An Online Learning Approach to Sequential User-centric Selection Problems

Junpu Chen, Hong Xie

College of Computer Science, Chongqing University
ironman98@sina.cn, xiehong2018@cqu.edu.cn

Abstract

This paper proposes a new variant of multi-play MAB model, to capture important factors of the sequential user-centric selection problem arising from mobile edge computing, ridesharing applications, etc. In the proposed model, each arm is associated with discrete units of resources, each play is associate with movement costs and multiple plays can pull the same arm simultaneously. To learn the optimal action profile (an action profile prescribes the arm that each play pulls), there are two challenges: *(1) the number of action profiles is large, i.e., M^K, where K and M denote the number of plays and arms respectively; (2) feedbacks on action profiles are not available, but instead feedbacks on some model parameters can be observed.* To address the first challenge, we formulate a completed weighted bipartite graph to capture key factors of the offline decision problem with given model parameters. We identify the correspondence between action profiles and a special class of matchings of the graph. We also identify a dominance structure of this class of matchings. This correspondence and dominance structure enable us to design an algorithm named `OffOptActPrf` to locate the optimal action efficiently. To address the second challenge, we design an `OnLinActPrf` algorithm. We design estimators for model parameters and use these estimators to design a Quasi-UCB index for each action profile. The `OnLinActPrf` uses `OffOptActPrf` as a subroutine to select the action profile with the largest Quasi-UCB index. We conduct extensive experiments to validate the efficiency of `OnLinActPrf`.

1 Introduction

MP-MAB is a sequential decision making model, which was first studied by Anantharam *et al.* (Anantharam, Varaiya, and Walrand 1987a). The canonical MP-MAB model considers one decision maker, who sequentially makes decisions in a finite number of time slots. In each time slot, the decision to make is pulling a subset of arms with a given cardinality and as a result each pulled arm generates a reward according to an unknown probability distribution associated with it. The objective of the decision maker is to maximize the cumulative reward. This MP-MAB model has a number of applications such as web advertising and cognitive radio (Komiyama, Honda, and Nakagawa 2015). Various variants of MP-MAB were proposed (Chen, Wang, and Yuan 2013; Gai, Krishnamachari, and Jain 2012; Kveton et al. 2014; Anantharam, Varaiya, and Walrand 1987a; Zhou and Tomlin 2018) and one can refer to Section 2 for more details.

However, the canonical MP-MAB model and its variants do not provide satisfactory tools for the sequential user-centric selection problems. To illustrate, consider the following user-centric selection problems arise from mobile edge computing systems and ridesharing applications.

Example 1 *(1)* **Sequential user-centric server selection in mobile edge computing.** *There are a finite number of edge servers and users, which can be modeled as arms and plays respectively. A user offloading a task to an edge server corresponds a play pulling an arm, and it is associated with a communication cost. In each time slot, the units of computing resources in an edge server can be modeled as the resource of the arm and it can be stochastic due to resource scheduling. Multiple users can offload tasks to the same server and the mobile edge computing platform assigns resource to users according to certain policies. To avoid conflicts and improve cumulative utility of all users, in each time slot, users can collaborate to select the appropriate edge servers to offload. (2)* **Sequential user(driver)-centric location selection in Ridesharing.** *There are a finite number of pickup locations and drivers, which can be modeled as arms and plays respectively. A driver moving to a pickup location corresponds a play pulling an arm, and it is associated with a cost. The riding requests arrive at each arm in a time slot can be modeled as the resource and it can be stochastic due to uncertainty in the arrival. Multiple drivers can drive to the same location and the platform assigns requests to drivers according to certain policies. To avoid conflicts and improve cumulative rewards of all drivers, drivers can collaborate to select the location to move.*

Example 1 illustrates three common factors of the sequential user-centric selection problems: (1) each arm is associated with stochastic units of resources; (2) each play is associated with different costs in pulling different arms; (3) multiple plays can pull the same arm. We generalize MP-MAB to capture those important factors. To illustrate, consider the following simplified example of our model.

