AI-Assisted Personalized Calendars: Enhancing Semester Planning for Students

Szeyi Chan*, **Kaiqi Zhang***

Northeastern University

Abstract

College students frequently face challenges managing their coursework effectively due to various academic obligations, including assignments, readings, exams, and final projects. This study introduces a personalized calendar organization tool to enhance student scheduling by learning from human preferences. In this study, we compared two reinforcement learning algorithms, Linear Upper Confidence Bound and Preference Perceptron, to learn from and adapt to user feedback. Our approach models the problem as a contextual bandit framework and we evaluate the algorithm based on the regret minimization speed, a metric indicative of learning efficiency and user satisfaction. The results demonstrated that the Preference Perceptron algorithm outperformed the Linear UCB algorithm, as evidenced by a faster decay in expected regret values. This suggests that the Preference Perceptron algorithm can more effectively adapt to user preferences, offering significant implications for developing advanced, personalized planning tools for educational contexts.

Introduction

The effective management of academic responsibilities poses a significant challenge for college students, who must navigate a complex array of coursework tasks, including assignments, readings, exams, and final projects. While various digital and non-digital planning tools are available, these tools often rely mainly on user manual inputs. Additionally, most of these tools use static algorithms that fail to learn from user interactions, making them less effective over time as the algorithm cannot adapt to the user's evolving needs. The lack of personalization and adaptability in these tools highlights a significant gap in helping students manage academic tasks and adapt to individual learning preferences.

This study aims to address existing limitations by comparing algorithms tailored for a personalized calendar organization tool designed to optimize student scheduling. We adopt a dynamic approach that adapts to individual preferences through learning. By conceptualizing our problem within the contextual bandit framework, we enable the dynamic integration of user-specific data and preferences to predict optimal calendar options accurately. Specifically, we investigate the efficacy of two reinforcement learning algorithms, Linear Upper Confidence Bound (Linear UCB) and Preference Perceptron (PP), implemented within this framework. This framework allows the tool to provide personalized suggestions by learning from real-time user feedback.

Our comparative analysis focuses on the performance of these algorithms based on the speed of regret minimization—a metric that measures the efficiency of learning algorithms through their ability to minimize mistakes over time. This metric correlates with user satisfaction and the overall effectiveness of the scheduling tool. The two algorithms utilize different user feedback mechanisms in our study. The Linear UCB algorithm employs a noisy score rating system, while the PP algorithm uses improved actions provided by the user. These differences in feedback mechanisms lead to the different results of our study. The PP algorithm outperformed the Linear UCB algorithm, demonstrating a faster decay in regret values. This suggests that the PP algorithm is more capable of rapidly adapting to user preferences than the Linear UCB algorithm, which can be used to develop more refined and efficient personalized planning tools for students. The contribution of this study also shows the possibility of developing more refined and efficient personalized planning tools to cater to the unique needs of individual students in the future.

In this paper, we start by discussing the background of the multi-armed bandit problem. We then review existing approaches in related work and proceed to define our problem formulation within the contextual bandit framework. Our experimental approach includes a comparative analysis of two algorithms. The source code for experiments are publicly available[1]. We then present our findings from the regret analysis of both algorithms. Finally, we identify potential areas for future research.

Background

In this section, we briefly review the background of the multi-armed bandit problem. We also discuss the applicability and limitations of different multi-armed bandit problems.

[1] https://github.com/SzeyiReinaChan/PersonalizedCalendar

Multi-Armed Bandit Problem (MAB)

In the Multi-Armed Bandit Problem, each round $t \in T$ involves the algorithm choosing one of K possible actions, or "arms", with the algorithm collecting a reward only for the selected action, without knowledge of the potential rewards from the others (Slivkins et al. 2019). Furthermore, the contextual bandit problem extends this model by incorporating additional data—commonly referred to as context data or features—that help decide which arm to pull and predict the action that can receive the most reward (Agarwal et al. 2014; Li et al. 2010). This approach is often used in systems that provide personalized recommendations or targeted advertising to enhance the decision-making process, where the algorithm can adapt to the individual characteristics of each user as "context" during the process (Li et al. 2010).

In the contextual bandit problem, the algorithm actively collects user feedback, for example, through clicking online ads. Coactive learning offers an alternative method for gathering feedback. Instead of receiving a simple binary reward (e.g., clicked or not clicked), the coactive learning process involves two steps: the algorithm presents an initial action and then collects an improved or modified response from the user. This approach allows users to actively participate in refining the outcome, thereby helping to co-construct a more effective response (Shivaswamy and Joachims 2015).

