta_{t+1}}^T x_{t+1,a}$before an arm is selected. Upon observing the rewards and updating all the relevant parameters , the worker receives the next user in queue for the next iteration

The sequential algorithms keep a pseudocount of features encountered in previous time steps using a covariance matrix V_t, whose value at time t is computed as:
$V_t \leftarrow \lambda I_d + \sum_{t=1}^{t-1} x_{t,a_t}^T x_{t,a_t}$ (λ being a regularization term) or iteratively as $V_t \leftarrow V_{t-1} + x_{t,a_t}^T x_{t,a_t}$. The sole worker in sequential algorithms uses the entire history of encountered context vectors through the covariance matrix and also the pseudocount of rewards for corresponding vectors via a parameter b_t (computed as $b_t \leftarrow b_{t-1} + x_{t,a_t} r_t$) when choosing a news article for a particular user.
The parallelized version of covariance matrix $\mathrm{V}_{t,p}$ for Lazy and Non-Lazy algorithm for both Thompson sampling and LinUCB are to be used for calculation of $\widehat{\theta_t}$ at time step t are calculated as follows:
For Lazy, at time t, the cumulative value of covariance, iteratively calculated for each p in order, from the start of the experiment stands at $\mathrm{V_{t,p}} = \lambda \mathrm{I_d} + \sum_{\mathrm{e=1}}^{\mathrm{t\text{-}1}} \sum_{\mathrm{f=1}}^{\mathrm{P}} \mathrm{x_{e,f,a}}\, \mathrm{x_{e,f}^T} + \sum_{\mathrm{k=1}}^{\mathrm{p-1}} \mathrm{x_{t,k,a}}\, \mathrm{x_{t,k,a}^T}$
For Non-Lazy at time t, each of the p workers use the same covariance matrix, the cumulative value of covariance from the start of the experiment stands at: $\mathrm{V_{t,1}} = \lambda \mathrm{I_d} + \sum_{\mathrm{e=1}}^{\mathrm{t\text{-}1}} \sum_{\mathrm{f=1}}^{\mathrm{P}} \mathrm{x_{e,f,a}}\, \mathrm{x_{e,f}^T}$
In covariance matrix for Lazy the additional term of sum outer product of context vector for corresponding selected arm by preceding workers leads to more variable selection of arms within the batch, more specifically, this additional summation induces higher selection of articles which were underexplored given the context. This selection of more suboptimal articles(articles which do not have highest expected reward given the context at that particular time step) earlier in the experiment leads to higher reward in the long term as articles with better expected reward, given the context could be discovered.
In Non-Lazy Parallel algorithms, at time step t, the entire batch of P workers use the covariance matrix and the pseudo reward count updated at the end of previous time steps, meaning the workers are unaware of the encountered context vector information(via covariance matrix), arm selection and corresponding reward for preceding workers in the same batch.
This “unawareness” leading to reduced number of computations within the same batch increases serving speed, but the reward is expected to be less because of less information being used during decision making.
In Lazy Parallel algorithms, the workers in the same batch are aware of the pseudocount of context features encountered from t=1 up to preceding workers in the same batch via the covariance matrix $V_{t,p}$ but are unaware of the corresponding pseudo feature-reward

count that comes with the parameter b_t. The relative higher extent of information accessibility of preceding workers within the batch leads one to expect higher reward in Lazy compared to Non-Lazy and hence slower speed due to increased computation involved within a batch.

The specific method of how covariance matrix is used in each of the algorithms to make decision is shown in the sections that follow.

2. Research Purpose

Purpose of this research has been to find out the extent to which parallelized modification of contextual bandit algorithms impact the rewards obtained from recommending news articles to users. The reason parallelized implementation of contextual bandit algorithms is desirable, despite the hypothesized superiority of sequential algorithms in terms of reward gain, is because of the speed gain and the ability to serve a multitude of users at a single time step. While in-depth analysis of the steps in each of the algorithms reveal the impact on reward and speed to be expected, the extent to which the expectations might hold in a practical scenario is investigated using a real-world news article recommendation dataset

3. Research Methods

Experiment Procedure

Dataset used: R6A - Yahoo! Front Page Today Module User Click Log Dataset, version 1.0

Processor Used: Intel Core i7-10700KF 5.1 GHz

Dataset contents:

6 user features, 6 Article features, ID of the article being displayed to the user observed reward for the article shown, pool of article ID's at time t from which article to be displayed is chosen.

