# Feedback-Aware Social Event-Participant Arrangement

Jieying She[†], Yongxin Tong[‡], Lei Chen[†], Tianshu Song[‡]
[†]Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong SAR, PR China
[‡]SKLSDE Lab, School of Computer Science and Engineering, Beihang University, PR China
[†]{jshe, leichen}@cse.ust.hk [‡]{yxtong, songts}@buaa.edu.cn

## ABSTRACT

Online event-based social networks (EBSNs) and studies on global event-participant arrangement strategies for EBSNs are becoming popular recently. Existing works measure satisfaction of an arrangement by a linear combination of few factors, weights of which are predefined and fixed, and do not allow users to provide feedbacks on whether accepting the arrangement or not. Besides, most of them only consider offline scenarios, where full information of users is known in advance. However, on real-world EBSN platforms, users can dynamically log in the platform and register for events on a first come, first served basis. In other words, online scenarios of event-participant arrangement strategies should be considered. In this work, we study a new event-participant arrangement strategy for online scenarios, the Feedback-Aware Social Event-participant Arrangement (FASEA) problem, where satisfaction scores of an arrangement are learned adaptively and users can choose to accept or reject the arranged events. Particularly, we model the problem as a contextual combinatorial bandit setting and use efficient and effective algorithms to solve the problem. The effectiveness and efficiency of the solutions are evaluated with extensive experimental studies and our findings indicate that the state-of-the-art Thompson Sampling that is reported to work well under basic multi-armed bandit does not perform well under FASEA.

## 1. INTRODUCTION

Event-based social network (EBSN)[30] is a new type of online-to-offline social media that is becoming popular in recent years, on which *online* users can organize and register for *offline* social events. The most successful EBSN platform is Meetup[1], on which users can create groups and organize events by groups for other users to join. Other popular EBSNs include Plancast[2] and Eventbrite[3]. Such EBSNs facilitate social event organization and ease the recruitment of activity participants[30][37].

However, existing EBSNs only provide an information sharing platform that lacks strategic event organization and personalized event planning[38][37]. Recent works start to study global event arrangement strategies for EBSNs. [25] studies an event-user assignment problem and [4] proposes a game theoretical approach that can be applied for event arrangement. [38] and [37] further consider practical constraints such as conflicts of events, location information, and travel expenditure of users. [44] considers a fairness-oriented arrangement strategy. [17] considers diverse user choices. Note that all the works above only address offline scenarios, where the information (e.g. capacity, interest) of all the users is known before making arrangement and the arrangement is made simultaneously for all the users. However, in real world, users often log in the platform dynamically and register for events on a first come, first served basis. [39] extends [38] to an online setting.

Though online scenarios are brought into attention recently, there are still two other major drawbacks with existing works. First, existing works measure the quality of an arrangement by a single value or a linear combination of few factors, whose weights are predefined and fixed[25][38][37][4]. However, in many practical scenarios, the satisfaction of event organizers and users towards the arrangement can be complicated. For example, a user may prefer a less interesting event that is closer to her/his home location[30], or an event organized by a reputable organizer may be more attractive to users. Unfortunately, it is not easy to determine which factors contribute to the satisfaction and it is even harder to quantify and predefine the factors in practice. Second, existing works do not allow users to provide feedbacks on whether they are satisfied with the arrangement or not. In other words, they assume that the users will always accept and follow their arrangement. But in real world, users may not be satisfied with the arrangement. For example, we may arrange a heavy metal concert to a user who is interested in rock music, but the user may prefer folk rock rather than heavy metal and thus would rather reject the arrangement. Therefore, users should be allowed to accept or reject the arranged events and we can learn users' interest more accurately from their feedbacks.

In particular, the setting of the Multi-Armed Bandit (MAB) problem is quite suitable for addressing the above issues. MAB has been studied for decades[6], the most basic stochastic version[23] of which is that given a set of $m$ arms, each associated with an unknown distribution of rewards, one re-

[1]http://www.meetup.com
[2]http://plancast.com
[3]http://www.eventbrite.com


*SIGMOD'17, May 14-19, 2017, Chicago, IL, USA*
 ISBN 978-1-4503-4197-4/17/05…$15.00
DOI: http://dx.doi.org/10.1145/3035918.3064020

peatedly picks up and plays one of the $m$ arms in each round aiming to maximize the total expected rewards gained in multiple rounds. Only the reward of the arm played is observed and the reward can only be observed after the arm is played. In particular, the gap between the total rewards gained by an algorithm and those gained by playing the optimal arm is termed as *regret*, and thus the goal of an algorithm is to minimize its regret. Numerous variants of MAB have been studied through the years. One particular variant related to our event-participant arrangement setting is called the stochastic contextual bandit problem, where a set of features for each arm, a.k.a. contexts, are revealed in each round before one plays an arm. The expected rewards of the arms are functions of the features. One widely used reward function is the linear reward function, where the expected reward of an arm is a linear combination of its feature values, but the weights of the linear combination are unknown[26][13][1]. By playing arms in multiple rounds, a contextual bandit algorithm repeatedly explores the unknown weights based on the observed rewards while exploits the estimated optimal arm to minimize its regret. Another variant related to our setting is called the combinatorial semi-bandit problem, where a subset of arms rather than a single arm are played in each round and a reward is observed individually for each arm played[11][12][49]. Particularly, [36] studies the two variants simultaneously, a.k.a. the contextual combinatorial bandit problem. We can regard each arm as an event and playing a subset of arms in each around as making an online multiple-event arrangement for each newly-login user. Users' feedbacks on accepting/rejecting the arranged events are the observed rewards. Then, the contextual combinatorial bandit problem models well the event-participant arrangement scenario, where the contribution of each event/user factor (feature) to the satisfaction of the arrangement is unknown but can be estimated based on users' feedbacks.

Specifically, we model the arrangement problem as contextual combinatorial bandit, which we called the Feedback-Aware Social Event-participant Arrangement (FASEA) problem. Given a set of events $V$, each $v$ of which is associated with a capacity $c_v$, and a set of conflicting event pairs $CF$, a set of users $U$ log in the EBSN platform one by one as in [39]. When a newly-login user appears, a context vector $\boldsymbol{x}_v$ is revealed for each event $v$, which summarizes both the event and the user. The user also specifies her/his capacity $c_u$, which is the maximum number of events s/he would like to attend. We then respond to the user immediately with a proposed event-participant arrangement, which is a set of non-conflicting events whose capacities are not yet full, for the newly-arrived user. The user accepts each arranged event with probability $E[r_v|\boldsymbol{x}_v] = \boldsymbol{x}_v^T\boldsymbol{\theta}$, where $\boldsymbol{\theta}$ is an unknown and fixed weight vector that can be regarded as determining how each factor(feature) contributes to the satisfaction of a user towards an event. We then observe the reward, i.e. the user's feedback, of each arranged event, which is either 1 if it is accepted by the user or 0 otherwise. The total reward for this round is the number of accepted events. Based on the revealed feature vectors and the observed rewards for the arranged events, we estimate $\boldsymbol{\theta}$ and adjust our arrangement strategy adaptively. The FASEA problem is then to maximize the total number of accepted events arranged in multiple rounds, or equivalently to minimize the total regret w.r.t. the optimal strategy that has full knowledge of $\boldsymbol{\theta}$.

We use the following toy example for illustration.

EXAMPLE 1. *Suppose there are 4 events $v_1$ (football), $v_2$ (basketball), $v_3$ (concert) and $v_4$ (BBQ), among which football ($v_1$) is conflicting with basketball ($v_2$), on an EBSN platform, each with different topics, locations, time, organizers, etc. When a user $u_j$ logs in, who wishes to attend 2 events on the weekend, based on the attributes of the user, e.g. his/her interested topics, home location, age, etc., a set of features $\boldsymbol{x}_i$ regarding the user for each event $v_i$ are derived. Each event has a reward, which can be regarded as its accept ratio by the user and depends linearly on its features. Based on our current estimation of the accept ratios of the events, we arrange non-conflicting and non-full events $v_1$ and $v_3$ to the user $u_j$. $u_j$ then gives her feedback on accepting only $v_1$. We then adjust the estimates of the accept ratios of the events based on the feedback. When the next user $u_{j+1}$ logs in at the next time step, we arrange another set of events based on the adjusted estimates and adjust the estimates again based on the user's feedback.*

There are two particularly popular methods for MAB problems. The first is the Upper Confidence Bound (UCB) algorithm[5], which estimates an upper confidence bound on the reward of each arm based on the previous observations and plays the arm with the largest upper confidence bound in each round. The arms with large upper confidence bounds are either those that are actually better than the others or those that are under-explored and thus have loose (larger) upper confidence bounds. Therefore, UCB actually trades off exploration and exploitation simultaneously. UCB has been popular since it provides sound theoretical guarantees. Particularly, [36] proposes a UCB-based algorithm for contextual combinatorial bandit. Another state-of-the-art method is the Thompson Sampling (TS) algorithm[41], which samples a reward-depending parameter from a posterior distribution of the parameter and adjusts the distribution according to the observed rewards each time. TS had been less popular than UCB since it is challenging to analyze the performance of TS theoretically. In recent years, TS has regained its popularity as empirical studies[9] show that TS outperforms state-of-the-art bandit algorithms in practice. Theoretical analysis for TS has been provided in recent works[1][16][49]. Particularly, [1] proposes a TS-based algorithm for contextual bandit with linear payoff. Though existing studies[9] suggest that TS outperforms UCB under MAB, there still lacks of complete evaluation for UCB and TS under contextual combinatorial bandit and especially event-participant arrangement to the best of our knowledge. Therefore, in this paper, we use UCB and TS to solve the FASEA problem and evaluate their performance along with another popular greedy strategy.

The contributions of this paper are summarized as follows.

- We formulate the event arrangement problem under a contextual combinatorial bandit setting, called the Feedback-Aware Social Event-participant Arrangement (FASEA) problem, which makes online arrangement adaptively based on users' feedbacks.
- We present effective and efficient solutions based on state-of-the-art frameworks to solve the FASEA problem.

Table 1: Summary of Symbol Notations of FASEA

| Notation | Description |
|---|---|
| $t$ | Time step |
| $T$ | Total number of time steps |
| $V = \{v_1, \cdots, v_{\|V\|}\}$ | Set of events |
| $U = \{u_1, \cdots, u_t, \cdots, u_T\}$ | Users arrive at different time steps |
| $c_v$ | Capacity of $v$ |
| $c_u$ | Capacity of $u$ |
| $\boldsymbol{x}_{t,v}$ | Context of event $v$ at time step $t$ |
| $\boldsymbol{\theta}$ | Weight vector of the linear payoff |
| $r_{t,v}$ | Reward of event $v$ at time step $t$ |
| $A_t$ | Arrangement for $u_t$ |
| $A_t^*$ | Optimal arrangement for $u_t$ |
| $r_{t,A_t}$ | Reward of arrangement $A_t$ for $u_t$ |
| $Reg(T)$ | Regret of the arrangements made in $T$ time steps |
| $CF$ | A set of conflicting event pairs |
| $cr$ | Conflict ratio, $\frac{\|CF\|}{\|V\|(\|V\|-1)/2}$ |
| $\boldsymbol{Y}, \boldsymbol{b}, \lambda, \alpha, \delta, \epsilon$ | Algorithm parameters |

- We study the effectiveness and efficiency of the presented algorithms extensively on both synthetic and real datasets. Particularly, according to our findings, the state-of-the-art framework Thompson Sampling that is reported to perform well under basic multi-armed bandit[9] performs badly under FASEA.