The linear contextual bandit problem calculates rewards through a linear relationship with the context-action pair, where contextual information is still affected in decision-making process. Unlike the general contextual bandit problem, which does not assume a linear relationship between context and action, the linear model simplifies the analysis of the algorithm's performance. Consequently, the linear contextual bandit model requires less data to provide reliable estimates, making the analysis process more straightforward (Abe, Biermann, and Long 2003).

Related Work

Improving calendar organization through learning from human preferences has been an active area of research. Previous studies such as Myers et al. (2007), Das Swain et al. (2023), and Zaidenberg and Reignier (2011) have explored various approaches to adapt calendar systems based on user feedback and preferences. Oh and Smith (2005) further develop this by proposing hybrid frameworks that combine multiple learning strategies to enhance prediction accuracy and user satisfaction.

Bayesian models were used in Mynatt and Tullio (2001) with probabilistic methods to model user attendance and schedule preferences uncertainties. These models are good at handling the uncertainties and variabilities in human behavior. However, these models were not used in our study due to the computational demands and the necessity for extensive historical data to form accurate prior distributions. Our study's focus on real-time learning and adaptation from limited user interactions made Bayesian approaches less suitable compared to our study's more straightforward and computationally efficient linear models.

Mitchell et al. (1994) demonstrates the application of decision trees in adapting to user preferences over time. Decision trees provide an interpretable framework for decision-making. Although this decision tree can model more complex behaviors than linear methods, it struggles to adapt to user preferences in real-time compare to our study. Following Gervasio et al. (2005), our study adopts a linear preference model, which assumes the reward structure as a linear combination of the observed features and the user's unknown preference vector. This model is particularly suited for contexts where decisions need to be updated dynamically for personalized scheduling applications.

Furthermore, Q-learning, a form of model-free reinforcement learning, involves learning an action-value function that gives the expected utility of taking a given action in a given state and following a fixed policy thereafter. However, Q-learning is not suitable for our study because the context does not depend on historical data, and thus, lacks the dynamic mechanisms required in our case.

Finally, we decided not to classify our problem as an MAB problem because MABs typically do not consider the context or features associated with each action. In contrast, our study requires a model incorporating user-specific data, such as individual calendars, and preferences to predict the most suitable calendar options. Thus, the contextual bandit framework is a more appropriate choice due to its dynamic ability to handle such features.

Project Description

Problem Fomulation

In this study, our problem of the calendar scheduling system was addressed within the context of a multi-armed bandit problem that incorporates contextual information. This calendar, denoted as set C with E events. These events are categorized into two types: irrelevant events (e_i) and relevant events (e_r). Irrelevant events are those pre-existing engagements or appointments already scheduled on the calendar. In contrast, relevant events are the activities related to schoolwork that need to be scheduled, such as homework or reading assignments. Each e_i include constraints reflect user predefined preference. Open slots on the calendar are denoted as -1. Example of a calendar for each user (u_i) can be expressed as follows: $C_u = \{e_{i1}, e_{i2}, e_{r1}, e_{r2}, -1, -1...\}$.

The state (s) of the system, also referred to as the context, is represented by the calendar that includes only the irrelevant events at any given time, denoted as $s = \{e_{i1}, e_{i2}, -1, -1, -1, -1...\}$. An action ($a$) within this system is defined as the process of adding all relevant events to the calendar. In this dynamic environment, the set of feasible actions is determined by the current state or context, denoted as s. Specifically, an action is considered eligible if it fills in slots currently unassigned (i.e., slots marked with "-1") and adheres to the constraints imposed by the revalent events. Consequently, the set of available actions a_t at each round t is contingent upon the context s_t at that time. This implies that the action set may vary at each round, reflecting the adaptive nature of the decision-making process of contextual MAB.

Each pair of state and action, (s, a), is associated with

a feature vector through a mapping function f, where $f(s,a) \in R^d$. This feature vector provided a multi-dimensional representation of the state-action pair that captures essential characteristics relevant to scheduling. In practical application, we adapted the features outlined by Gervasio et al. to fit our problem context. We employed binary features, with values corresponding directly to the presence or absence within event e_r—assigned as 1 if the feature is present in e_r, and 0 otherwise.