Example 2 *Consider one decision maker and $M = 2$ arms. The decision maker needs to assign $K = 2$ plays in each time slot. Here, each arm can be mapped as* *an edge server and each play can be interpreted as a user in mobile edge computing system. Let* $(a_{t,1}, a_{t,2}) \in \{(1,1),(2,1),(1,2),(2,2)\}$ *denote the action profile in time slot t, where* $a_{t,1}$ *and* $a_{t,2}$ *denote the arm pulled by play 1 and play 2 respectively. Namely, we have* $M^K = 4$ *possible action profiles. In each time slot, both arm 1 and arm 2 have one unit of resource. The reward of a play getting one unit of resource from arm 1 and arm 2 are* 0.33 *and* 0.44 *respectively. Let* $c_{k,m}$ *denote the movement cost of play* $k \in \{1,2\}$ *for pulling arm* $m \in \{1,2\}$*. We consider the following movement cost* $c_{1,1} = 0.1, c_{1,2} = 0.3, c_{2,1} = 0.6, c_{2,2} = 0.2$*. Then the optimal action profile can be* $(1,2)$*, i.e, play 1 pulls arm 1 and play 2 pulls arm 2.*

Example 2 illustrates a simplified case of our model and it shows that the number of all possible action profiles is M^K. Namely, given all model parameters, exhaustive search of optimal action profile is computationally infeasible when the number of arms and plays are large. The first question is: *How to design computationally efficient searching algorithms to locate the optimal action profile?* In practice, the resource and the reward associated with each unit of resource is uncertain, and the uncertainty unknown to the decision maker. The decision maker can collect feedbacks or reward on these uncertainties. The second question is: *How to infer the optimal action profile from these feedbacks?* We answer these three questions and our contributions are:

- We formulate a new variant of MP-MAB model to capture important factors of the sequential user-centric selection problem. In proposed model, each arm is associated with discrete and stochastic units of resource, the reward associated with each unit of resource is also stochastic, and multiple plays can pull the same arm simultaneously (reward of each play is determined by the assignment). Furthermore, each play is associated with a cost in pulling an arm and the cost can be different across different plays or different arms. The objective is to maximize the total profit in a finite number of time slots.
- We formulate a completed bipartite graph, which capture important factors of the offline optimization problem with given model parameters. We show that locating the optimal action profile is equivalent to finding the maximum $\mathcal{U}$-saturated and $\mathcal{V}$-monotone matching (defined in Section 4). We apply the a maximum matching algorithm to search a maximum $\mathcal{U}$-saturated matching. We design an algorithm, which we call `OffOptActPrf`, to locate a maximum $\mathcal{U}$-saturated and $\mathcal{V}$-monotone matching and transform the located matching into an action profile. We prove that this action profile is optimal.
- For the online problem with some unknown model parameters, we design algorithms to estimate model parameters and derive confidence interval sequence for it. Based on the confidence interval sequence, we design a quasi-UCB index for each action profile. We apply the `OffOptActPrf` to located the action profile with the largest quasi-UCB index, resulting in our `OnLinActPrf` algorithm. Extensive experiments show the efficiency of `OnLinActPrf`.

2 Related Work

Modeling Perspective. The canonical MP-MAB model was proposed by Anantharam *et al.* (Anantharam, Varaiya, and Walrand 1987a). Anantharam *et al.* (Anantharam, Varaiya, and Walrand 1987b) extended the canonical from IID reward to Markovian reward. Combinatorial bandits (Cesa-Bianchi and Lugosi 2012; Chen, Wang, and Yuan 2013; Combes et al. 2015b) generalizes the reward function of the canonical MP-MAB from linear to non-linear. Various variants of combinatorial bandits were studied: (1) combinatorial bandits with semi-bandit feedback (Chen, Wang, and Yuan 2013; Chen et al. 2016; Gai, Krishnamachari, and Jain 2012; Combes et al. 2015b), i.e., the reward of each pulled arm is revealed; (2) combinatorial bandits with bandit feedback: (Cesa-Bianchi and Lugosi 2012; Combes et al. 2015b), i.e., only one reward associated with the pulled arm set is revealed; (3) combinatorial bandits with different combinatorial structures, i.e., matroid (Kveton et al. 2014), m-set (Anantharam, Varaiya, and Walrand 1987a), permutation (Gai, Krishnamachari, and Jain 2012), etc. Cascading bandit (Combes et al. 2015a; Kveton et al. 2015; Wen et al. 2017) extends the the reward function of the canonical MP-MAB from linear to a factorization form over the set of selected arms. MP-MAB with a reward function depending on the order of plays is considered in (Lagrée, Vernade, and Cappé 2016; Komiyama, Honda, and Takeda 2017). This reward function is motivated by click model of web applications. MP-MAB with switching cost is considered in (Agrawal et al. 1990; Jun 2004), where only the number of switches is considered. MP-MAB with budget constraint is considered in (Xia et al. 2016; Zhou and Tomlin 2018) and with stochastic number of plays in each round is considered in (Lesage-Landry and Taylor 2017), which is motivated from power system. Different from these models, our model allows assigning multiple plays to the same arm and the reward as well as feedbacks depend on the resource of the arm, while previous models only allows at most one play to an arm. Furthermore, we consider movement cost of plays. Our model is motivated by sequential user-centric selection problem.