The rest of this paper is organized as follows. In Section 2, we formally formulate the FASEA problem. Then we present algorithms based on the two state-of-the-art frameworks in Section 3 and Section 4. We evaluate the performance of the algorithms in Section 5. Related works are reviewed in Section 6. We finally conclude the work in Section 7.

## 2. PROBLEM STATEMENT

Let $V$ be a set of events. For each $v \in V$, it is associated with a capacity $c_v$ on the maximum number of attendees. Following [38][39], we constrain that users cannot attend conflicting events and use the following definition to denote the set of event pairs $CF$ that are conflicting with each other.

Definition 1 (Conflicting Event Pair[38][39]). *A pair of events $\{v_i, v_j\}$ are conflicting if a user can attend at most one of the two events but not both.*

Let $U$ denote the set of users, where each user $u$ arrives at the platform in an online way. Note that the size of $U$ is unknown in advance. At each time step $t = 1, 2, \cdots$, a user $u_t$ arrives at the platform and her/his capacity $c_{u_t}$ and contexts are revealed. Particularly, a feature vector $\boldsymbol{x}_{t,v} \in \mathbb{R}^d$ with $\|\boldsymbol{x}_{t,v}\| \leq 1$ is observed for each $v \in V$ at time step $t$, which consists of the values of the factors that affect the user's satisfaction towards event $v$. Particularly, under the setting of contextual bandit with linear payoff[5][26][13][1], the satisfaction of a user towards an event is defined as the reward of the event as follows.

Definition 2 (Reward of an Event). *The reward $r_{t,v}$ of an event $v$ at time step $t$ is a random variable with expectation $E[r_{t,v}|\boldsymbol{x}_{t,v}] = \boldsymbol{x}_{t,v}^T\boldsymbol{\theta}$, where $\boldsymbol{\theta} \in \mathbb{R}^d$ with $\|\boldsymbol{\theta}\| \leq 1$ is the weight vector of the linear function, which is fixed but unknown.*

Then for each new-coming user $u_t$ at time step $t$, we arrange a set of non-conflicting events $A_t \subset V$ where $|A_t| \leq c_{u_t}$ for $u_t$ and $u_t$ chooses to accept or reject each arranged event. The observed reward of an arranged event is 1 if it is accepted or 0 otherwise. Therefore, $\boldsymbol{x}_{t,v}^T\boldsymbol{\theta}$ can be regarded as the probability that $u_t$ accepts $v$. The reward $r_{t,A_t}$ of the arrangement for $u_t$ is then the number of accepted events.

$$r_{t,A_t} = \sum_{v \in A_t} \mathbf{1}(v \text{ is accepted}) \tag{1}$$

where $\mathbf{1}(\cdot)$ is the indicator function. Particularly, the performance of an algorithm is compared against an optimal strategy that has full knowledge of $\boldsymbol{\theta}$ and selects an optimal set of events $A_t^*$, where $A_t^* = \arg\max_{\text{feasible } A} \sum_{v \in A} \boldsymbol{x}_{t,v}^T\boldsymbol{\theta}$, at each time step. And the gap between the rewards gained by an algorithm in $T$ time steps and those gained by the optimal strategy is called the regret of the algorithm, termed as $Reg(T)$:

$$Reg(T) = \sum_{t=1}^{T} r_{t,A_t^*} - \sum_{t=1}^{T} r_{t,A_t} \tag{2}$$

where $r_{t,A_t^*}$ is the reward of the arrangement made by the optimal strategy at time step $t$. We finally define the Feedback-Aware Social Event-participant Arrangement (FASEA) problem as follows.

Definition 3 (FASEA Problem). *A set of events $V$, each $v$ of which has capacity $c_v$, and a set of conflicting event pairs $CF$ are given. At time step $t$, a user $u_t$ arrives at the EBSN platform with capacity $c_u$, which is unknown before $u$ appears, and a context $\boldsymbol{x}_{t,v}$ for each $v \in V$ is observed. Find an arrangement $A_t$ for each user $u_t$ and receive user's feedback on accepting or rejecting the arranged events such that the total number of accepted events is maximized and the following constraints[38][39] are satisfied:*

- *The arrangement for a new-coming $u$ must be decided before the next user appears and cannot be revoked.*
- *The capacity of each event and the capacity of each user are not exceeded.*
- *The events arranged to a user are not conflicting with each other.*

The notations of symbols used in this paper are summarized in Table 1.

**Remark 1.** Note that $\boldsymbol{\theta}$ is shared by the users in $U$ in FASEA. $U$ can be regarded as the same user sequentially arrives at the platform at different time steps and can as well be treated as a set of users with similar interests. It is also easy to extend FASEA to the scenario where different models ($\boldsymbol{\theta}$'s) are estimated for different users. That is, an individual $\boldsymbol{\theta}$ is learned for each user but the information of events (conflicts and capacities) is shared among the users. In other words, when an arranged event is accepted by a user, its capacity is reduced by 1, which will affect all the other users.

**Remark 2.** Note that the set of events $V$ does not change in FASEA. However, it is easy to extend FASEA to the scenario where different sets of events $V_t$ are revealed at different time steps. For example, when a user logs in on Monday, $V$ could be the set of events on Tuesday and when a user logs in on Friday, $V$ could be the set of events on the weekend.

**Algorithm 1:** TS

**input :** $V$, $CF$, $\{c_v|v \in V\}$, $\lambda$, $\delta$, $R$

1 $Y \leftarrow \lambda \boldsymbol{I}_{d\times d}$;
2 $\boldsymbol{b} \leftarrow \boldsymbol{0}_d$;
3 **for** $t \leftarrow 1$ **to** $T$ **do**
4 observe $c_{u_t}$ and $\{\boldsymbol{x}_{t,v}\}$;
5 $q \leftarrow R\sqrt{9d\ln(\frac{t}{\delta})}$ [2];
6 $\hat{\boldsymbol{\theta}}_t \leftarrow Y^{-1}\boldsymbol{b}$;
7 sample $\tilde{\boldsymbol{\theta}}_t$ from distribution $\mathcal{N}(\hat{\boldsymbol{\theta}}_t, q^2Y^{-1})$;
8 **for** $v \in V$ **do**
9 $\hat{r}_{t,v} \leftarrow \boldsymbol{x}_{t,v}^T\tilde{\boldsymbol{\theta}}_t$;
10 $A_t \leftarrow$Oracle-Greedy($V$, $CF$, $\{\hat{r}_{t,v}\}$, $\{c_v\}$, $c_{u_t}$);
11 arrange $A_t$ to $u_t$ and observe $\{r_{t,v}|v \in A_t\}$;
12 reduce the capacities of the accepted events by 1;
13 $Y \leftarrow Y + \sum_{v\in A_t}\boldsymbol{x}_{t,v}\boldsymbol{x}_{t,v}^T$;
14 $\boldsymbol{b} \leftarrow \boldsymbol{b} + \sum_{v\in A_t} r_{t,v}\boldsymbol{x}_{t,v}$;

## 3. THOMPSON SAMPLING BASED ALGORITHM

In this section, we present the first algorithm based on the state-of-the-art framework Thompson Sampling. The main idea of Thompson Sampling is to assume a prior distribution on the reward-depending parameter and play the arm based on its posterior probability of being the best arm in each round[1]. After observing the reward of the arm played, the posterior probability will be adjusted. Then under contextual bandit, a weight factor $\boldsymbol{\theta}$ is sampled from its posterior distribution and the best arm based on the sampled $\boldsymbol{\theta}$ is played at each time step. Notice that different with UCB, Thompson Sampling does not estimate an upper confidence bound for each arm individually but maintains the overall posterior distribution of the reward-depending parameter $\boldsymbol{\theta}$. Particularly, [1][2] develop a Thompson Sampling algorithm for contextual bandit with linear payoff and we extend it to our contextual combinatorial setting that considers capacities/conflicts of events and arranges multiple events each time, which we call TS.

The TS algorithm is illustrated in Algorithm 1. Specifically, we maintain a $d \times d$ matrix $Y$ and a $d \times 1$ vector $\boldsymbol{b}$, which are updated in each round using the observed contexts and rewards of the arranged events. We make arrangement for each new-coming user in lines 3-14. When a user arrives at time step $t$, we first observe her/his capacity $c_{u_t}$ and the feature vectors in line 4. In line 6, we estimate $\boldsymbol{\theta}$ based on $Y$ and $\boldsymbol{b}$ following ridge regression[26]. Then based on the estimated $\hat{\boldsymbol{\theta}}_t$, a $\tilde{\boldsymbol{\theta}}_t$ is sampled from the Normal distribution $\mathcal{N}(\hat{\boldsymbol{\theta}}_t, q^2Y^{-1})$ [1][2] in line 7. Then based on the sampled $\tilde{\boldsymbol{\theta}}_t$, we calculate the estimated expected reward of each event in lines 8-9. We then make an arrangement $A_t$ for the current user based on the estimated expected rewards in line 10 with an oracle algorithm. As mentioned previously that making a non-conflicting arrangement for a user is NP-hard[38], we adopt a greedy strategy Oracle-Greedy to make an arrangement approximately, which will be explained shortly. We then arrange events $A_t$ to $u_t$ and observe the rewards of the arranged events, i.e. whether they are accepted or rejected. We finally update $Y$ and $\boldsymbol{b}$ accordingly in lines 13-14. Note that the parameter $R$ in Algorithm 1 is to restrict $r_{t,v} - \boldsymbol{x}_{t,v}^T\boldsymbol{\theta}$ such that it is conditionally $R$-sub-Gaussian according to [1][2], and this assumption is satisfied whenever $r_{t,v} \in [\boldsymbol{x}_{t,v}^T\boldsymbol{\theta} - R, \boldsymbol{x}_{t,v}^T\boldsymbol{\theta} + R]$ [1][2]. Thus, under FASEA, $R$ is simply 1.

**Algorithm 2:** Oracle-Greedy

**input :** $V$, $CF$, $\{\hat{r}_{t,v}|v \in V\}$, $\{c_v|v \in V\}$, $c_u$
**output:** $A_t$

1 $A_t \leftarrow \emptyset$;
2 **for** $v \in V$ *in order of non-increasing* $\hat{r}_{t,v}$ **do**
3 break if $|A_t| \geq c_u$;
4 **if** $v$ *does not conflict with events in* $A_t$ *and* $c_v > 0$ **then**
5 $A_t \leftarrow A_t \cup \{v\}$;
6 **return** $A_t$

As making a non-conflicting arrangement for a user is NP-hard as [38] indicates, the Oracle-Greedy algorithm we adopt is an approximation algorithm. The algorithm is illustrated in Algorithm 2. When making arrangement, we visit each event in non-increasing order of $\hat{r}_{t,v}$ until the number of arranged events has reached the capacity of $u_t$. We then arrange each visited non-full event to $u_t$ if it does not conflict with the events that are already arranged.

EXAMPLE 2. *Back to our running example. Recall that we have 4 events and $v_1$ is conflicting with $v_2$. Features and $c_u$ of the first two rounds are presented in Table 2. In the first round, $\boldsymbol{\theta}$ is sampled as $< -11.28, 0.93, -13.07, 18.60 >$ and the estimated expected rewards of $v_1$ to $v_4$ are respectively -3.94, -0.30, 1.74, -13.07. Then $v_2$ and $v_3$ are arranged to $u_1$, which are all rejected. Then after updating $Y$ and $\boldsymbol{b}$, in the second round, $v_3$ is arranged to $u_2$, which is then accepted by $u_2$.*

Notice that Oracle-Greedy may include events with $\hat{r} \leq 0$ into $A_t$, which could result in smaller $\sum_{v\in A_t}\hat{r}_{t,v}$. However, actually it is beneficial to do so: As Oracle-Greedy visits $v$ in non-increasing order of $\hat{r}$, events with $\hat{r} \leq 0$ are included into $A_t$ simply because $A_t$ is not yet full and other events with $\hat{r} > 0$ (visited previously) cannot be added to $A_t$ due to conflicts or such events are full. On the other hand, as $\hat{r}$ are only estimated expected rewards, events with $\hat{r} \leq 0$ could actually have true expected rewards larger than 0. In other words, events with $\hat{r} \leq 0$ may be accepted by users as the estimation of the expected rewards may not be accurate. Therefore, as no other events with $\hat{r} > 0$ could be added to $A_t$ while $A_t$ is not yet full, it does no harm to include events with $\hat{r} \leq 0$ into $A_t$. Then when we next study the approximation ratio of Oracle-Greedy, we only consider $\sum_{v\in A_t|\hat{r}_{t,v}>0}\hat{r}_{t,v}$.