Performance measure: $Click\_Through\_Rate(CTR) = \frac{Total\_Cumulative\_Clicks}{Total\_Cumulative\_Users\_Served}$

Hyperparameters and parameters are first initialized. At each iteration, user features and article features are used to form the context vector. If the selected article in experiment matches the selected arm in mentioned in the dataset, the rewards for that context is observed.

Taking into account the original papers [1],[2],[3] where the algorithms were first published, the following modified approach to implementing the algorithms were taken:

Algorithm: Non-Lazy Thompson Sampling and Non-Lazy LinUCB

```
o $V_{t,p} = \lambda I_d$ , $b_t = 0_d$, set algorithm {LinUCB ,Thompson Sampling}, set $\alpha$
o For t = 1,2,3,…,T do
    ▪ $\widehat{\theta}_t \leftarrow V_t^{-1} b_{t-1}$
    ▪ For p = 1,2,3…P
        • For articles $a$ = 1,2,3…K do
            o If algorithm is LinUCB :
                ▪ Reward expectation for $a = \widehat{\theta}_t^T x_{t,p,a} + \alpha\sqrt{x_{t,p,a}^T V_{t,1}^{-1} x_{t,p,a}}$
            o If algorithm is Thompson Sampling
                ▪ Reassign $\widehat{\theta}_t \leftarrow Sample\ from\ N(\widehat{\theta}_t, \nu^2 V_{t,1}^{-1})$
                ▪ Reward expectation for $a = \widehat{\theta}_t^T x_{t,p,a}$
        • End the loop for articles
        • For each p Choose the article with highest reward expectation
        • Observe the reward∈{clicked or not} for all workers
    ▪ End the loop for p
    ▪ Compute: $V_{t+1,1} = V_{t,1} + \sum_{p=1}^{P} x_{t,p,a_t}^T x_{t,p,a_t}$
    ▪ Compute: $b_t = b_{t-1} + \sum_{p=1}^{P} x_{t,p,a_t} r_{t,p}$
    ▪ Compute Click-Through-Rate: $CTR_t = \frac{\{(CTR_{t-1} * (t-1) * P + (\sum_{p=1}^{P}\{r_{t,p}\}))\}}{P * t}$
o End loop for time t
```

Algorithm: Lazy Thompson Sampling and Lazy LinUCB

```
o $V_{t,p} = \lambda I_d$ , $b_t = 0_d$, set algorithm {LinUCB ,Thompson Sampling}, set $\alpha$
o For t = 1,2,3,…,T do
    ▪ $\widehat{\theta}_t \leftarrow V_{t,1}^{-1} b_{t-1}$
    ▪ For p = 1,2,3…P
        • Compute $V_{t,p} = V_{t-1,1} + \sum_{k=1}^{p-1} x_{t,k,a_t}^T x_{t,k,a_t}$
        • For articles $a$ = 1,2,3…K do
            o If algorithm is LinUCB :
                ▪ Reward expectation for $a = \widehat{\theta}_t^T x_{t,p,a} + \alpha\sqrt{x_{t,p,a}^T V_{t,p-1}^{-1} x_{t,p,a}}$
            o If algorithm is Thompson Sampling
                ▪ Reassign $\widehat{\theta}_t \leftarrow Sample\ from\ N(\widehat{\theta}_t, \nu^2 V_{p-1,1}^{-1})$
                ▪ Reward expectation for $a = \widehat{\theta_t^T} x_{t+1,p,a}$
        • End the loop for articles
        • For each p Choose the article with highest reward expectation
        • Observe the reward∈{clicked or not} for all workers
    ▪ End the loop for p
    ▪ Compute: $V_{t+1,1} = V_{t,1} + \sum_{p=1}^{P} x_{t,p,a_t}^T x_{t,p,a_t}$
    ▪ Compute: $\mathrm{b_t} = \mathrm{b_{t-1}} + \sum_{p=1}^{P} x_{t,p,a_t} r_{t,p}$
    ▪ Compute Click-Through-Rate: $CT\mathrm{R_t} = \frac{\{(\mathrm{CTR_{t-1}*(t-1)*P+(\sum_{p=1}^{P}\{r_{t,p}\})})\}}{\mathrm{P*t}}$
o End loop for time t
```