Algorithmic perspective. Anantharam *et al.* (Anantharam, Varaiya, and Walrand 1987a) established asymptotic lower bound for the canonical MP-MAB and an algorithm achieving the lower bound asymptotically was proposed. Gai *et al.* (Gai, Krishnamachari, and Jain 2012) proposed a UCB style algorithm for MP-MAB. This UCB style algorithm was improve by Chen *et al.* (Chen, Wang, and Yuan 2013), i.e., they showed that CUCB achieves a smaller regret upper bound. Komiyama *et al.* (Komiyama, Honda, and Nakagawa 2015) showed that Thompson sampling is optimal, i.e., achieving the regret lowerd bound established by Anantharam *et al.* (Anantharam, Varaiya, and Walrand 1987a). A number of sophisticated algorithms were proposed for various variants of the MP-MAB model. For example, algorithms for combinatorial bandits (Combes et al. 2015b), cascading bandits (Combes et al. 2015a), MP-MAB with budget constraint (Zhou and Tomlin 2018), just to name a few. These algorithms were designed to exploit the particular combinatorial structure in the action space or objective function. These algorithms can not be applied to our model, because our model allows assigning multiple plays to the same arm and the reward as well as feedbacks depend on the stochastic resource of the arm, while previous models only allows at most one play to an arm. Furthermore, we consider movement cost of plays, which further complicates the problem. We design a computational efficient algorithm with sub-linear regret to address the challenge.

3 System Model & Problem Formulation

System Model

We consider a sequential user-centric decision problem, where the decision maker needs to make decisions in $T \in \mathbb{N}_+$ time slots sequentially. In each time slot, the decision is to assign $K \in \mathbb{N}_+$ plays denoted by $[K] \triangleq \{1, \dots, K\}$ to the arm set denoted by $[M] \triangleq \{1, \dots, M\}$. In each time slot, each play pulls one arm and different plays can pull the same arm. A play (or an arm) can model a users (or an edge server) in mobile edge computing systems, a driver (or pickup location) in ride sharing applications, etc. The decision maker corresponds to all users collaborate as a whole. Pulling one arm models that a user offload a task to an edge server, a driver moves to a pickup location, etc.

Arm model. The amount of resource and reward associated with arm $m \in [M]$ is characterized by a pair of random vectors $[\boldsymbol{D}_m, \boldsymbol{R}_m]$, where $\boldsymbol{D}_m \triangleq [D_{t,m} : \forall t \in [T]]$ and $\boldsymbol{R}_m \triangleq [R_{t,m} : \forall t \in [T]]$, where $[T] \triangleq \{1, \dots, T\}$. More specifically, $D_{t,m}$ denotes the number of units of resource associated with arm m in time slot t. The $D_{t,m}$ is a random variable capturing uncertainty in resource of an arm, and its support is $[d_{max}] \triangleq \{1, \dots, d_{max}\}$, where $d_{max} \in \mathbb{N}_+$. A unit of resource can model a CPU in an edge server, a ridesharing request arrived at a pickup location, etc. We consider a stationary resource model, i.e., $D_{1,m}, \dots, D_{T,m}$ are independent and identically distributed (IID) random variables for each given m. Furthermore, $D_{t,m}, \forall t, m,$ are independent. Denote the probability mass function of $D_{t,m}$ as $\boldsymbol{p}_m \triangleq [p_{m,d} : \forall d \in [d_{max}]]$:

$$p_{m,d} = \mathbb{P}[D_{t,m} = d], \quad \forall d \in [d_{max}], m \in [M].$$

Denote the probability mass matrix as:

$$\boldsymbol{P} \triangleq [p_{m,d} : \forall d \in [d_{max}], m \in [M]].$$

The probability mass matrix $\boldsymbol{P}$ is unknown to the decision maker. In time slot t, the resource profile denoted by $\boldsymbol{D}_t \triangleq [D_{t,m} : \forall m \in [M]]$ is revealed to the decision maker.