The preferences of human users towards these features are quantitatively expressed through a vector $\theta \in R^d$. Each component of this vector signifies the degree of user preference for the corresponding feature in the vector $f(s,a)$.

In our study, we simulate user feedback in two methods. The first method is through a noisy score rating system. Specifically, we model human feedback as distributed according to a Gaussian distribution, where the Gaussian noise model captures the variability and imperfections typically present in human-provided scores. The mean of this distribution, and thus the expected value of the human feedback, is given by the linear predictor, expressed as $E[humanfeedback] = x^\top \theta$, where $x = f(s,a)$. In here, x represents the feature vector associated with the user interaction, and θ represents the parameter vector that characterizes the typical user response. The second method of feedback simulation involves the assumption of improved actions. In our study, we assume that the reward$(s,a) = f(s,a)\dot{\theta}$ for the improved actions ($\bar{a} \in A$) is greater than or equal to the action chosen by the algorithm a_t, formally denoted as $\bar{a}^\top \theta \geq a_t^\top \theta$.

The reward function (r) is formulated as $r(s,a) = \theta \cdot f(s,a)$, where it computes a scalar value indicative of the overall user satisfaction with the action taken in a given state. This reward function effectively evaluates how well the addition of relevant events aligns with human preferences as quantified by θ.

As mentioned, our problem is modeled as a contextual multi-armed bandit problem without state transition. In practice, the state s is randomly generated in each round t, independent of previous states or actions to simplify the history independency in sequential decision-making problems. This modeling approach allows for the focus to be on selecting the optimal action (i.e., the scheduling of relevant events) based on the current state and maximizing the expected total reward, which, in this scenario, translates to maximizing user satisfaction.

In practice, our goal is to achieve the maximum possible total reward over T rounds. Formally expressed as $E\left[\sum_{t=1}^{T} r_{t,a_t}\right]$, where a_t represents the action that maximizes the expected payoff at each round t. Equivalently, this can be viewed as minimizing the regret associated with not always being able to choose the optimal action due to uncertainty or incomplete information. Total regret is formally defined as the difference between the reward accrued by always choosing the optimal action and the actual reward accrued by the algorithm. The cumulative regret over T rounds is defined by:

$$REG_T = E\left[\sum_{t=1}^{T} r(s_t, a_t^*) - r(s_t, a_t)\right] \quad (1)$$

where a_t is the action selected by the algorithm at round t, and a_t^* is the optimal action that would have yielded the maximum expected reward at round t. Our goal is to minimize total regret, which effectively maximizes total payoff.

Overall, the formulation of our problem sets the groundwork for developing a calendar scheduling system that can dynamically and efficiently add relevant events to a calendar while considering user preferences and pre-existing commitments to enhance overall satisfaction and usability.

Algorithms

In this study, we applied the Linear UCB algorithm and the PP method to learn users' preferences based on their feedback. We will first detail the Linear UCB algorithm, followed by a description of the PP approach utilized in our research.

Linear UCB

Algorithm 1: LinUCB With Contextual Bandits

1: Input: $\alpha \in R^+$
2: **for** $t = 1, 2, \ldots, T$ **do**
3: Observe features of all action $a \in \mathcal{A}_t : x_{t,a} \in R^d$
4: **for all** $a \in \mathcal{A}_t$ **do**
5: **if** a is new then **then**
6: $A_a \leftarrow \lambda I_d$ (d-dimensional identity matrix)
7: $b_a \leftarrow 0_{d\times 1}$ (d-dimensional zero vector)
8: **end if**
9: $\hat{\theta}_{t,a} \leftarrow A_a^{-1} b_a$
10: $p_{t,a} = \hat{\theta}_{t,a}^\top x_{t,a} + \alpha\sqrt{x_{t,a}^\top A_a^{-1} x_{t,a}}$
11: **end for**
12: Choose arm $a_t = \arg\max_{a\in\mathcal{A}_t} p_{a,t}$ with ties broken arbitrarily, and observe payoff r_t
13: $A_{a_t} \leftarrow A_{a_t} + x_{a,t} x_{a,t}^\top$
14: $b_{a_t} \leftarrow b_{a_t} + r_t x_{a,t}$
15: **end for**

Algorithm 1 performs iterative calculations to optimize user preferences through the Upper Confidence Bound (UCB) approach. Initially, we define the exploration parameter α as $1 + \sqrt{(\ln(2/\delta)/2)}$ (Li et al. 2010), which balances the exploration-exploitation trade-off in decision making under uncertainty.