While standalone algorithms for sequential algorithms exist, and were at the initial phase of the project, implemented as such, the sequential algorithms turn out to be a special case of the parallelized algorithms when the number of workers, p, is set as 1. So, the results for sequential algorithms presented here were obtained by setting p = 1 for parallelized algorithms, similar to how it's illustrated in [3]

For efficient computation of inverse of covariance matrix, the following derivation, using the work of Kennet S Miller[4], was used:

$$V_{t,1}{}^{-1} = \left(V_{t-1,1} + \sum_{p=1}^{P} x_{t,p,a_t}^T x_{t,p,a_t} \right)^{-1} = V_{t-1,1}{}^{-1} - \frac{1}{1+g} V_{t-1,1}{}^{-1} x_{t,p,a_t}^T x_{t,p,a_t} V_{t-1,1}{}^{-1}$$

Where $\mathrm{g} = \mathrm{trace}(x_{t,p,a_t}^T x_{t,p,a_t} V_{t-1,1}{}^{-1})$.

This computation allows us to avoid an expensive inverse calculation after every update of covariance matrix. Instead the inverse calculated in previous time step cou ld be used with the outer product of context vectors to find the inverse of updated matrix.

4. Research Schedule

Month	Task Completion
March	1. Introducing team members with professor 2. Read the paper titled "Contextual Bandits with Linear Payoff Functions" 3. Read the paper titled "A Contextual-Bandit Approach to Personalized News Article Recommendation"

April		1. Found two books to have better understanding of both theoretical & practical aspects. 2. Implementation of epsilon-Greedy Algorithm (One of the bandit algorithms) in python 3. 3. Configured environment for reading data to simulate (R6 dataset of Yahoo)
May		1. Finished reading the book Bandit Algorithm for website optimization by John Myles White 2. Simulation of LinUCB Disjoint on toy dataset
June		1. Implementation of Thompson Sampling Method 2. AICP Interim Presentation
July		1. Implementation of LinUCB(with common reward weights) 2. Implementation of Thompson sampling Algorithm in Yahoo R6 dataset 3. Implementation of LinUCB on Yahoo R6 Dataset 4. Understanding the posterior distribution derivation of reward weights in Thompson sampling algorithm. 5. Figuring out the appropriate way to derive unique context vector for each arm 6. Analyzing and understanding the specific contributions and impact of the parameters in the working of the algorithm
August		1. Addressing the issue of underuse of available cores in processor to run code that ultimately led to extremely long runtime. 2. Discussing methods to improve the low cumulative reward from both Thompson sampling and linUCB. 3. Evaluating implementation of all algorithms using Numba (an open-source JIT compiler that translates a subset of Python and NumPy into fast machine code using LLVM, via the llvmlite Python package.) to speedup runtime 4. After numerous implementations, fixating on use of Linux OS to speedup runtime by 12x and discarding use of Numba & intel math library(MKL). 5. Evaluating implementation of Thompson sampling through use of increasing standard deviation in posterior distribution of weights with an aim to increase cumulative rewards and CTR. 6. Discussing contents of the paper “Parallelizing Contextual Linear Bandits” 7. Discussing methods of implementing the four algorithms mentioned within the paper 8. Evaluating implementation of Cholesky Sampling while sampling weights of linear reward function while implementing Thompson Sampling 9. Finding a specific way to initiate the algorithms and understand the significance of doubling rounds and exploring ways to implement doubling round routine