In time slot t, if a play gets one unit of resource from arm m, it receives a reward, which is a sample from the random variable $R_{t,m}$. Given an arm m and time slot t, the reward associated with different units of resource are IID samples of $R_{t,m}$. We consider a stationary reward, i.e., $R_{1,m}, \dots, R_{T,m}$ are IID random variables. Furthermore, $R_{t,m}, \forall t, m,$ are independent. The support of $R_{t,m}$ is denoted by $\mathcal{R} \subseteq \mathbb{R}$. We denote the mean of IID random variables $R_{1,m}, \dots, R_{T,m}$ as: $\mu_m = \mathbb{E}[R_{t,m}], \forall t \in [T], m \in [M]$. Denote the reward mean vector as $\boldsymbol{\mu} \triangleq [\mu_m : \forall m \in [M]]$. The reward mean vector $\boldsymbol{\mu}$ is unknown to the decision maker.

Play model. Let $a_{t,k} \in [M]$ denote the arm pulled by play $k \in [K]$. Denote the action profile of all plays as $\boldsymbol{a}_t \triangleq [a_{t,k} : \forall k \in [K]]$. Denote the number of plays that pull arm m in time slot t as $n_{t,m} \triangleq \sum_{k \in [K]} \mathbb{1}_{\{a_{t,k}=m\}}$. If the number of units of resource associated with arm m exceeds the number of plays that pull this arm, i.e., $D_{t,m} \geq n_{t,m}$, each play receives one unit of resource and $D_{t,m} - n_{t,m}$ units of resource are left. On the contrary, i.e., $D_{t,m} < n_{t,m}$, only $D_{t,m}$ units of resource is allocated to $D_{t,m}$ plays (one unit of resource per play) and $n_{t,m} - D_{t,m}$ plays do not get resource. Denote $\boldsymbol{X}_t \triangleq [X_{t,k} : \forall k \in [K]]$, where $X_{t,k}$ denotes the reward of play k in round t. The reward $X_{t,k}$ is a sample of $R_{t,a_{t,k}}$ if play k receives one unit of resource from arm $a_{t,k}$, otherwise we set $X_{t,k} = \text{null}$ by default.

Let $c_{k,m} \in \mathbb{R}_+ \cup \{+\infty\}$ denote the movement cost of play k pulling arm m. For example, $c_{k,m}$ captures the cost of a driver moving to a pickup location, the communication cost of a user offloading a task to an edge server. Note that $c_{k,m} = +\infty$ models that arm m is not available to play k. Denote the cost profile of play k as $\boldsymbol{c}_k \triangleq [c_{k,m} : \forall m \in [M]]$. The cost profile $\boldsymbol{c}_k$ is known to the decision maker. Let $C(\boldsymbol{a}_t)$ denote cost associated with the action profile $\boldsymbol{a}_t$: $C(\boldsymbol{a}_t) = \sum_{k \in [K]} c_{k,a_{t,k}} = \sum_{k \in [K]} \sum_{m \in [M]} c_{k,m} \mathbb{1}_{\{a_{t,k}=m\}}$.

The Decision Problem

Let $\overline{R}_m(\mathcal{J}, \mu_m, \boldsymbol{p}_m)$ and $U_m(\mathcal{J}, \mu_m, \boldsymbol{p}_m)$ denote total reward and utility respectively, received by a set of $\mathcal{J} \in [K]$ plays for pulling arm m in a time slot. Here we omit the index t due to that we consider stationary stochastic resource and reward. The $\overline{R}_m(\mathcal{J}, \mu_m, \boldsymbol{p}_m)$ can be expressed as

$$\begin{aligned}\overline{R}_m(\mathcal{J}, \mu_m, \boldsymbol{p}_m) &\triangleq \mu_m \mathbb{E}\left[\min\{|\mathcal{J}|, D_{t,m}\}\right] \\ &= \mu_m \left(\sum_{d=1}^{|\mathcal{J}|} d p_{m,d} + \sum_{d=|\mathcal{J}|+1}^{d_{max}} |\mathcal{J}| p_{m,d} \right).\end{aligned}$$

The utility is defined as

$$U_m(\mathcal{J}, \mu_m, \boldsymbol{p}_m) \triangleq \overline{R}_m(\mathcal{J}, \mu_m, \boldsymbol{p}_m) - \sum_{k \in \mathcal{J}} c_{k,m}.$$

Let $U(\boldsymbol{a}_t, \boldsymbol{\mu}, \boldsymbol{P})$ denote the utility of all plays under action profile $\boldsymbol{a}_t$, formally

$$U(\boldsymbol{a}_t, \boldsymbol{\mu}, \boldsymbol{P}) \triangleq \sum_{m \in [M]} U_m(\{k | a_{t,k} = m\}, \mu_m, \boldsymbol{p}_m).$$