Table 2: Features and $c_u$ of running example.

| Round | Features | $c_u$ |
|---|---|---|
| 1 | $v_1$: 0.1, 0, 0.5, 0.2<br>$v_2$: 0.2, 0.1, 0, 0.1<br>$v_3$: 0.2, 0.3, 0, 0.2<br>$v_4$: 0, 0, 1, 0 | 2 |
| 2 | $v_1$: 0.2, 0.1, 0.2, 0.1<br>$v_2$: 0.1, 0.2, 0, 0.1<br>$v_3$: 0, 0, 0, 0.5<br>$v_4$: 0.2, 0.1, 0.4, 0 | 1 |

THEOREM 1. *The approximation ratio of Oracle-Greedy is* $\frac{1}{c_u}$, *i.e.* $\sum_{v\in A_t|\hat{r}_{t,v}>0}\hat{r}_{t,v} \geq \frac{1}{c_u}\sum_{v\in A_t^{OPT}|\hat{r}_{t,v}>0}\hat{r}_{t,v}$, *where* $A_t^{OPT}$ *is the optimal arrangement based on the estimated rewards* $\hat{r}_{t,(\cdot)}$.

PROOF. Please refer to the Appendix. □

**Complexity Analysis.** Oracle-Greedy takes $O(|V|\log|V|)$ to sort $V$ in order and at most $c_u|V|$ time to check conflicts. Therefore, the time complexity of Oracle-Greedy is $|V|(\log|V|+c_u)$ and its space complexity is $O(|V|)$. In each round of TS, it takes $O(d^3)$ time to calculate the inverse of $Y$ and $O(d^2)$ time to calculate $\hat{\boldsymbol{\theta}}_t$. Then sampling $\tilde{\boldsymbol{\theta}}_t$ takes additional $O(d^3)$ time. Calculating the estimated expected rewards takes $O(d|V|)$ time. Updating $Y$ and $\boldsymbol{b}$ takes $O(c_u d^2)$ time. Therefore, the overall time complexity of TS to process a user is $O(d^2(c_u+d)+d|V|+|V|(\log|V|+c_u))$. The major space is consumed by $Y$ besides storing expected rewards of $V$ and thus the space complexity of TS is $O(d^2+|V|)$. Since $d$ is usually small in practice or after applying dimensionality reduction, TS is still efficient in time and space.

# 4. UCB BASED ALGORITHM

The Upper Confidence Bound (UCB) framework is popular for MAB problems due to its sound theoretical guarantees. Since it was introduced in [5], it has been adapted to solving different variants of MAB problems, such as contextual bandit with linear payoff [5][26][36] and combinatorial bandit [11][12][36].

The main idea of the UCB framework is to infer an upper confidence bound on the reward of each arm and the arm(s) with the largest upper bound(s) is(are) played in each round. One typical way to infer the upper bound is to use concentration inequalities, which bound the deviation of a random variable from its expectation. Initially, all the arms have the same upper confidence bound, which is quite loose. Then as we play arms repeatedly, the observations of the arms played will help estimate their upper confidence bounds more accurately. Therefore, the arms without enough exploration, which generally have large(loose) upper confidence bounds, can be explored with high probability and thus the algorithm can avoid being trapped in local optimum. In addition, by playing the arm(s) with the largest upper bound(s), i.e. the largest possible reward(s), we are actually trying to play the most optimal arm(s) in each round. Therefore, UCB is a general framework that does exploration and exploitation simultaneously. Adaptations of UCB to different variants of MAB generally calculate different upper confidence bounds on the arms based on different concentration inequalities. Particularly, we adapt the UCB framework from [36], a UCB-based algorithm for contextual combinatorial bandit based on [26][13], by providing Oracle-Greedy to select events and considering capacities/conflicts of events to solve our FASEA problem.

The adapted UCB[36] algorithm is illustrated in Algorithm 3. When a new-coming user arrives, we first estimate $\hat{\boldsymbol{\theta}}_t$ in line 5 similar as TS. Then for each event, we estimate its expected reward based on the estimated $\hat{\boldsymbol{\theta}}_t$ in line 7 and calculate an upper confidence bound on its reward in line 8 based on the concentration inequality[48][26]. We then use Oracle-Greedy again to find an approximate arrangement for the user based on the estimated upper confidence bounds in line 9 and observe the rewards and update the capacities and estimates in lines 10-13.

**Algorithm 3:** UCB

**input** : $V$, $CF$, $\{c_v|v\in V\}$, $\lambda$, $\alpha$
1 $Y \leftarrow \lambda \boldsymbol{I}_{d\times d}$;
2 $\boldsymbol{b} \leftarrow \boldsymbol{0}_d$;
3 **for** $t \leftarrow 1$ **to** $T$ **do**
4  observe $c_{u_t}$ and $\{\boldsymbol{x}_{t,v}\}$;
5  $\hat{\boldsymbol{\theta}}_t \leftarrow Y^{-1}\boldsymbol{b}$;
6  **for** $v \in V$ **do**
7  $\tilde{r}_{t,v} \leftarrow \boldsymbol{x}_{t,v}^T\hat{\boldsymbol{\theta}}_t$;
8  $\hat{r}_{t,v} \leftarrow \tilde{r}_{t,v} + \alpha\sqrt{\boldsymbol{x}_{t,v}^T Y^{-1}\boldsymbol{x}_{t,v}}$;
9  $A_t \leftarrow$Oracle-Greedy($V$, $CF$, $\{\hat{r}_{t,v}\}$, $\{c_v\}$, $c_{u_t}$);
10  arrange $A_t$ to $u_t$ and observe $\{r_{t,v}|v\in A_t\}$;
11  reduce the capacities of the accepted events by 1;
12  $Y \leftarrow Y + \sum_{v\in A_t}\boldsymbol{x}_{t,v}\boldsymbol{x}_{t,v}^T$;
13  $\boldsymbol{b} \leftarrow \boldsymbol{b} + \sum_{v\in A_t} r_{t,v}\boldsymbol{x}_{t,v}$;

**Algorithm 4:** eGreedy

**input** : $V$, $CF$, $\{c_v|v\in V\}$, $T$, $\epsilon$
1 $Y \leftarrow \lambda \boldsymbol{I}_{d\times d}$;
2 $\boldsymbol{b} \leftarrow \boldsymbol{0}_d$;
3 **for** $t \leftarrow 1$ **to** $T$ **do**
4  observe $c_{u_t}$ and $\{\boldsymbol{x}_{t,v}\}$;
5  sample $p$ from $\mathcal{U}(0,1)$;
6  **if** $p \leq \epsilon$ **then**
7  $A_t \leftarrow$ at most $c_u$ non-conflicting events selected randomly;
8  **else**
9  $\hat{\boldsymbol{\theta}}_t \leftarrow Y^{-1}\boldsymbol{b}$;
10  $\hat{r}_{t,v} \leftarrow \boldsymbol{x}_{t,v}^T\hat{\boldsymbol{\theta}}_t, \forall v \in V$;
11  $A_t \leftarrow$Oracle-Greedy($V$, $CF$, $\{\hat{r}_{t,v}\}$, $\{c_v\}$, $c_{u_t}$);
12  arrange $A_t$ to $u_t$ and observe $\{r_{t,v}|v\in A_t\}$;
13  reduce the capacities of the accepted events by 1;
14  $Y \leftarrow Y + \sum_{v\in A_t}\boldsymbol{x}_{t,v}\boldsymbol{x}_{t,v}^T$;
15  $\boldsymbol{b} \leftarrow \boldsymbol{b} + \sum_{v\in A_t} r_{t,v}\boldsymbol{x}_{t,v}$;

EXAMPLE 3. *Back to our running example. When* $u_1$ *arrives, the estimated expected rewards of* $v_1$ *to* $v_4$ *are respectively 1.10, 0.49, 0.82 and 2.00. Then* $v_1$ *and* $v_4$ *are arranged to* $u_1$, *which are both accepted by* $u_1$. *Then after updating* $Y$ *and* $\boldsymbol{b}$, *in the second round,* $v_3$ *is arranged to* $u_2$, *which is also accepted by* $u_2$.

**Complexity Analysis.** Compared with TS, UCB saves $O(d^3)$ time of sampling but takes additional $O(d^2)$ time for each event to calculate the upper confidence bounds. Therefore, the overall time complexity of UCB to process a user is $O(d^2(c_u+d+|V|))+|V|(\log|V|+c_u))$. Similarly, the space complexity of UCB is $O(d^2+|V|)$.

## 4.1 Two Heuristics

In UCB, exploration is performed by the upper confidence bound in line 8 of Algorithm 3. In the literature, $\epsilon$-Greedy [47] is another heuristic that trades off between exploration and exploitation, the main idea of which is to greedily play some random arms in certain rounds while play the arms with the largest estimated expected rewards in the other rounds. Specifically, extending to FASEA, with probability

$\epsilon$, we arrange events randomly, which is exploration; with probability $1-\epsilon$, we arrange events greedily based on the current estimates, which is exploitation. We call this heuristic eGreedy, which is illustrated in Algorithm 4. Note that the difference between eGreedy and UCB is that (1) events are arranged randomly with probability $\epsilon$ (lines 6-7); (2) events are arranged greedily based on their estimated expected rewards with probability $1-\epsilon$ (lines 9-11).

Another heuristic is a pure greedy strategy, which is a special case of UCB or eGreedy: always arranging events based on their estimated expected rewards greedily. Specifically, letting $\alpha = 0$ for UCB or $\epsilon = 0$ for eGreedy, this pure greedy strategy only conducts exploitation each time. We call this strategy Exploit. The time complexity of eGreedy and Exploit to process a user is both $O(d^2(c_u+d)+d|V|+|V|(\log|V|+c_u))$ and their space complexity is $O(d^2+|V|)$.