During each round t, the algorithm executes the following steps to select actions and update rewards:

- Generation of possible calendar sets: Line 3 constructs all possible calendars ($\mathcal{A}_t$), representing different potential calendars, along with the corresponding feature vectors $x_{t,a}$ for each calendar a.
- Preference estimation: For each possible calendar a, the algorithm computes the estimated preference vector $\hat{\theta}_{t,a}$

using $\hat{\theta}_{t,a} \leftarrow A_a^{-1} b_a$, where A_a and b_a are dynamically updated parameters that incorporate observed user feedback to refine the estimate of the preference.

- UCB calculation: The main computational step occurs in Line 10, where the algorithm calculates the Upper Confidence Bound for each possible calendar ($a \in \mathcal{A}_t$). The first part calculate the estimated reward $\hat{\theta}_{t,a}^\top x_{t,a}$, then calculate the uncertainty bonus $\alpha\sqrt{x_{t,a}^\top A_a^{-1} x_{t,a}}$. The sum of these two calculation gives the UCB, which the algorithm uses to select the action with the highest potential for user satisfaction.
- Reward update: Following the selection of action a_t by $a_t = \arg\max_{a \in \mathcal{A}_t} p_{a,t}$, the algorithm updates the values of A_{a_t} and b_{a_t} based on the observed reward r_t, performed in line 13 and 14.

By iteratively updating the estimates and selections based on user interactions, Algorithm 1 effectively adapts to changing user preferences while ensuring a robust exploration of possible calendars.

Preference Perceptron (PP)

Algorithm 2: Preference Perceptron

1: Initialize $\hat{\theta}_1 \leftarrow 0$
2: **for** $t = 1$ to T **do**
3: Observe all features $x_{t,a}$
4: Present $a_t \leftarrow \arg\max_{a \in \mathcal{A}} \hat{\theta}_t^\top x_{t,a}$
5: Obtain feedback $\bar{a}_t$
6: Update $\hat{\theta}_{t+1} \leftarrow \hat{\theta}_t + x_{t,\bar{a}} - x_{t,a}$
7: **end for**

During each round t, the algorithm executes the following steps to select actions and update preferences:

- Action selection: First observes all the possible calendars (context) x_t for round t to determine potential actions in line 3. Then selecting the action y_t that maximizes the estimated preference based on the current model, calculated as $\hat{\theta}_t^\top x_{t,a}$ in line 4.
- Feedback acquisition: Line 5 acquires user feedback $\bar{a}_t$, which indicates the one of the calendar that users prefer of all the possible calendars x_t.
- Preference estimation update: To adjust the preference estimation with the feedback from users, line 6 updates the preference vector $\hat{\theta}_{t+1}$ using $\hat{\theta}_t + x_{t,\bar{a}} - x_{t,a}$. This update reduces the discrepancy between predicted and actual preferences to improve the model's accuracy.

By iteratively acquiring feedback from users after presenting the possible calendar in every round, Algorithm 2 can effectively adapt to user preferences by updating preferences to enhance decision-making processes over successive rounds.

Experiment

We conducted a comparative analysis between Linear UCB and PP on learning and adapt to users' preferences. We compared the trend of the regret value and human feedback noise for algorithm evaluation. To investigate dynamic scheduling challenges, a simulation environment was developed, mimicking a weekly calendar with 21 time slots distributed over seven days, each day segmented into three time slots: morning, afternoon, and evening.

Event Constraints and Calendar Simulation

The scheduling constraints differentiate between relevant and irrelevant events to examine the system's efficiency in prioritizing educational activities according to user preferences. Specifically:

- Relevant Events (e_r): The simulator schedules 'homework assignments' and 'reading book chapter' events related to the student's coursework. These relevant events required active scheduling into appropriate slots based on predefined constraints. The scheduling constraints reflect user preferences before the initial assignment. 'Homework assignments' events are restricted to Monday through Thursday, totaling 12 available slots, while 'reading book chapter' events can be scheduled in any of the 21 available slots throughout the week.
- Irrelevant Events (e_i): Events such as 'gym' and 'yoga class' represent pre-existing fixed appointments in the user's calendar and are considered irrelevant to the coursework. In our simulator, these events are assigned randomly, possibly including both e_i, one of the e_i, and none of the e_i. For instance, a 'gym' event might occupy one of the 21 total slots, and a 'yoga class' event could fill one of the remaining 20 slots or not fill any of the slots. This setup represents the user's calendar C_u and forms the initial state s of the user.