Our objective is to select the action profile to maximize the total utility in T time slots, i.e., maximize $\sum_{t=1}^{T} U(\boldsymbol{a}_t, \boldsymbol{\mu}, \boldsymbol{P})$. The optimal action profile is

$$\boldsymbol{a}^* \in \arg\max_{\boldsymbol{a} \in \mathcal{A}} U(\boldsymbol{a}, \boldsymbol{\mu}, \boldsymbol{P}), \tag{1}$$

where $\mathcal{A} \triangleq [M]^K$. Note that $\boldsymbol{a}^*$ is unknown to the decision maker as $\boldsymbol{\mu}$ and $\boldsymbol{P}$ are unknown to the decision maker. Furthermore, even given $\boldsymbol{\mu}$ and $\boldsymbol{P}$, locating $\boldsymbol{a}^*$ is nontrivial, as there are in total $|\mathcal{A}| = M^K$ action profiles.

Denote $\mathcal{H}_t \triangleq (\boldsymbol{D}_1, \boldsymbol{X}_1, \boldsymbol{a}_1 \dots, \boldsymbol{D}_t, \boldsymbol{X}_t, \boldsymbol{a}_t)$ as the historical data of the decision maker up to time slot t. Our objective is to design an algorithm denoted by `OnLinActPrf` to select action profile relying on $\mathcal{H}_{t-1}$ in each time slot t, i.e., $\boldsymbol{a}_t = \texttt{OnLinActPrf}(\mathcal{H}_{t-1})$. We quantify the performance of OnLinActPrf via the regret:

$$R_T \triangleq \sum_{t=1}^{T} \{U(\boldsymbol{a}^*, \boldsymbol{\mu}, \boldsymbol{P}) - \mathbb{E}[U(\boldsymbol{a}_t, \boldsymbol{\mu}, \boldsymbol{P})]\},$$

where $\boldsymbol{a}_t = \texttt{OnLinActPrf}(\mathcal{H}_{t-1})$.

4 The Offline Optimization Problem

Bipartite Graph Representation

To locate the optimal action profile $\boldsymbol{a}^*$ expressed in Equation (1), exhaustive search is computationally expensive, because the total number of action profiles is $|\mathcal{A}| = M^K$. Consider $M = 10$ arms and $K = 100$ plays, we have $|\mathcal{A}| = 10^{100}$. To facilitate the design of computationally efficient algorithms to locate $\boldsymbol{a}^*$, we next formulate a bipartite graph to capture important factors of the offline optimization problem stated in Equation (1).

Bipartite graph. We formulate a complete weighted bipartite graph with node set $\mathcal{U} \cup \mathcal{V}$ and edge set $\mathcal{U} \times \mathcal{V}$, where $\mathcal{U} \cap \mathcal{V} = \emptyset$ and

$$\mathcal{U} \triangleq \{u_1, \dots, u_K\}, \quad \mathcal{V} \triangleq \bigcup_{m \in [M]} \mathcal{V}_m,$$
$$\mathcal{V}_m \triangleq \{v_{m,1}, \dots, v_{m,K}\}.$$

The node $u_k \in \mathcal{U}$ corresponds to play $k \in [K]$. The node set $\mathcal{V}_m$ corresponds to arm $m \in [M]$. The physical meaning of each node $v_{m,k} \in \mathcal{V}_m$, where $k \in [K]$, will be made clear when we introduce weights of edges. To introduce the intuition behind weights of edges, we need the following lemma.

Lemma 1 *Let $\mathcal{J} \subseteq [K]$ denote a subset of plays. We have*

$$\begin{aligned} &U_m(\mathcal{J} \cup \{k\}, \mu_m, \boldsymbol{p}_m) - U_m(\mathcal{J}, \mu_m, \boldsymbol{p}_m) \\ &= \mu_m P_{m,|\mathcal{J}|+1} - c_{k,m}, \end{aligned} \tag{2}$$

where $k \in [K] \setminus \mathcal{J}$, and $P_{m,n} \triangleq \sum_{d=n}^{d_{max}} p_{m,d}$.