September	1. Implementations complete except for a minor error in the methods used to count doubling rounds. Solution to which has been found and to be implemented soon. 2. GPU-based implementation of the algorithm using CUDA completed, hence reaching the conclusion that CPU based methods is more convenient (due to the latency of transferring data to vram from ram) unless the entire structure of the code is to be changed. 3. graphs representing CTR as the experiment progresses produced.
October	1. Fixating on methods derived from Gershgorin circle theorem to detect doubling rounds. (not added into final report as it does not include any real world insight while serving news to the user except to prove some theoretical bounds) 2. Reading the paper "META-LEARNING FOR CONTEXTUAL BANDIT EXPLORATION" too look for new research direction.
November	1. Found a new direction of research for a novel algorithm which is expected to be implemented soon. 2. Run experiments for the graphs of final report & poster. 3. Report & Poster writing.

5. Research Results

While plenty of graphs were generated throughout the duration of project to help lead the research. Few of the results, after reaching the final refined stages of the project have been presented below:

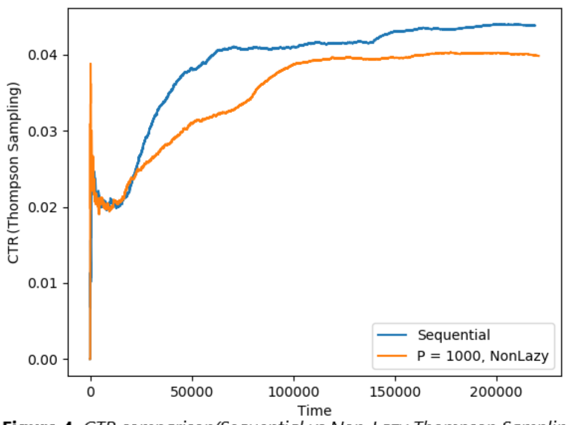

Figure 4. *CTR comparison(Sequential vs Non-Lazy Thompson Sampling)*

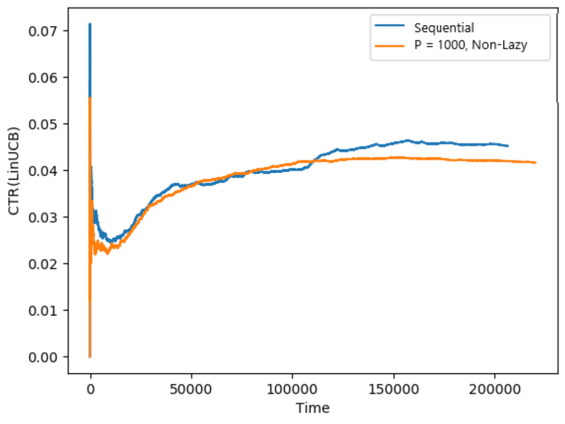

Figure 5. *CTR comparison (Sequential vs Non-Lazy LinUCB)*

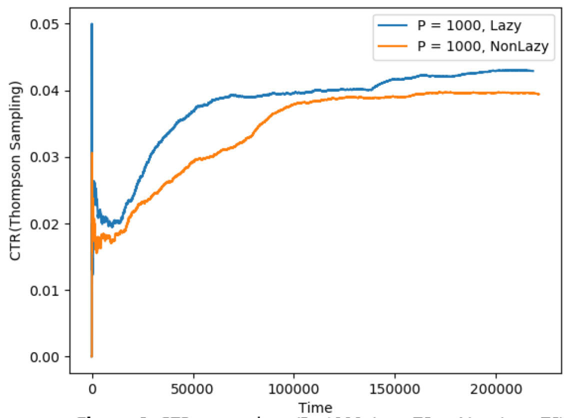

Figure 6. *CTR comparison (P=1000, Lazy TS vs Non-Lazy TS)*

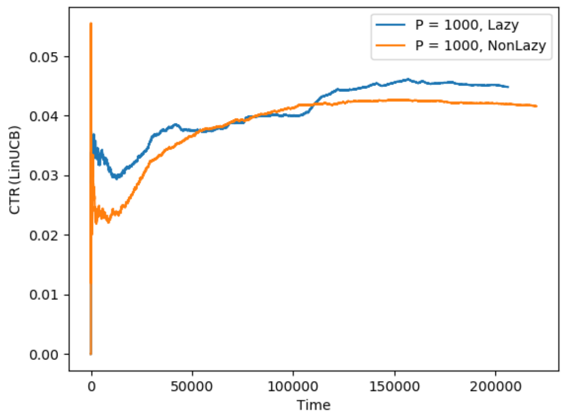

Figure 7. *CTR comparison (P=1000, Lazy LinUCB vs Non-Lazy LinUCB)*

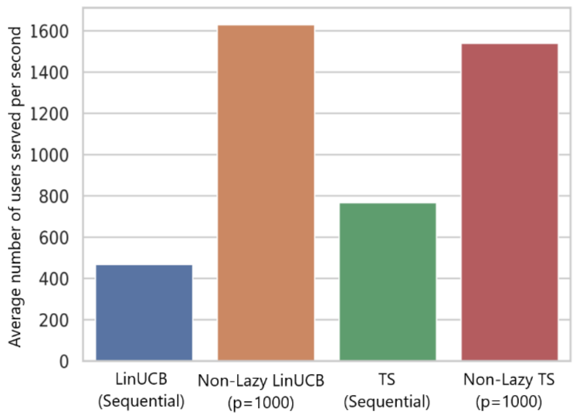

Figure 8. *Speed comparison (Sequential vs Non Lazy Algorithms)*

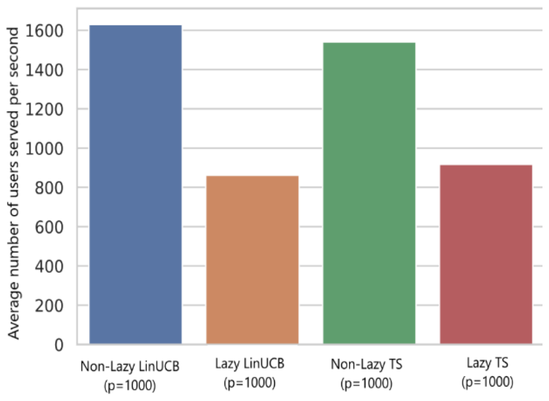

Figure 9. Speed Comparison (*Lazy vs Non-Lazy Algorithms)*