After initializing e_i, the system determines the placement of e_r. The remaining 19 to 21 slots are considered, ensuring that they do not overlap. For example, slots for 'homework assignments' are selected from the non-overlapping slots within the constrained 12, maintaining a structured yet randomized placement process.

Each simulation run starts with a calendar randomly populated with irrelevant events (e_i). Two algorithms then schedule relevant events, considering the available slots and their constraints. The experimental protocol conducts ten repetitions, each with simulation rounds of 100, $1,000$, and $10,000$ per algorithm. Initial simulations with fewer rounds offered limited insights into the algorithms' long-term performance. Therefore, increasing the number of rounds to $10,000$ enables the algorithms to face various scenarios. This approach allows the algorithms to adapt to users' decision-making processes based on continuous feedback and rewards. It provides the observation of long-term trends in how these algorithms learn and evolve.

Experimental Results

In our experiment, we initially examined the average regret, formally defined in Equation 2. This measure quantifies the

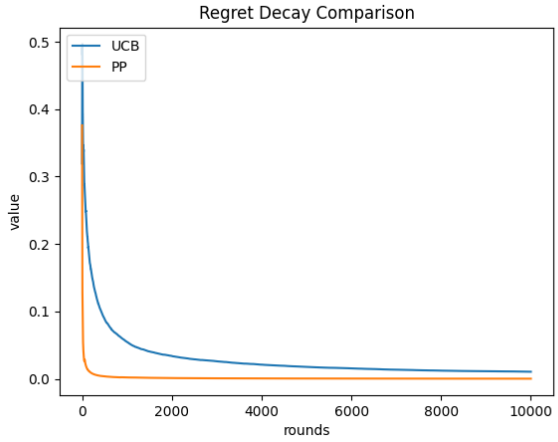

Figure 1: Comparison of average regret decay over 10 repeats for Linear UCB and PP algorithms.

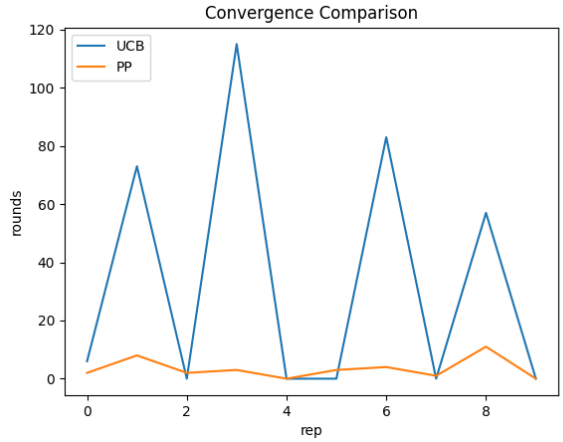

Figure 2: Regret convergence comparison of PP vs Linear UCB with convergence threshold of 0.25.

algorithm's performance over T time steps. In our study, regret refers to the difference between the reward obtained from the optimal action a_t and the reward from the action actually taken a_t, summed over all time steps and averaged. This equation quantifies the performance loss incurred by not selecting the optimal action in each state. The function $r(s_t, a_t)$ denotes the reward for the actual action taken at time t for the optimal action that could have been taken at the same time.

$$AvgREGT = \frac{1}{T}\sum_{t=1}^{T}(r(s_t, a_t^*) - r(s_t, a_t)) \qquad (2)$$

In general, the result shows that both algorithms performed well when the event constraints and event preferences were relatively stable and consistent across rounds, allowing the algorithms to learn and adapt effectively.

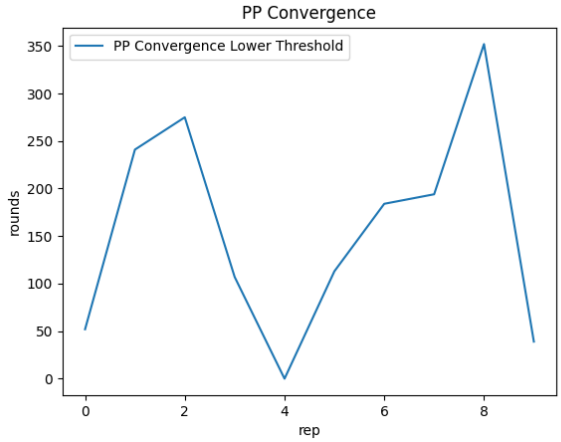

Figure 3: Regret convergence of PP algorithm with convergence threshold of 0.01.