Lemma 1 states a closed-form formula for the marginal utility contribution of play k for joining a set of $\mathcal{J}$ plays to pull arm m. From the formula, i.e., Equation (2), one can observe that the marginal utility contribution of play k depends on the cardinality of $\mathcal{J}$, instead of the index of each play in the set $\mathcal{J}$. For the ease of presentation, we therefore denote the marginal utility contribution of play k, i.e,. $U_m(\mathcal{J} \cup \{k\}, \mu_m, \boldsymbol{p}_m) - U_m(\mathcal{J}, \mu_m, \boldsymbol{p}_m)$ by $\Delta_m(k, |\mathcal{J}|)$. Equation (2) implies that

$$\Delta_m(k, |\mathcal{J}|) = \mu_m P_{m,|\mathcal{J}|+1} - c_{k,m}.$$

Note that $U_m(\emptyset, \mu_m, \boldsymbol{p}_m) = 0$. We can then express the utility associated with arm m as

$$U_m(\mathcal{J}, \mu_m, \boldsymbol{p}_m) = \sum_{j=1}^{|\mathcal{J}|} \Delta_m(k_j, j-1), \tag{3}$$

where $k_j \in \mathcal{J}$ denotes the index of a play in set $\mathcal{J}$ and $k_1 < k_2 < \ldots < k_{|\mathcal{J}|}$.

Equation (3) guides us to define weights of edges as follows. Let $W : \mathcal{U} \times \mathcal{V} \to \mathbb{R}$ denote a function, which prescribes a weight for each edge. Formally, the weight of the edge $(u_k, v_{m,j})$ is

$$W(u_k, v_{m,j}) = \Delta_m(k, j-1), \quad \forall k, j \in [K], m \in [M].$$

Namely, the weight $W(u_k, v_{m,j})$ quantifies the marginal utility contribution of play k for pulling arm m, when there are already $j-1$ plays pulling arm m. Formally, we denote the complete weighted bipartite graph as

$$G = (\mathcal{U} \cup \mathcal{V}, \mathcal{U} \times \mathcal{V}, W).$$

Due to page limit, we present illustrating examples in our supplementary file.

Connecting Action Profiles and Matchings

From action profiles to matchings. Let $\mathcal{M} \subseteq \mathcal{U} \times \mathcal{V}$ denote a matching in graph G, which is a set of pairwise non-adjacent edges, i.e., $|\{u|(u,v) \in \mathcal{M}\}| = |\{v|(u,v) \in \mathcal{M}\}| = |\mathcal{M}|$. In the following definition we define a class of matchings, which has connection to the action profiles.

Definition 1 *A matching $\mathcal{M}$ is $\mathcal{U}$-saturated if it satisfies $\{u|(u,v) \in \mathcal{M}\} = \mathcal{U}$. A matching $\mathcal{M}$ is $\mathcal{V}$-monotone if for all $m \in [M]$ it holds that*

$$\big(\{v|(u,v) \in \mathcal{M}\} \cap \mathcal{V}_m\big) \in \big\{\{v_{m,1}, \dots, v_{m,k}\} | k \in [K]\big\}.$$

The property $\mathcal{U}$-saturated means that each play node is an endpoint of one of the edges in the matching $\mathcal{M}$. The $\mathcal{V}$-monotone property means that end points of the matching corresponding to arm m, i.e., $\{v|(u,v) \in \mathcal{M}\} \cap \mathcal{V}_m$, forms an increasing set, i.e., it can be expressed as $\{v_{m,1}, \dots, v_{m,k}\}$, where $k = |\{v|(u,v) \in \mathcal{M}\} \cap \mathcal{V}_m|$ denotes the number of plays who pull arm m. Due to page limit, we present illustrating examples on our supplementary file.

In the following lemma, we states how an action profile can be mapped to a $\mathcal{U}$-saturated and $\mathcal{V}$-monotone matching.

Lemma 2 *Action profile $\boldsymbol{a} \in \mathcal{A}$ can be mapped into a $\mathcal{U}$-saturated and $\mathcal{V}$-monotone matching denoted by $\widetilde{\mathcal{M}}(\boldsymbol{a})$ as:*

$$\widetilde{\mathcal{M}}(\boldsymbol{a}) \triangleq \big\{(u_k, v_{a_k, r_k(\boldsymbol{a})}) | k \in [K]\big\},$$

where $r_k(\boldsymbol{a})$ is defined as:

$$r_k(\boldsymbol{a}) \triangleq |\{j | j \in [K], j \le k, a_j = a_k\}|.$$

Furthermore, it holds that

$$U(\boldsymbol{a}, \boldsymbol{\mu}, \boldsymbol{P}) = \sum_{(u,v) \in \widetilde{\mathcal{M}}(\boldsymbol{a})} W(u, v),$$

and $\widetilde{\mathcal{M}}(\boldsymbol{a}) \neq \widetilde{\mathcal{M}}(\boldsymbol{a}')$ for any $\boldsymbol{a} \neq \boldsymbol{a}'$.