# 5. PERFORMANCE EVALUATION

## 5.1 Experiment Setup

Table 3: Real Dataset of FASEA

| **Category** | **Sub-Category** |
|---|---|
| Pop Concert | Pop, classic, folk, jazz |
| Theater | Drama, opera, musical, children drama |
| Sports | Basketball, football, boxing |
| Folk Art | Cross talk, magic, acrobatics |
| Music | Piano, orchestral, choral |
| Movie | Adventure, cartoon, romance, fantasy, documentary, horror, comedy |
| **Other Features** | **Values** |
| Performers | Male, female, group |
| Country/District | Hong Kong, Taiwan, Mainland China, Japan, USA, UK, France, Denmark Germany, Canada, Poland |
| Lowest Price | 0-49, 50-99, 100-149, 150-199, 200-299 300-399, 400-599, $\geq$600 |
| Day of Week | Wed, Fri, Sat, Sun, Any |
| Normalized Distance | [0, 1] |

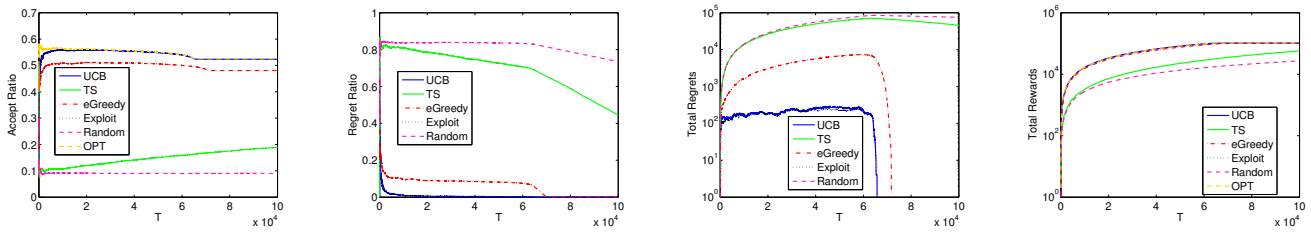
Table 4: Synthetic Dataset of FASEA

| Factor | Setting |
|---|---|
| $\|V\|$ | 100, **500**, 1000 |
| $T$ | 100000 |
| $d$ | 1, 5, 10, 15, **20** |
| Distribution of $\boldsymbol{\theta}$ | Uniform: [-1, 1], Power: 2, $\boldsymbol{\mathcal{N}(0,1)}$ |
| Distribution of $\boldsymbol{x}$ | Uniform: [-1, 1], Power: 2, $\boldsymbol{\mathcal{N}(0,1)}$, Shuffle |
| $c_v$ | $\mathcal{N}(100,100)$, $\boldsymbol{\mathcal{N}(200,100)}$, $\mathcal{N}(500,200)$ |
| $c_u$ | Uniform: [1, 5] |
| $cr$ | 0, **0.25**, 0.5, 0.75, 1 |
| Algorithm parameter $\lambda$ | 0.5, **1**, 2 |
| UCB parameter $\alpha$ | 1, 1.5, **2**, 2.5 |
| TS parameter $\delta$ | 0.05 **0.1**, 0.2 |
| eGreedy parameter $\epsilon$ | 0.05, **0.1**, 0.2 |

We use both real and synthetic datasets for experiments. For real dataset, we collected 50 popular events in Beijing from Damai.com and asked 19 users to provide ground-truth feedbacks. Specifically, six categories of events were collected, which are respectively pop concerts, theaters, sports, folk arts, music and movies, and each category is further classified into several sub-categories as shown in Table 3. We also collected other features such as performers, location, the lowest price, etc. for each event. Details of categorization and other features are presented in Table 3. For each categorical feature, we encode it into a binary vector following [26]. For example, the performer(s) of an event can be male, female or group, and thus the corresponding feature values are respectively encoded as $<0,1>$, $<1,0>$ and $<1,1>$. We calculate the earth distances between users' home locations and the events' locations and further normalize each distance to interval $[0,1]$ and use it as a numerical feature. Concatenating the categorical and the numerical features, each event is represented by a 20-dimensional feature vector. We finally normalize the feature vectors by dividing each feature value by $d=20$ to satisfy that $\|\boldsymbol{x}_{t,v}\| \leq 1$. Events' time and location information is used to decide which event pairs are conflicting. For example, a concert at 2016.10.21 7:30 pm is conflicting with another one at 2016.10.21 7:00 pm. 19 users were asked to give "Yes" or "No" feedbacks on attending each of the 50 events or not. We test two types of user capacities. One is that each user has capacity of 5, which we denote as "$c_u = 5$", and another one is that the capacity of each user is equal to the number of events with feedbacks of "Yes", which we denote as "$c_u$ = full". For example, a user with 12 "Yes" and another one with 11 "Yes" will have "$c_u$ = full" capacities of 12 and 11 respectively. In the real dataset, to test how quickly each algorithm can learn users' favored events, we display the same set of feature vectors in each round for the 50 events. We further calculate at most how many events can be accepted by each user given their ground-truth feedbacks and the conflict information, which we denote as "Full Knowledge".

For synthetic dataset, we generate real $\boldsymbol{\theta}$ and feature values following Uniform, Normal and Power distributions. For feature vectors, we further generate their values in a "shuffle" way to generate more "random" features: the value of each dimension $i$ is generated following Uniform, Normal with mean $i/d$ and Power distributions in turn. For example, the values of the 1st, 4th, ..., dimensions follow Uniform distribution, that of the 2nd dimension follow Normal distribution with mean $2/d$ and those of the 3rd, 6th, ..., dimensions follow Power distribution. $\boldsymbol{\theta}$ and feature vectors are normalized to unit lengths. Then the feedback of an event is 1 with probability $\boldsymbol{x}_{t,v}^T\boldsymbol{\theta}$ and 0 otherwise. Capacities of events and users are generated following Normal and Uniform distributions, respectively. Statistics and configuration of synthetic data are illustrated in Table 4, where default values are marked in bold font. Particularly, $cr$ denotes the conflict ratio of events, which is $\frac{|CF|}{|V|(|V|-1)/2}$. We denote "OPT" as the strategy that knows the true values of $\boldsymbol{\theta}$ and uses Oracle-Greedy to select events greedily based on the true expected rewards of the events.

In addition to "Full Knowledge" for real dataset and "OPT" for synthetic dataset, we compare 5 algorithms on both real and synthetic dataset, which are respectively UCB, TS, eGreedy, Exploit and Random. The Random algorithm visits each $v \in V$ in a random order and the rest is the same as lines 3-5 of Oracle-Greedy. We introduce the following metrics to evaluate the algorithms: (1) Accept ratio. The ratio of the number of accepted events to the number of arranged events at each time step $t$, i.e. $\frac{r_{t,A_t}}{|A_t|}$. (2) Total rewards. The accumulated number of accepted events in multiple rounds, i.e. $\sum_{t=1}^{T} r_{t,A_t}$. (3) Total regrets. The accumulated regrets (compared with "Full Knowledge" for real dataset and "OPT" for synthetic dataset) in multiple rounds. (4) Regret ratio. The ratio of total regrets to total rewards. (5) Average




(a) Accept Ratio (b) Regret Ratio (c) Total Regrets (d) Total Rewards

Figure 1: Results of FASEA under default setting.

running time of each round and memory consumption. Note that we calculate the accept ratio of "Full Knowledge" as the maximum number of non-conflicting events that can all be accepted by a user divided by the user's capacity, assuming that we still arrange $c_u$ events to a user even if it is impossible to arrange $c_u$ non-conflicting events all with feedbacks of "Yes". Otherwise, the accept ratio of "Full Knowledge" would always be 1, which would be meaningless to compare with. The algorithms are implemented in C++, and the experiments were performed on a Windows 7 machine with Intel i7-2600 3.40GHZ 8-core CPU and 8GB memory.

## 5.2 Experiment Results

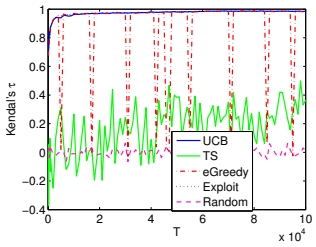



Figure 2: Kendall's rank correlation coefficients under default setting of FASEA.

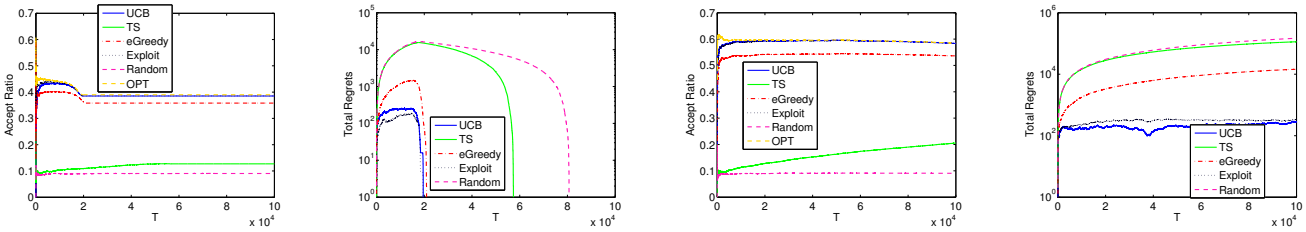
In this section, we compare all the experiments under FASEA.

**Results under default setting.** Figure 1 shows the results under default setting, which is marked in bold font in Table 4. First, we can observe that all the algorithms except Random performs better as $t$ increases, as their accept ratios gradually increase and their regret ratios gradually decrease. It indicates that the estimation of the algorithms on $\boldsymbol{\theta}$ becomes more accurate as they observe more feedbacks. Notice that the total regrets and regret ratios of the algorithms drop suddenly around $t = 70000$. The reason is that as under FASEA, events have limited capacities, OPT has assigned all the events when $t = 65664$ and thus no events are available when $t > 65664$ for OPT and thus the total rewards of OPT no longer increase afterwards. Therefore, the gap between the total rewards of the other algorithms and those of OPT will be narrowed down. Thus, algorithms that reduce their regrets to 0 faster, e.g. UCB and Exploit, perform better than the others. And we can also notice that the accept ratios of UCB, Exploit and eGreedy drop a bit before becoming constant, at the time step where their regrets become zero. The drop of accept ratios is because only few events are available at that time and those few events may not be accepted by some upcoming users. Note that the total regrets result can reflect the regret ratio and total rewards results. Second, except Random, TS performs the worst and Exploit is slightly better than UCB, which is different from the literature that TS generally performs better, though under basic MAB setting rather than contextual combinatorial setting. One possible reason is that under contextual bandit, all the arms (events) are correlated due to the shared $\boldsymbol{\theta}$ and thus playing one arm can help estimate all the other arms while under basic MAB, all the arms have independent reward distributions and thus playing one arm cannot help estimate others. Therefore, the sampling strategy of TS that intends to exploit the uncertainty of the rewards of arms does not help to estimate the expected rewards of arms in reality, and UCB, eGreedy, Exploit that can improve their estimation in each round simply based on the feedbacks observed can gradually improve their performance.

For further investigation, we study whether the algorithms can estimate the expected rewards of the events accurately by comparing their rankings on the expected rewards of the events with the ground-truth ranking (based on the ground-truth $\boldsymbol{\theta}$). Figure 2 shows the Kendall's rank correlation coefficient of the rankings of the algorithms compared with that of OPT under the default setting of FASEA, where the Kendall's rank correlation is calculated as follows: $\tau = \frac{(\#\text{of concordant pairs}) - (\#\text{of discordant pairs})}{n(n-1)/2}$ [19]. Specifically, we calculate the correlation coefficient between two rankings of the events based on their estimated / ground-truth expected rewards at time steps 100, 200, ..., 900, 1000, 2000, 3000, ..., 100000. We can on one hand observe that the correlation coefficients of UCB and Exploit approach to 1 as $t$ increases, indicating that these two algorithms estimate the expected rewards of the events more accurately, which explains their good performance. Notice that eGreedy generally has large correlation coefficients but has low correlation with OPT sometimes due to its random strategy. On the other hand, the correlation between TS and OPT fluctuates a lot though increases gradually. The fluctuation indicates that TS could bring a lot of noise due to its sampling strategy, which could explain the bad performance of TS. Finally, as expected, Random is generally uncorrelated with OPT.