Figure 1 shows the result of the average regret for both algorithms across 10,000 rounds over 10 repetitions.It shows that the average regret for the PP algorithm shows a steeper decline, implying a more rapid alignment with optimal choices. In contract, the Linear UCB algorithm shows a more gradual decline. The graph's trend lines converge over time, with the PP algorithm outperforming the Linear UCB algorithm.

Figure 2 shows the convergence performance of both the PP and Linear UCB algorithms under a threshold setting of 0.25 across the average value of 10,000 rounds over 10 repetitions. The data shows that the Linear UCB algorithm converges significantly slower than the PP. On average, Linear UCB requires 33.4 rounds to reach the convergence threshold, whereas the PP converges in just 3.4 rounds. Additionally, Figure 3 shows the convergence behavior of the PP under threshold of 0.01, also across 10 repetitions, where it takes an average of 155.7 rounds to converge.

Our results showed that the Linear UCB algorithm consistently recorded higher cumulative regret compared to the PP algorithm throughout the simulation iterations, pointing to the observation that the PP algorithm was more effective at balancing exploration and exploitation, adapting more swiftly to changes in preferences. Although the Linear UCB algorithm demonstrated adaptability, its performance is behind, indicating that in environments with stable and consistent preferences, the PP strategy tends to outperform.

While the PP algorithm has an initial advantage in learning rate, Linear UCB might eventually match its performance as it continues to refine its model with additional data. The fast learning capability of the PP algorithm could enhance user experience, particularly in settings like our study on calendar automation. Initially, users might need to interact more with the application to make changes to the presented actions. Still, this time investment can lead to future benefits as the system becomes more autonomous. Conversely, although the Linear UCB algorithm might require less user input in every round, its slower convergence could

delay the realization of automation benefits, possibly resulting in users leaving the application before the benefits are realized. We observed that this shows the importance of considering both user engagement and the temporal dynamics of learning in the development and deployment of the algorithms.

Influence of Human Feedback Noise

To assess the influence of human feedback on algorithmic efficacy, we performed experiments simulating human noise in the algorithmic decision-making process. For the Linear UCB algorithm, we experiment using the Gaussian distribution across three variance levels: 0.1, 1, and 10. The PP algorithm was evaluated under three conditions: with consistent optimal feedback, with optimal feedback provided 50% of the time, and without optimal feedback. In this context, "optimal" refers to feedback actions that are superior to those selected by the algorithm. Each algorithm was subjected to 10 repetitions, with each trial consisting of 2000 iterations, to ensure the reliability of the results.

For the Linear UCB algorithm, a clear inverse relationship between human noise levels and learning efficiency is observed. As indicated by the upward shift of the curve in Figure 4, increased noise corresponds to higher regret values, meaning slower learning. At the lowest noise level (0.1), the algorithm demonstrates optimal performance with minimal regret. However, with the introduction of higher noise levels, the learning curve slightly flattens, showing a decrease in algorithmic efficiency.

In contrast, the result of the PP algorithm shows a divergent response to varying degrees of human noise in Figure 5. With optimal feedback, it surpasses the Linear UCB algorithm in the lower regret values throughout the iterations. With optimal feedback available only half the time, the PP algorithm maintains a steady learning rate, though average regret values increased, stabilizing around 0.5. In scenarios when users are not providing any beneficial feedback, the algorithm encounters its greatest regret values, showing a slow learning efficiency.

This experiment shows the influence of human feedback noise on each algorithm's learning behaviors. Comprehending how feedback affects algorithmic performance is important for selecting appropriate algorithms that can integrate real-world human input effectively. Specifically, in our study, when developing calendars that learn from human preferences, improving their reliability and relevance in practical applications.

Future work and Conclusion

From our experiments, we observed that using a linear reward format, although effective in reducing storage requirements and run runtime, still resulted in slow learning rates due to the large size of the calendar. To enhance the learning speed and overall performance of the algorithm in this calendar-based scheduling system, future work could explore incorporating a structured hierarchy within the calendar. For instance, categorizing time slots into broader segments and refining decisions within these categories could allow the algorithm to handle decisions more efficiently.