Lemma 2 states that each action profile can be mapped into a $\mathcal{U}$-saturated and $\mathcal{V}$-monotone matching. It also derive a closed-form formula for the $\mathcal{U}$-saturated and $\mathcal{V}$-monotone matching that corresponds to the action profile $\boldsymbol{a}$. In Lemma 2, the $r_k(\boldsymbol{a})$ is the rank of plays k among all plays who pull the same arm as play k, and the rank is calculated according to their indexes in ascending order. Due to page limit, we present illustrating examples on our supplementary file.

From matchings to action profiles. In the following lemma, we states how a $\mathcal{U}$-saturated and $\mathcal{V}$-monotone matching can be mapped into an action profile.

Lemma 3 *Suppose a matching $\mathcal{M}$ is $\mathcal{U}$-saturated and $\mathcal{V}$-monotone. It can be mapped into action profile denoted by $\widetilde{\boldsymbol{a}}(\mathcal{M}) \triangleq (\widetilde{a}_k(\mathcal{M}) : \forall k \in [K])$, where*

$$\widetilde{a}_k(\mathcal{M}) = \underset{m \in [M]}{\arg} \big(\{(u_k, v) | v \in \mathcal{V}_m\} \cap \mathcal{M} \neq \emptyset\big).$$

Furthermore, it holds that

$$U(\widetilde{\boldsymbol{a}}(\mathcal{M}), \boldsymbol{\mu}, \boldsymbol{P}) = \sum_{(u,v) \in \mathcal{M}} W(u, v).$$

Lemma 3 states that each $\mathcal{U}$-saturated and $\mathcal{V}$-monotone matching can be mapped into an action profile. It also derives a closed-form formula for the action profile that corresponds to the $\mathcal{U}$-saturated and $\mathcal{V}$-monotone matching. Lastly, the action profile $\widetilde{\boldsymbol{a}}(\mathcal{M})$ corresponding to matching $\mathcal{M}$ has the nice property that its utility equal the total weights of $\mathcal{M}$. Due to page limit, we present illustrating examples in our supplementary file.

Locating the Optimal Action Profile

Lemma 2 and 3 imply that locating the action profile with the highest total utility can be boiled down to searching the $\mathcal{U}$-saturated and $\mathcal{V}$-monotone matching with the maximum weights. Lemma 2 implies that the number of $\mathcal{U}$-saturated and $\mathcal{V}$-monotone matchings is no less than the total number of action profiles, i.e., M^K. In other words, exhaustive searching over the space of $\mathcal{U}$-saturated and $\mathcal{V}$-monotone matchings is computationally expensive. Furthermore, identify all $\mathcal{U}$-saturated and $\mathcal{V}$-monotone matchings is also not a computationally easy problem. We next state a lemma, which is useful for us to address these challenge.

Lemma 4 *The* $\Delta_m(k, j)$ *satisfies* $\Delta_m(k, j+1) \leq \Delta_m(k, j), \forall j = 0, \ldots, K-1$.

Lemma 4 states that the marginal reward gain function $\Delta_m(k, j)$ is non-increasing in j. In other words, there is a diminishing return effect in the marginal contribution by adding plays to pull an arm. Lemma 4 implies the following characterization of $\mathcal{U}$-saturated matching.

Lemma 5 *Consider a $\mathcal{U}$-saturated $\mathcal{M}$. If it is not $\mathcal{V}$-monotone, there exists a $\mathcal{U}$-saturated and $\mathcal{V}$-monotone matching $\mathcal{M}'$ such that* $\sum_{(u,v) \in \mathcal{M}} W(u, v) \leq \sum_{(u,v) \in \mathcal{M}'} W(u, v)$.

Lemma 5 states that the maximum weight of $\mathcal{U}$-saturated matchings equals the maximum weight of $\mathcal{U}$-saturated and $\mathcal{V}$-monotone matchings. Locating the maximum weight matching from a space of all $\mathcal{U}$-saturated matchings is a well studied problem. The Hungarian algorithm and its variants such as Crouse *et al. (Crouse 2016)* provide computationally efficient algorithms for this problem. One problem is that there are some maximum weighted $\mathcal{U}$-saturated matching may not be $\mathcal{V}$-monotone. Due to page limit, we present illustrating examples on our supplementary file.