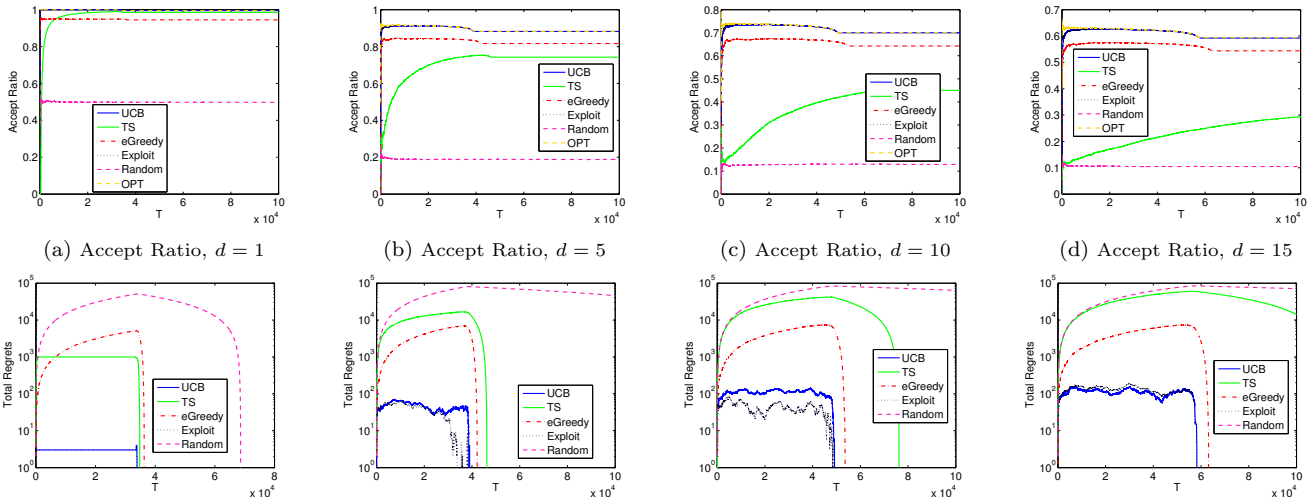
As for efficiency, the average running time of each round and the memory consumption of the algorithms are presented in the column "$|V| = 500$" of Table 5. We can see that all the algorithms are quite efficient and eGreedy and Exploit are the most efficient in time (except Random). As TS takes additional time to sample a $\boldsymbol{\theta}$ each time, TS is less efficient than eGreedy and Exploit.

In the following groups of experiments, for brevity, we mainly present the results of accept ratios and total regrets




(a) Accept Ratio, $|V| = 100$ (b) Total Regrets, $|V| = 100$ (c) Accept Ratio, $|V| = 1000$ (d) Total Regrets, $|V| = 1000$

Figure 3: Results of FASEA when $|V| = 100$ and 1000.




(a) Accept Ratio, $d = 1$ (b) Accept Ratio, $d = 5$ (c) Accept Ratio, $d = 10$ (d) Accept Ratio, $d = 15$

(e) Total Regrets, $d = 1$ (f) Total Regrets, $d = 5$ (g) Total Regrets, $d = 10$ (h) Total Regrets, $d = 15$

Figure 4: Results of FASEA on effect of dimension.

Table 5: Average Running Time and Memory Consumption of FASEA with Varying $|V|$.

| Algorithm | Avg Time(sec) | | |
|---|---|---|---|
| | $\|V\| = 100$ | $\|V\| = 500$ | $\|V\| = 1000$ |
| UCB | 0.0015 | 0.0055 | 0.0090 |
| TS | 0.0024 | 0.0033 | 0.0042 |
| eGreedy | 0.0007 | 0.0015 | 0.0025 |
| Exploit | 0.0008 | 0.0014 | 0.0027 |
| Random | 2.1e-5 | 8.4e-5 | 0.0001 |
| **Algorithm** | **Memory(MB)** | | |
| | $\|V\| = 100$ | $\|V\| = 500$ | $\|V\| = 1000$ |
| UCB | 4.13 | 5.49 | 9.52 |
| TS | 4.15 | 5.52 | 9.55 |
| eGreedy | 4.06 | 5.31 | 9.59 |
| Exploit | 4.05 | 5.30 | 9.57 |
| Random | 4.04 | 5.41 | 9.46 |

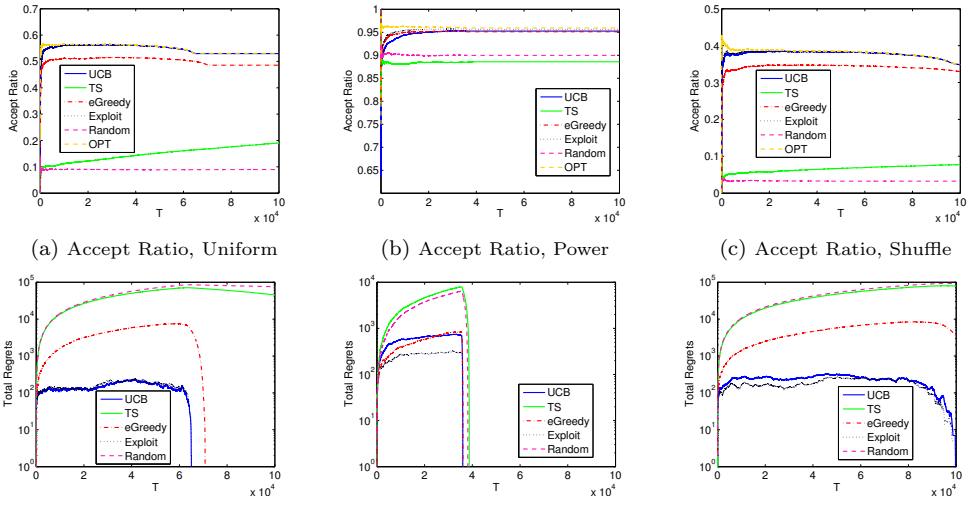
Table 6: Average Running Time and Memory Consumption of FASEA on Effect of Dimension.

| | Avg Time(sec) | | | |
|---|---|---|---|---|
| | $d = 1$ | $d = 5$ | $d = 10$ | $d = 15$ |
| UCB | 0.0014 | 0.0020 | 0.0028 | 0.0039 |
| TS | 0.0003 | 0.0007 | 0.0013 | 0.0021 |
| eGreedy | 0.0003 | 0.0005 | 0.0007 | 0.0011 |
| Exploit | 0.0003 | 0.0005 | 0.0008 | 0.0012 |
| Random | 7.7e-5 | 7.3e-5 | 6.9e-5 | 6.5e-5 |
| | **Memory(MB)** | | | |
| | $d = 1$ | $d = 5$ | $d = 10$ | $d = 15$ |
| UCB | 5.43 | 5.43 | 5.44 | 5.47 |
| TS | 5.45 | 5.45 | 5.47 | 5.48 |
| eGreedy | 5.37 | 5.38 | 5.39 | 5.40 |
| Exploit | 5.37 | 5.37 | 5.36 | 5.39 |
| Random | 5.36 | 5.36 | 5.37 | 5.39 |

as the results of the other two metrics can be reflected from these two metrics.

**Effect of $|V|$.** The results when $|V|$ is respectively 100 and 1000 are presented in Figure 3. Similar to the previous group of experiments, we can observe that except Random, TS is still the worse while UCB and Exploit are still the best. Comparing with Figure 1, we can observe that the algorithms (except Random) generally have larger accept ratios when $|V|$ is larger. The possible reason is that as the features are randomly generated following the same distribution, more events are likely to have large expected rewards and thus to have feedbacks of 1 when $|V|$ is larger. Notice that the total regrets drop earlier when $|V|$ is 100 while the total regrets do not drop when $|V|$ is as large as 1000, which indicates that enough events are still available at later time steps when $|V|$ is 1000. Table 5 shows the efficiency results of the algorithms. We can observe that the algorithms consume more time and space when $|V|$ is larger, which is expected. Notice that since UCB takes additional time to calculate an upper confidence bound for each event, the running time of UCB increases more obviously than the other algorithms do when $|V|$ increases. Note that the increase of memory consumption is also due to the increasing size of input data. Again, all the algorithms are quite efficient in both time and space and eGreedy and Exploit are again the fastest.

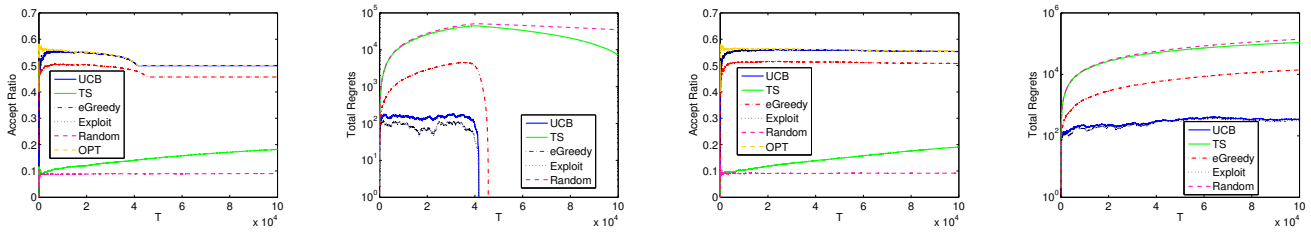
**Effect of dimension.** Figure 4 presents the results when varying $d$. We can observe that all the algorithms perform




(a) Accept Ratio, Uniform (b) Accept Ratio, Power (c) Accept Ratio, Shuffle

(d) Total Regrets, Uniform (e) Total Regrets, Power (f) Total Regrets, Shuffle

Figure 5: Results of FASEA when $\boldsymbol{\theta}$ and features follow different distributions.




(a) Accept Ratio, $\mathcal{N}(100, 100)$ (b) Total Regrets, $\mathcal{N}(100, 100)$ (c) Accept Ratio, $\mathcal{N}(500, 200)$ (d) Total Regrets, $\mathcal{N}(500, 200)$

Figure 6: Results of FASEA when $c_v$ follows $\mathcal{N}(100, 100)$ and $\mathcal{N}(500, 200)$.

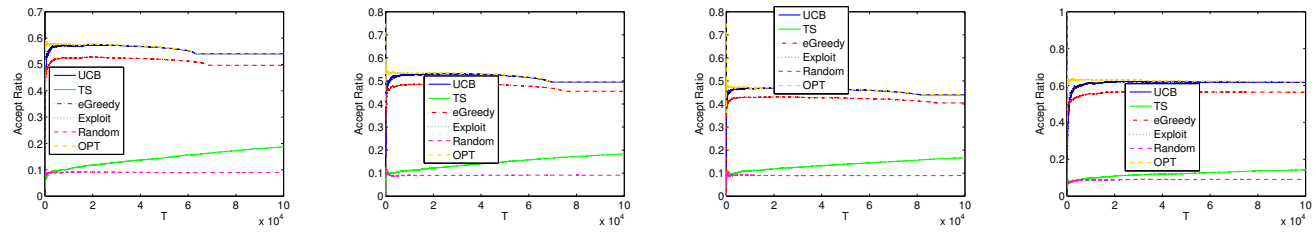
better when $d$ is smaller, especially TS, as TS can finally compete with UCB, eGreedy and Exploit when $d$ is only 1. With this observation, we conjecture that another possible reason that TS performs badly is that TS needs to sample a $d$-dimension vector each time and thus the noise of sampling is enlarged when $d$ increases, which could greatly impact the accuracy of the estimation on the expected rewards of the events. Notice that again due to the limited capacities of events, the accept ratios and total regrets of the algorithms drop suddenly at certain time steps. The efficiency results are presented in Table 6. We can observe that more time and space is consumed when $d$ is larger, which is expected.

**Results following other distributions.** The results when $\boldsymbol{\theta}$ and the features follow different distributions are presented in Figure 5. We can observe similar results. Notice that the accept ratios of the algorithms are generally large and the regrets of the algorithms drop earlier when $\boldsymbol{\theta}$ and features follow Power distribution. The reason is that under Power distribution, the values of the elements of $\boldsymbol{\theta}$ and the features are generally large (closer to 1) and thus the expected rewards of the events are generally large and much more events have feedbacks of 1, which also explains why Random performs well. As the efficiency results are similar, we omit them for brevity.