In figure 4 to 6 above, the plots show curves flattening around constant values of CTR from last third of the experiment. This convergence is indicative of the algorithms learning the true reward of the news articles. The initial dip is indicative of the algorithms choosing suboptimal arms in order to learn more about expected reward of news articles in pool given the particular user features.

Non Lazy Parallel vs Sequential algorithms

Since covariance matrix is not updated within the batch a particular time step t, the expectation is that Non-Lazy Parallel will suffer loss in reward compared to their sequential counterpart. The convergence points in **Figure 4** and **Figure 5** confirm this, however the span of difference in rewards is smaller than 0.01 in CTR. However, the average number of users served per second is 3.48 times higher for Non-Lazy Parallel in LinUCB and around 2.00 times higher in Non-Lazy Parallel Thompson Sampling as illustrated in **Figure 8**.

Lazy Parallel vs Non-Lazy Parallel algorithms

The covariance calculation formula, $V_{t,p} = \lambda I_d + \sum_{e=1}^{t\text{-}1} \sum_{f=1}^{P} x_{e,f,a}\, x_{e,f}^{T} + \sum_{k=1}^{p-1} x_{t,k,a}\, x_{t,k,a}^{T}$ for

Lazy algorithms differ from the Non-lazy in that the latter’s covariance formula does include the last summation term. Therefore, when experiment has progressed long enough so that T*P >> P, the difference in the value of covariance matrix in both algorithms become negligible and hence under scenarios that involve running the experiment for scenarios where T>>P, theoretical implications would support the notion of sacrificing reward via use of non-Lazy algorithms over Lazy ones to exploit the significantly higher runtime speed. In situations where this scenario does not arise, for covariance related issues discussed before, the resulting less information access of Non-Lazy algorithm is expected lead to loss in rewards.