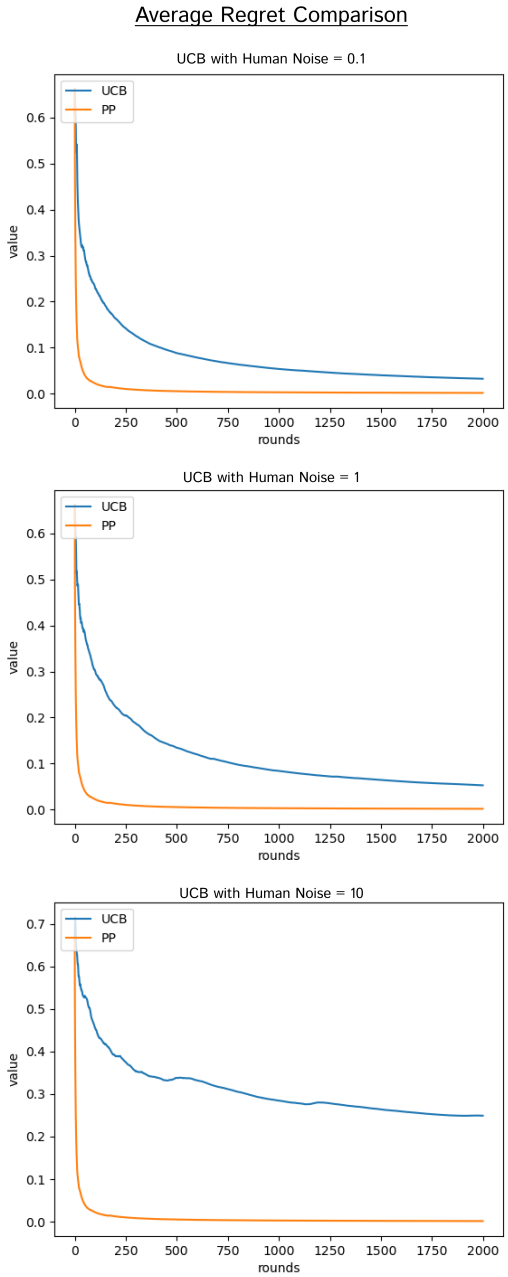

Figure 4: Comparison of average regret values with human noise using the **Linear UCB algorithm**, calculated over 10 repetitions of 2000 iterations each.

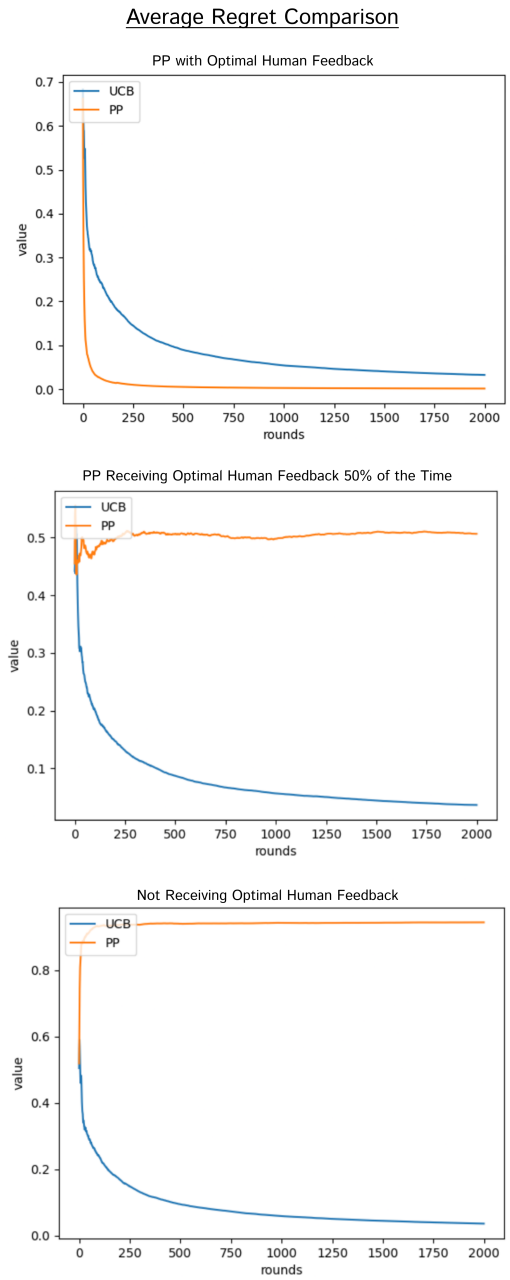

Figure 5: Comparison of average regret values with human noise using the **PP algorithm**, calculated over 10 repetitions of 2000 iterations each.