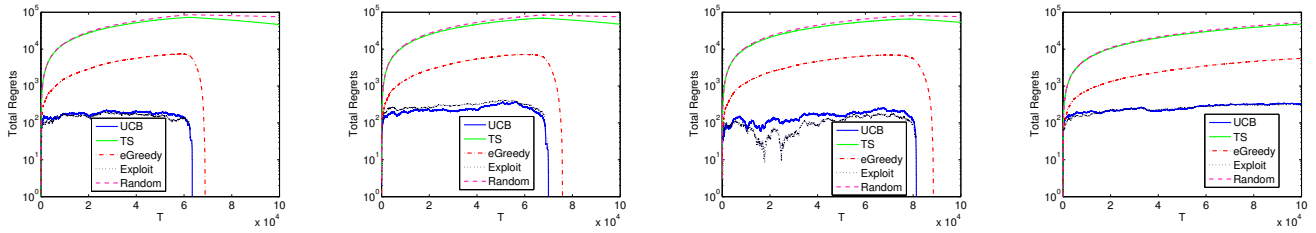
**Effect of $c_v$.** The results when varying $c_v$ are presented in Figure 6. We can observe that when $c_v$ follows $\mathcal{N}(100, 100)$, again as all the events become unavailable at certain time steps, the accept ratios and total regrets of the algorithms drop suddenly. When $c_v$ is generally larger, i.e. following $\mathcal{N}(500, 200)$, there are still quite a few events available even when $t$ is large and thus OPT can still assign events to late-coming users and the accept ratios and total regrets of the algorithms do not drop suddenly. The efficiency results are similar and we omit them for brevity.

**Effect of conflicts.** The results when varying $cr$ are presented in Figure 7, where the conflict ratio ($cr$) of events is 0, 0.5, 0.75 and 1 respectively. Note that when $cr = 0$, no events are conflicting and when $cr = 1$, all the events are conflicting with each other. Again we can see that except Random, TS is the worst while UCB and Exploit are still the best. We can observe that when $cr$ is smaller, the accept ratios and total regrets of the algorithms drop earlier and when $cr = 1$, the accept ratios and total regrets do not drop suddenly. The reason is that when $cr$ is small, more events are available for users arriving at earlier stages as fewer events are conflicting with each other and thus the events will become out of quotas earlier when $cr$ is small. Note that when $cr = 1$, actually only one event can be arranged to one user each time and thus quite a few events are still available when $t$ is large and thus OPT is still arranging events for late-coming users. The efficiency results are similar and we omit them for brevity.

**Effect of algorithm parameters.** Figure 8 presents the results with different $\lambda$ values. Note that Figure 8b shows the results of regret ratios of TS as the difference of total regrets is not obvious. We can observe that the algorithms generally perform better when $\lambda = 1$ or 2. Figure 9a presents the results of UCB with different $\alpha$ values. We can observe

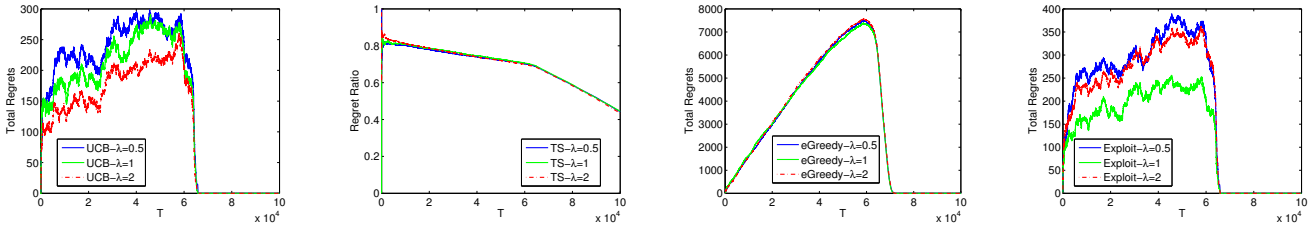



(a) Accept Ratio, $cr = 0$ (b) Accept Ratio, $cr = 0.5$ (c) Accept Ratio, $cr = 0.75$ (d) Accept Ratio, $cr = 1$




(e) Total Regrets, $cr = 0$ (f) Total Regrets, $cr = 0.5$ (g) Total Regrets, $cr = 0.75$ (h) Total Regrets, $cr = 1$

Figure 7: Results of FASEA when conflict ratio is 0, 0.5, 0.75 and 1.

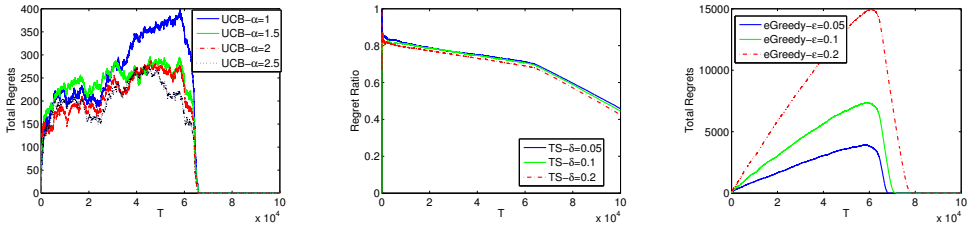



(a) UCB (b) TS (c) eGreedy (d) Exploit

Figure 8: Results of FASEA on effect of $\lambda$.

that UCB generally performs better when $\alpha = 2$. Figure 9b shows the effect of $\delta$ on TS and we can observe that TS performs worse when $\delta = 0.05$. Figure 9c shows the effect of $\epsilon$ on eGreedy. We can observe that generally eGreedy performs when better when $\epsilon$ is smaller, which indicates that the random strategy of eGreedy does not help improve its performance. Efficiency results are omitted for brevity.

**Real dataset.** We then study the performance of the algorithms on real datasets. The results of real dataset for $u_1$ are presented in Figure 10, where we present the (accumulative) accept ratios for the first 1000 rounds for clearer observation of the performance of the algorithms at early stages and present the total regrets for the whole 10000 rounds. Particularly, $c_u$ is 5 or full as explained in Section 5.1. Values of $c_u$ under the "$c_u$ = full" setting are presented in the last row of the table. We can observe that UCB performs the best when $c_u = 5$ and both UCB and Exploit perform well when $c_u$ = full. Similar to synthetic data results, TS does not perform well under both settings. Notice that even "Full Knowledge" cannot achieve accept ratio of 100% when $c_u$ = full due to conflicts of events. Due to limited space, we present the accept ratio results of the other users in Table 7. We can observe that in most cases UCB performs the best and for some users Exploit and eGreedy are better. Notice that Exploit has accept ratio of 0 for $u_8$, $u_{10}$, and $u_{16}$. This is because the events arranged by Exploit initially all have feedbacks of 0 and as the same set of feature vectors are observed in each round in the real dataset, Exploit cannot adjust its estimation of $\boldsymbol{\theta}$ (due to all feedbacks are 0) and thus keeps arranging the same set of events each time, which always have feedbacks of 0. Note that in such cases, UCB and eGreedy are more advantageous – even when all the arranged events have feedbacks of 0, UCB can still update its estimation due to the upper confidence bound and eGreedy will not keep arranging the same set of events with feedbacks of 0 due to its random strategy. Therefore, though both UCB and Exploit perform well in most cases of both synthetic dataset and real dataset, UCB is still more advantageous than Exploit in practice. Notice that we additionally compare with the OnlineGreedy-GEACC algorithm in [39] (named as Online in Table 7), where we use category-sub-categories as tags of events and asked users to select their preferred tags, which are used to calculate the interestingness values as in [39]. Note that since OnlineGreedy-GEACC does not change its strategy based on the observed feedbacks, it keeps making the same arrangement even running in multiple rounds. Therefore, the reported accept ratio of OnlineGreedy-GEACC is single-round rather than accumulative as the accept ratios of other feedback-aware algorithms are. Compared with OnlineGreedy-GEACC that does not utilizes users' feedbacks, we can observe that UCB and eGreedy are generally better, especially when the number of arranged events is limited in each round, i.e. $c_u = 5$. Notice that in some cases there is slight difference between the accept ratio of the best feedback-aware algorithm and that of OnlineGreedy (e.g. $u_4$ and $u_5$ when $c_u$ =full). This is because the accept ratios of feedback-aware algorithms are accumulative over 1000 rounds.




(a) Varying $\alpha$ (b) Varying $\delta$ (c) Varying $\epsilon$

Figure 9: Results of FASEA on effect of $\alpha$, $\delta$ and $\epsilon$.

Table 7: Accept Ratios of Real Dataset of FASEA after 1000 rounds.

| $c_u = 5$ | $u_1$ | $u_2$ | $u_3$ | $u_4$ | $u_5$ | $u_6$ | $u_7$ | $u_8$ | $u_9$ | $u_{10}$ | $u_{11}$ | $u_{12}$ | $u_{13}$ | $u_{14}$ | $u_{15}$ | $u_{16}$ | $u_{17}$ | $u_{18}$ | $u_{19}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UCB | **0.99** | **0.99** | 0.20 | **0.97** | **0.94** | **0.98** | 0.91 | 0.20 | **0.99** | **0.99** | **0.95** | **1.00** | **0.96** | **0.63** | **0.98** | **0.95** | 0.20 | **0.97** | 0.59 |
| TS | 0.27 | 0.62 | 0.19 | 0.25 | 0.24 | 0.47 | 0.34 | 0.17 | 0.49 | 0.29 | 0.28 | 0.48 | 0.44 | 0.27 | 0.22 | 0.16 | 0.19 | 0.26 | 0.36 |
| eGreedy | 0.82 | 0.93 | 0.39 | 0.89 | 0.77 | 0.89 | 0.89 | **0.65** | 0.92 | 0.90 | 0.73 | 0.94 | 0.88 | 0.52 | 0.74 | 0.81 | **0.42** | 0.73 | 0.57 |
| Exploit | 0.96 | 0.98 | **0.40** | 0.91 | 0.80 | 0.86 | **0.96** | 0 | 0.97 | 0 | 0.77 | 0.99 | 0.94 | 0.59 | 0.79 | 0 | 0.20 | 0.78 | **0.60** |
| Random | 0.24 | 0.52 | 0.22 | 0.19 | 0.31 | 0.43 | 0.33 | 0.14 | 0.45 | 0.23 | 0.26 | 0.39 | 0.47 | 0.21 | 0.22 | 0.14 | 0.18 | 0.25 | 0.34 |
| Full Kn. | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Online[39] | 0.8 | 0.8 | 0.8 | 0.6 | 0.6 | 0.4 | 0.6 | 0.4 | 1 | 0.6 | 0.4 | 0.4 | 1 | 0.4 | 0.8 | 1 | 0.6 | 1 | 0.8 |
| $c_u$ =full | $u_1$ | $u_2$ | $u_3$ | $u_4$ | $u_5$ | $u_6$ | $u_7$ | $u_8$ | $u_9$ | $u_{10}$ | $u_{11}$ | $u_{12}$ | $u_{13}$ | $u_{14}$ | $u_{15}$ | $u_{16}$ | $u_{17}$ | $u_{18}$ | $u_{19}$ |
| UCB | **0.50** | **0.80** | **0.63** | **0.69** | 0.65 | **0.63** | 0.56 | 0.14 | **0.86** | **0.75** | 0.66 | **0.78** | 0.71 | **0.39** | **0.51** | **0.93** | 0.22 | **0.83** | 0.60 |
| TS | 0.30 | 0.57 | 0.23 | 0.27 | 0.33 | 0.48 | 0.35 | 0.15 | 0.59 | 0.27 | 0.32 | 0.47 | 0.52 | 0.26 | 0.26 | 0.14 | 0.20 | 0.31 | 0.42 |
| eGreedy | 0.48 | 0.71 | 0.27 | 0.56 | **0.66** | 0.62 | **0.59** | **0.29** | 0.84 | 0.70 | 0.67 | 0.75 | **0.74** | 0.38 | 0.47 | 0.57 | **0.31** | 0.61 | 0.59 |
| Exploit | **0.50** | 0.72 | 0.27 | 0.60 | **0.66** | **0.63** | 0.56 | 0 | 0.77 | 0 | **0.69** | 0.76 | **0.74** | 0.27 | 0.49 | 0 | 0.11 | 0.58 | **0.62** |
| Random | 0.24 | 0.52 | 0.23 | 0.20 | 0.31 | 0.45 | 0.32 | 0.15 | 0.47 | 0.23 | 0.26 | 0.39 | 0.48 | 0.22 | 0.21 | 0.14 | 0.18 | 0.26 | 0.36 |
| Full Kn. | 0.92 | 0.92 | 1 | 1 | 0.93 | 0.91 | 0.94 | 1 | 1 | 0.91 | 1 | 0.95 | 1 | 0.91 | 0.82 | 1 | 0.89 | 1 | 0.94 |
| Online[39] | 0.67 | 0.73 | 0.64 | 0.7 | 0.67 | 0.64 | 0.56 | 0.43 | 0.86 | 0.73 | 0.54 | 0.74 | 0.70 | 0.55 | 0.64 | 0.86 | 0.56 | 0.69 | 0.59 |
| $c_u$ | 12 | 26 | 11 | 10 | 15 | 22 | 16 | 7 | 22 | 11 | 13 | 19 | 23 | 11 | 11 | 7 | 9 | 13 | 17 |