As expected, the there is loss in reward for Non-Lazy algorithms, though still the span of difference of converged CTR value is less than 0.01. as can be seen in **Figure 6** and **Figure 7.** The speed, as shown in **Figure 9**,for Lazy LinUCB algorithm is 1.89 times faster and Lazy-Thompson Sampling is around 1.68 times faster. This is due to the increased frequency of covariance computation required within a batch.

6. Research Achievements

- With the aid of published research work, successful implementation of parallelized modification of the sequential algorithms and following successful tested on a real-world news article recommendation dataset that confirms hypothesis implied by the algorithm steps.
- A new hybrid algorithm currently being studied by the team, which has been possible because of the successful implementations in this work.

7. Expected Contribution / Future Plans

Parallelization evidently leads to a significant gain in speed at the expense of small loss in reward. However, the ability for a recommendation service to host a vast number of users in a short time is in the interest of service providers, therefore, the small loss in reward is worth sacrificing when a vast number of users require to be served.

This work is expected to give readers some intuition regarding the extent of speed gain that parallelization can offer at the expense of small decrease in reward. The specific steps within the algorithm that lead to such behavior has been pointed out and discussed and their extent of impact demonstrated using a real-world dataset. Another issue that arises when deploying the algorithms in a practical scenario is deciding which algorithm to use? The future plan of study, which we are working on as we speak, seeks to solve that problem using a hybrid algorithm that aim to learn which versions of algorithms at hand might be suitable given a particular user context.

8. Member's Role

	Name	Role	Details
1	Sayem	Leader	Scheduling meeting, fixing deadlines. Literature review through reading research papers and code implementation. Searched for and suggested to read and online videos to watch
2	Mubarrat	Data Analyst	Literature review through reading research papers and code implementation. Searched for and suggested to read and online videos to watch
3	Elkhan	Lead Programmer	Literature review through reading research papers and code implementation. Searched for and suggested to read and online videos to watch

9. References

1. Chu, Wei, et al. "Contextual bandits with linear payoff functions." Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics. JMLR Workshop and Conference Proceedings, 2011.

2. Agrawal, Shipra, and Navin Goyal. "Thompson sampling for contextual bandits with linear payoffs." International Conference on Machine Learning. PMLR, 2013.

3. Chan, Jeffrey, et al. "Parallelizing Contextual Linear Bandits." *arXiv preprint arXiv:2105.10590* (2021).

4. Miller, Kenneth S. "On the Inverse of the Sum of Matrices." Mathematics Magazine, vol. 54, no. 2, Mathematical Association of America, 1981, pp. 67–72, https://doi.org/10.2307/2690437.

10. Participants' Comments

(Professor) The AIStars team conducted the project successfully. All the team members

were hardworking throughout the year, with constant weekly meetings during the vacations.
The team kept trying to introduce novel ideas to improve the CTR of the news articles and also to reduce the computational cost of the algorithms.
The team showed improvement every other week. The final result is a first try of parallel bandit algorithms on the Yahoo! news article recommendation dataset. The high CTR and computation speed of the parallel algorithms show their practicability in real recommendation systems
(Student members)
- (Mubarrat) This has been a great experience. I learned many new things. Thanks to our professor, I could ease into the otherwise unfamiliar topic that was bandit algorithms.
- (Sayem) Bandit was totally an unknown arena of AI for me before joining AICP. The journey of a year has helped me to build somewhat ground to be confident about doing research on new algorithms, thanks to the professor! Also appreciate AICP, for making such a wonderful opportunity available for us.
- (Elkhan) Throughout this project, I got introduced to the basics of reinforcement learning and I learned how to implement these in python with theory in mind. I realized the importance of contextual bandits in real-life scenario since it is used by many famous companies around the globe.

We hereby submit the final report for the AICP in 2021 as stated above.

Date:	11 / 29 / 2021	
Professor:	Gi-Soo Kim	Gi-Soo (Signature)
Student leader:	MD Khalequzzaman Chowdhury Sayem	Sayem (Signature)