However, the use of a linear reward model has its limitations. In our study, we assumed that the reward dynamics are linear; if this assumption does not hold, the derived solutions may be incorrect. Future research could investigate the validity of this assumption and develop models that capture more complex reward structures.

Additionally, while the algorithms used in our study are broadly applicable to many general problems, they do not explicitly accommodate the unique constraints of the scheduling tasks in our calendar-based question. Future work could integrate specialized scheduling algorithms that are cognizant of event constraints, thereby enhancing the contextual relevance and effectiveness of the solutions.

Lastly, we note that while the PP algorithm can rapidly learn human preferences, it imposes a greater cognitive load compared to the Linear UCB approach. To address this, future research could explore hybrid models that combine the robust preference learning of PP with the more user-friendly feedback mechanisms of Linear UCB. Such hybrid approaches could potentially optimize both learning efficiency and user experience.

In conclusion, our investigation examined the performance of two algorithms, Linear UCB and PP, within a contextual bandit framework tailored for personalized calendar management. The comparative analysis revealed that the PP algorithm significantly outperformed Linear UCB in terms of regret minimization speed, demonstrating its capability to rapidly adapt to individual user feedback. Our study highlights the limitations of traditional static scheduling tools and underscores the potential of adaptive algorithms in developing personalized planning tools for students.

References

Abe, N.; Biermann, A. W.; and Long, P. M. 2003. Reinforcement learning with immediate rewards and linear hypotheses. *Algorithmica*, 37: 263–293.

Agarwal, A.; Hsu, D.; Kale, S.; Langford, J.; Li, L.; and Schapire, R. 2014. Taming the monster: A fast and simple algorithm for contextual bandits. In *International Conference on Machine Learning*, 1638–1646. PMLR.

Das Swain, V.; Hernandez, J.; Houck, B.; Saha, K.; Suh, J.; Chaudhry, A.; Cho, T.; Guo, W.; Iqbal, S.; and Czerwinski, M. P. 2023. Focused Time Saves Nine: Evaluating Computer–Assisted Protected Time for Hybrid Information Work. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, 1–18.

Gervasio, M. T.; Moffitt, M. D.; Pollack, M. E.; Taylor, J. M.; and Uribe, T. E. 2005. Active preference learning for personalized calendar scheduling assistance. In *Proceedings of the 10th international conference on Intelligent user interfaces*, 90–97.

Li, L.; Chu, W.; Langford, J.; and Schapire, R. E. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, 661–670.

Mitchell, T. M.; Caruana, R.; Freitag, D.; McDermott, J.; Zabowski, D.; et al. 1994. Experience with a learning personal assistant. *Communications of the ACM*, 37(7): 80–91.

Myers, K.; Berry, P.; Blythe, J.; Conley, K.; Gervasio, M.; McGuinness, D. L.; Morley, D.; Pfeffer, A.; Pollack, M.; and Tambe, M. 2007. An intelligent personal assistant for task and time management. *AI Magazine*, 28(2): 47–47.

Mynatt, E.; and Tullio, J. 2001. Inferring calendar event attendance. In *Proceedings of the 6th international conference on Intelligent user interfaces*, 121–128.

Oh, J.; and Smith, S. F. 2005. Calendar Assistants that Learn Preferences. In *AAAI Spring Symposium: Persistent Assistants: Living and Working with AI*, 7–13.

Shivaswamy, P.; and Joachims, T. 2015. Coactive learning. *Journal of Artificial Intelligence Research*, 53: 1–40.

Slivkins, A.; et al. 2019. Introduction to multi-armed bandits. *Foundations and Trends® in Machine Learning*, 12(1-2): 1–286.

Zaidenberg, S.; and Reignier, P. 2011. Reinforcement learning of user preferences for a ubiquitous personal assistant. *Advances in reinforcement learning*, 59–80.