**Further experiment results under basic contextual bandit.** Finally, we further study the performance of the algorithms under the basic contextual bandit model, where capacities of events are unlimited, no events are conflicting and only one event is arranged for one user each time. We denote the results under the basic contextual bandit as "Basic".

Figure 11 shows the results when varying $|V|$ and the other parameters are set to default values. Similar to the results under FASEA, we can observe that TS still performs badly under basic contextual bandit. Notice that since capacities of events are ignored, the regrets of the algorithms do not experience sudden drop as those under FASEA do.

Due to limited space, additional results when varying $d$ and distributions are in the appendix. Again we can observe that TS still performs badly under basic contextual bandit. Similarly, TS performs better when $d$ is smaller.

Finally, Figure 13 shows the results when $\boldsymbol{\theta}$ and the features follow other distributions and similar results can be observed.

**Summary.** We finally summarize our experimental findings.

- In most cases of both synthetic dataset and real dataset, UCB and Exploit perform the best while TS that is reported to perform well under basic multi-armed bandit[9] only performs better than Random. We conjecture that the sampling strategy of TS on $\boldsymbol{\theta}$ brings a lot of noise that affects all the estimates of the expected rewards of events, which could be the possible reason of the poor performance of TS.

- UCB can avoid being trapped in arranging the same set of events with feedbacks of 0 in real world and thus is more advantageous than Exploit.

- All the algorithms are efficient in time and space while eGreedy and Exploit are the most efficient in time.

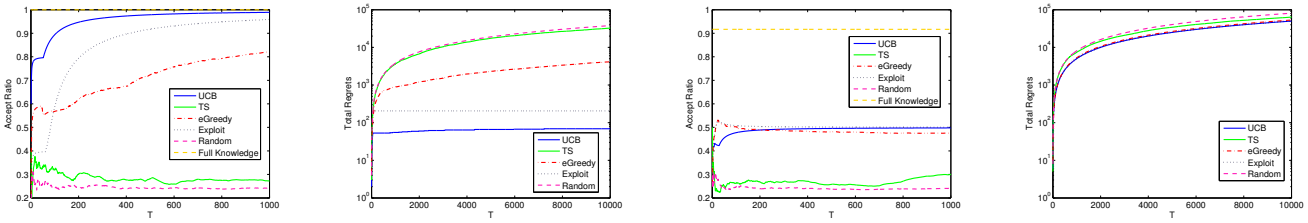
# 6. RELATED WORK

## 6.1 Mining and Managing EBSNs

[30] is the first study to formulate and analyze EBSNs at scale. A large category of works focus on recommendation problems, which utilize learning-based models to train data of EBSNs and then recommend certain items to related parties, e.g. [52][21][35][51][54] [32][10][31][34]. These works mainly focus on the benefits of one particular party, i.e. the ones being recommended to, and lack of a global strategy that satisfies most event organizers and users. Some other works try to find either influential event organizers[14] or influential participants[53]. But still, they only focus on one single side's benefit.

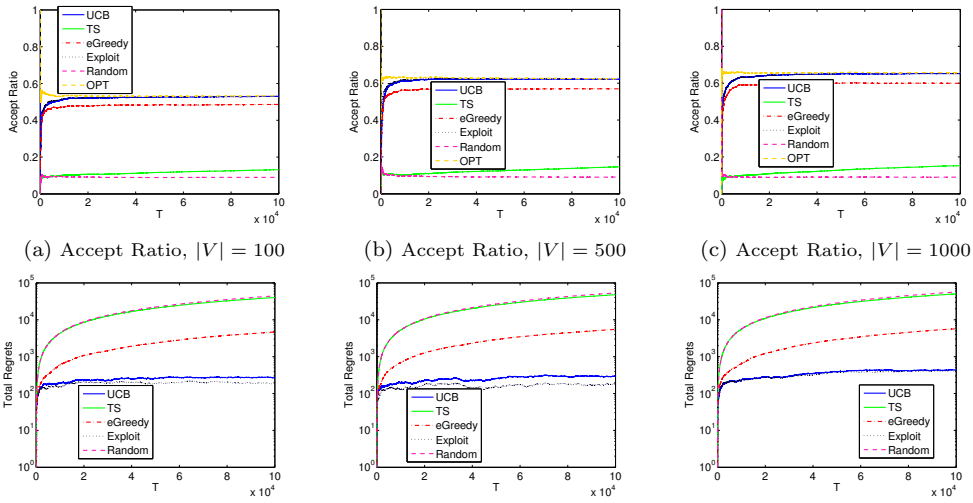
Global arrangement strategies are studied recently. [25] studies the Social Event Organization (SEO) problem that maximizes the sum of the overall satisfaction of users towards the arrangement and the social affinity among the users participating in the same event. [4] studies a graph partition problem based on a game theoretic approach, which can be applied to making event-participant arrangement. [38][37] study global arrangement problems that allow users to attend multiple events and consider conflict constraints and location information. [44] studies another objective function max-min, i.e. maximizing the benefit of the least satisfied user, regarding the event arrangement problem. [17] considers diverse user choices. However, all these works only consider offline scenarios. Recently, [39] extends [38] to online scenarios. But still, none of existing works uses general satisfaction measurement or allows users to give feedbacks on accepting the arrangement or not. In this work, we address online event arrangement that allows users to give feedbacks and learns how to make satisfactory arrangement based on users' feedbacks.

Most existing works on event arrangement are variants of the classic maximum weighted bipartite matching problem




(a) Accept Ratio, $c_u = 5$ (b) Total Regrets, $c_u = 5$ (c) Accept Ratio, $c_u$ = full (d) Total Regrets, $c_u$ = full

Figure 10: Results of real dataset of FASEA ($u_1$).




(a) Accept Ratio, $|V| = 100$ (b) Accept Ratio, $|V| = 500$ (c) Accept Ratio, $|V| = 1000$

(d) Total Regrets, $|V| = 100$ (e) Total Regrets, $|V| = 500$ (f) Total Regrets, $|V| = 1000$

Figure 11: Results when varying $|V|$ under basic contextual bandit.

[7][50][46][45][40] and its online scenario[18][33][20][42][43]. However, these works still do not consider users' feedbacks.

## 6.2 Multi-Armed Bandit

Multi-armed bandit (MAB) problem has been studied for decades due to its wide applications[6]. The main challenge of solving MAB is the exploitation-exploration tradeoff, where one is trying to maximize the total rewards gained by playing arms (exploitation) while having to explore different arms enough (exploration) to avoid being trapped in a local optimum. The most basic bandit problem is the stochastic multi-armed bandit[23], where each arm is associated with an unknown but fixed distribution of rewards. Numerous variants of MAB have been studied. One particular variant related to our event-participant arrangement setting is the stochastic contextual bandit problem. Particularly, [24][26][13][1][22] study the contextual bandit problem with linear payoff. As pointed out by [13] and [1], contextual bandit with linear payoff is also studied under other names, details of which can be referred to [13] and [1]. By playing arms in multiple rounds, a contextual bandit algorithm repeatedly explores or estimates the unknown weights based on the observed rewards and exploits the estimated optimal arm to maximize its total rewards. Another variant related to our setting is called the combinatorial bandit problem, where a subset of arms rather than a single arm are played in each round[3][8][29][15][11][12][28][16][49]. Particularly, [36] studies the two variants simultaneously, a.k.a. the contextual combinatorial bandit problem, while [27] further studies a contextual combinatorial cascading bandit problem. In this paper, we bring contextual combinatorial bandit with linear payoff to event-participant arrangement to make satisfactory arrangement for the users based on their feedbacks.

# 7. CONCLUSION

In this paper, we study a new event-participant arrangement strategy, called the Feedback-Aware Social Event-participant Arrangement (FASEA) problem, for online EBSNs that can learn the satisfaction of users towards the arrangement through their feedbacks on accepting or rejecting the events. We first model the problem as a contextual combinatorial bandit setting, a variant of multi-armed bandit. We then use a Thompson Sampling-based solution and a UCB-based solution to solve the problem. We further present two heuristics, eGreedy and Exploit, based on UCB. We evaluate the algorithms extensively on both real and synthetic datasets. Our experimental results indicate that TS that is reported to work well under basic multi-armed bandit[9] does not perform well under FASEA while UCB is the best in overall.

**Acknowledgements:** We are grateful to anonymous reviewers for their constructive comments on this work. This work is supported in part by the Hong Kong RGC Project 16202215, National Grand Fundamental Research 973 Program of China under Grant 2014CB340303, NSFC Grant No. 61502021, 61328202, 61300031 and 61532004, Microsoft Research Asia Collaborative Grant and NSFC Guang Dong Grant No. U1301253. Yongxin Tong and Lei Chen are the corresponding authors of this paper.

# 8. REFERENCES


[1] S. Agrawal and N. Goyal. Thompson sampling for contextual bandits with linear payoffs. In *ICML'13*, pages 127–135.

[2] S. Agrawal and N. Goyal. Thompson sampling for contextual bandits with linear payoffs. *arXiv preprint arXiv:1209.3352*, 2012.

[3] V. Anantharam, P. Varaiya, and J. Walrand. Asymptotically efficient allocation rules for the multiarmed bandit problem with multiple plays-part i: Iid rewards. *IEEE Transactions on Automatic Control*, 32(11):968–976, 1987.

[4] N. Armenatzoglou, H. Pham, V. Ntranos, D. Papadias, and C. Shahabi. Real-time multi-criteria social graph partitioning: A game theoretic approach. In *SIGMOD'15*, pages 1617–1628.

[5] P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *The Journal of Machine Learning Research*, 3:397–422, 2002.

[6] D. A. Berry and B. Fristedt. *Bandit problems: sequential allocation of experiments (Monographs on statistics and applied probability)*. Springer, 1985.

[7] R. E. Burkard, M. Dell'Amico, and S. Martello. *Assignment Problems, Revised Reprint*. SIAM, 2009.

[8] F. Caro and J. Gallien. Dynamic assortment with demand learning for seasonal consumer goods. *Management Science*, 53(2):276–292, 2007.

[9] O. Chapelle and L. Li. An empirical evaluation of thompson sampling. In *NIPS'11*, pages 2249–2257.

[10] C. C. Chen and Y.-C. Sun. Exploring acquaintances of social network site users for effective social event recommendations. *Information Processing Letters*, 116(3):227–236, 2016.

[11] W. Chen, Y. Wang, and Y. Yuan. Combinatorial multi-armed bandit: General framework and applications. In *ICML'13*, pages 151–159.

[12] W. Chen, Y. Wang, and Y. Yuan. Combinatorial multi-armed bandit and its extension to probabilistically triggered arms. 2014.

[13] W. Chu, L. Li, L. Reyzin, and R. E. Schapire. Contextual bandits with linear payoff functions. In *AISTATS'11*, pages 208–214.

[14] K. Feng, G. Cong, S. S. Bhowmick, and S. Ma. In search of influential event organizers in online social networks. In *SIGMOD'14*, pages 63–74.

[15] Y. Gai, B. Krishnamachari, and R. Jain. Combinatorial network optimization with unknown variables: Multi-armed bandits with linear rewards and individual observations. *IEEE/ACM Transactions on Networking (TON)*, 20(5):1466–1478, 2012.

[16] A. Gopalan, S. Mannor, and Y. Mansour. Thompson sampling for complex online problems. In *ICML'14*, pages 100–108.

[17] J. Huang, Y. Zhou, X. Jia, and H. Sun. A novel social event organization approach for diverse user choices. *The Computer Journal*, 2016.

[18] R. M. Karp, U. V. Vazirani, and V. V. Vazirani. An optimal algorithm for on-line bipartite matching. In *STOC'90*, pages 352–358.

[19] M. G. Kendall. A new measure of rank correlation. *Biometrika*, 1938.

[20] T. Kesselheim, K. Radke, A. Tönnis, and B. Vöcking. An optimal online algorithm for weighted bipartite matching and extensions to combinatorial auctions. In *ESA'13*, pages 589–600.

[21] H. Khrouf and R. Troncy. Hybrid event recommendation using linked data and user diversity. In *RecSys'13*, pages 185–192.

[22] A. Krishnamurthy, A. Agarwal, and M. Dudik. Contextual semibandits via supervised learning oracles. 2015.

[23] T. L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

[24] J. Langford and T. Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In *NIPS'08*, pages 817–824.

[25] K. Li, W. Lu, S. Bhagat, L. V. S. Lakshmanan, and C. Yu. On social event organization. In *KDD'14*, pages 1206–1215.

[26] L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *WWW'10*, pages 661–670.

[27] S. Li, B. Wang, S. Zhang, and W. Chen. Contextual combinatorial cascading bandits. In *ICML'16*, pages 1245–1253.

[28] T. Lin, B. D. Abrahao, R. D. Kleinberg, J. Lui, and W. Chen. Combinatorial partial monitoring game with linear feedback and its applications. In *ICML'14*, pages 901–909.

[29] H. Liu, K. Liu, and Q. Zhao. Logarithmic weak regret of non-bayesian restless multi-armed bandit. In *ICASSP'11*, pages 1968–1971.

[30] X. Liu, Q. He, Y. Tian, W.-C. Lee, J. McPherson, and J. Han. Event-based social networks: linking the online and offline social worlds. In *KDD'12*, pages 1032–1040.

[31] C. Luo, W. Pang, Z. Wang, and C. Lin. Hete-cf: Social-based collaborative filtering recommendation using heterogeneous relations. In *ICDM'14*, pages 917–922.

[32] A. Q. Macedo, L. B. Marinho, and R. L. Santos. Context-aware event recommendation in event-based social networks. In *RecSys'15*, pages 123–130.

[33] A. Mehta. Online matching and ad allocation. *Theoretical Computer Science*, 8(4):265–368, 2012.

[34] T.-A. N. Pham, X. Li, G. Cong, and Z. Zhang. A general graph-based model for recommendation in event-based social networks. In *ICDE'15*, pages 567–578.

[35] Z. Qiao, P. Zhang, Y. Cao, C. Zhou, L. Guo, and B. Fang. Combining heterogenous social and geographical information for event recommendation. In *AAAI'14*, pages 145–151.

[36] L. Qin, S. Chen, and X. Zhu. Contextual combinatorial bandit and its application on diversified online recommendation. In *SDM'14*, pages 461–469.

[37] J. She, Y. Tong, and L. Chen. Utility-aware social event-participant planning. In *SIGMOD'15*, pages 1629–1643.

[38] J. She, Y. Tong, L. Chen, and C. C. Cao. Conflict-aware event-participant arrangement. In *ICDE'15*, pages 735–746.

[39] J. She, Y. Tong, L. Chen, and C. C. Cao. Utility-aware social event-participant planning and its variant for online setting. *Transactions on Knowledge and Data Engineering*, 2016.

[40] Y. Sun, J. Huang, Y. Chen, R. Zhang, and X. Du. Location selection for utility maximization with capacity constraints. In *CIKM'12*, pages 2154–2158.

[41] W. R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 1933.

[42] H. F. Ting and X. Xiang. Near optimal algorithms for online maximum edge-weighted b-matching and two-sided vertex-weighted b-matching. *Theoretical Computer Science*, 607:247–256, 2015.

[43] Y. Tong, J. She, B. Ding, L. Wang, and L. Chen. Online mobile micro-task allocation in spatial crowdsourcing. In *ICDE'16*, pages 49–60.

[44] Y. Tong, J. She, and R. Meng. Bottleneck-aware arrangement over event-based social networks: the max-min approach. *World Wide Web*, 19(6):1151–1177, 2015.

[45] L. H. U, K. Mouratidis, M. L. Yiu, and N. Mamoulis. Optimal matching between spatial datasets under capacity constraints. *ACM Transactions on Database Systems (TODS)*, 35(2):9, 2010.

[46] L. H. U, M. L. Yiu, K. Mouratidis, and N. Mamoulis. Capacity constrained assignment in spatial databases. In *SIGMOD'08*, pages 15–28.

[47] J. Vermorel and M. Mohri. Multi-armed bandit algorithms and empirical evaluation. In *ECML'05*, pages 437–448.

[48] T. J. Walsh, I. Szita, C. Diuk, and M. L. Littman. Exploring compact reinforcement-learning representations with linear regression. In *AUAI'09*, pages 591–598.

[49] Z. Wen, A. Ashkan, H. Eydgahi, and B. Kveton. Efficient learning in large-scale combinatorial semi-bandits. In *ICML'15*.

[50] R. C.-W. Wong, Y. Tao, A. W.-C. Fu, and X. Xiao. On efficient spatial matching. In *VLDB'07*, pages 579–590.

[51] H. Yin, B. Cui, L. Chen, Z. Hu, and C. Zhang. Modeling location-based user rating profiles for personalized recommendation. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 9(3):19, 2015.

[52] H. Yin, Y. Sun, B. Cui, Z. Hu, and L. Chen. Lcars: a location-content-aware recommender system. In *KDD'13*, pages 221–229.

[53] Z. Yu, R. Du, B. Guo, H. Xu, T. Gu, Z. Wang, and D. Zhang. Who should i invite for my party?: combining user preference and influence maximization for social events. In *UbiComp'15*, pages 879–883.

[54] W. Zhang and J. Wang. A collective bayesian poisson factorization model for cold-start local event recommendation. In *KDD'15*, pages 1455–1464.

# APPENDIX

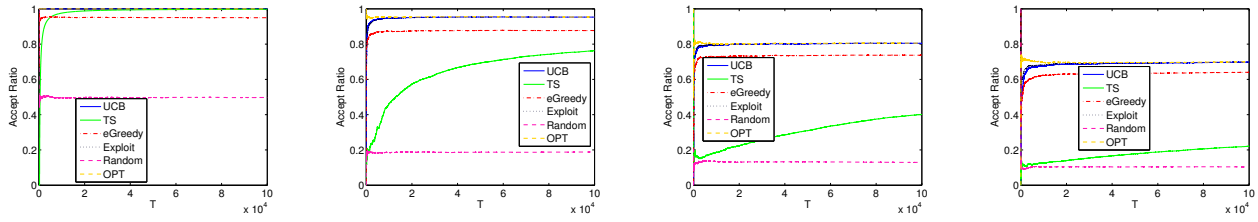
**Proof of Theorem 1**

PROOF. For brevity, in the following proof, we simply regard $A_t$ and $A_t^{OPT}$ as that they do not include events with $\hat{r}_{t,v} \leq 0$. Notice that for each $v \in A_t^{OPT} \setminus A_t$, it is not arranged by Oracle-Greedy due to two cases: (1)it is conflicting with events that are already arranged in $A_t$ before $v$ is visited; (2)$|A_t| = c_u$ before $v$ is visited and thus $v$ is not visited by Oracle-Greedy. Therefore, there must exist an event $v' \in A_t \setminus A_t^{OPT}$ that leads to that $v$ is not included in $A_t$: either due to $v'$ is conflicting with $v$ or $v'$ "occupies the place" of $v$ and $v'$ must be visited before $v$ by Oracle-Greedy. Therefore, $\hat{r}_{t,v'} \geq \hat{r}_{t,v}$. Notice that $v'$ leads to at most $c_u$ such $v$'s that $v \in A_t^{OPT} \setminus A_t$ (due to the first conflicting case). Therefore, $c_u \times \sum_{v \in A_t \setminus A_t^{OPT}} \hat{r}_{t,v} \geq \sum_{v \in A_t^{OPT} \setminus A_t} \hat{r}_{t,v}$. Therefore, we have

$$\begin{aligned}\sum_{v \in A_t} \hat{r}_{t,v} &= \sum_{v \in A_t \cap A_t^{OPT}} \hat{r}_{t,v} + \sum_{v \in A_t \setminus A_t^{OPT}} \hat{r}_{t,v} \\ &\geq \sum_{v \in A_t \cap A_t^{OPT}} \hat{r}_{t,v} + \frac{1}{c_u} \sum_{v \in A_t^{OPT} \setminus A_t} \hat{r}_{t,v} \\ &\geq \frac{1}{c_u} \sum_{v \in A_t^{OPT}} \hat{r}_{t,v} \end{aligned} \tag{3}$$

□

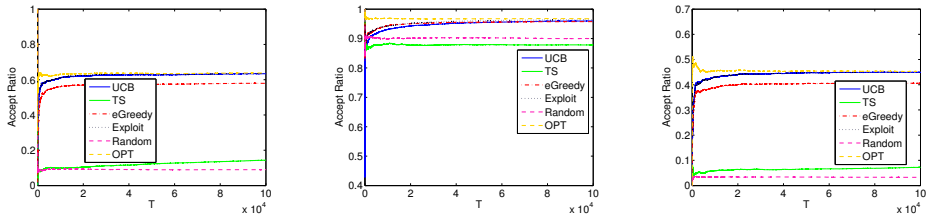
**Additional results under basic contextual bandit.**




(a) Accept Ratio, $d = 1$ (b) Accept Ratio, $d = 5$ (c) Accept Ratio, $d = 10$ (d) Accept Ratio, $d = 15$

(e) Total Regrets, $d = 1$ (f) Total Regrets, $d = 5$ (g) Total Regrets, $d = 10$ (h) Total Regrets, $d = 15$

Figure 12: Results when varying $d$ under basic contextual bandit.




(a) Accept Ratio, Uniform (b) Accept Ratio Power (c) Accept Ratio, Shuffle

(d) Total Regrets, Uniform (e) Total Regrets, Power (f) Total Regrets, Shuffle

Figure 13: Results when $\boldsymbol{\theta}$ and features follow different distributions under basic contextual bandit.