Algorithm 1: `OffOptActPrf` $(\boldsymbol{\mu}, \boldsymbol{P}, C)$

```
1:  $P_{m,j} \leftarrow \sum_{d=j}^{d_{max}} p_{m,d}, \forall m \in [M], j \in [1, \ldots, K]$
2:  $\Delta_m(k, j) \leftarrow \mu_m(P_{m,j+1}) - c_{k,m}, \forall k \in [K], m \in [M], j \in [0, \ldots, K-1]$
3:  $W(u_k, v_{m,j}) \leftarrow \Delta_m(k, j-1), \forall k \in [K], m \in [M], j \in [1, \ldots, K]$
4:  $G \leftarrow (\mathcal{U} \cup \mathcal{V}, \mathcal{U} \times \mathcal{V}, W)$
5:  $\mathcal{M} \leftarrow$ MaximumWeightedMatching(G)
6:  $\mathcal{M}' \leftarrow \emptyset$, $K' \leftarrow K$
7:  for $m \in [M]$ do
8:      if $K' == 0$ then
9:          break loop
10:     end if
11:     $\mathcal{E}_m \leftarrow \{(u, v) | (u, v) \in \mathcal{M}, v \in \mathcal{V}_m\}$
12:     $\mathcal{K}_m \leftarrow \{k | k \in [K], (\{u_k\} \times \mathcal{V}_m) \cap \mathcal{E}_m \neq \emptyset\}$
13:     $K' \leftarrow K' - |\mathcal{K}_m|$
14:     $\mathcal{M}' \leftarrow \mathcal{M}' \cup \{(u_{k_{m,l}}, v_{m,l}) | \forall l \in [|\mathcal{K}_m|]\}$
15: end for
16: $\tilde{a}_k(\mathcal{M}') \leftarrow \underset{m \in [M]}{\arg} (\{(u_k, v) | v \in \mathcal{V}_m\} \cap \mathcal{M}' \neq \emptyset), \forall k \in [K]$
17: return $\tilde{\boldsymbol{a}}(\mathcal{M}') = [\tilde{a}_k(\mathcal{M}') : k \in [K]]$
```

We need and algorithm to transform a non $\mathcal{U}$-saturated and $\mathcal{V}$-monotone matching into a $\mathcal{U}$-saturated and $\mathcal{V}$-monotone one. Our idea is as follows. We rearrange the edges with end points associated with the same arm to make them monotone, while keeping the end points corresponds to plays unchanged. We do this rearrangement for each arm respectively. To illustrate, let us consider arm m. We denote a subset of edges in matching $\mathcal{M}$ that with end points associated with arm m as $\mathcal{E}_m = \{(u, v) | (u, v) \in \mathcal{M}, v \in \mathcal{V}_m\}$. Then, we select out the corresponding plays: $\mathcal{K}_m \triangleq \{k | k \in [K], (\{u_k\} \times \mathcal{V}_m) \cap \mathcal{E}_m \neq \emptyset\}$ Finally, we rearrange the edges of $\mathcal{E}_m$ to the following set of edges $\mathcal{M}_m = \{(u_{k_{m,1}}, v_{m,1}), \ldots, (u_{k_{m,|\mathcal{K}_m|}}, v_{m,|\mathcal{K}_m|})\}$, where $k_{m,j} \in \mathcal{K}_m$ and $k_{m,1} < k_{m,2} < \ldots < k_{m,|\mathcal{K}_m|}$. Due to page limit, we present illustrating examples on our supplementary file.

Algorithm 1 combines the above elements to locate the optimal action profile for the offline optimization problem. In particular, Algorithm 1 first construct a complete weighted bipartite graph (line 1-4). Then it locates a $\mathcal{U}$-saturated maximum weighted matching, and this can be achieved by a variant Hungarian algorithm David *et al. (Crouse 2016)* (line 5). Then it transform the matching to be a $\mathcal{U}$-saturated and $\mathcal{V}$-monotone one (line 6-15). Finally, it maps the matching to an action profile (line 16-17).

Due to page limit, we state the optimality and complexity of Algorithm 1 in our supplementary file (Chen and Xie 2021).

5 The Online Learning Problem

Parameter estimation

Note that in time slot $t+1$, the decision maker has access to the historical feedbacks up to time slot t: $\mathcal{H}_t \triangleq (\boldsymbol{D}_1, \boldsymbol{X}_1, \boldsymbol{a}_1 \ldots, \boldsymbol{D}_t, \boldsymbol{X}_t, \boldsymbol{a}_t)$. From the $\mathcal{H}_t$, we estimate the probability mass function via the empirical average:

$$\widehat{p}_{m,d}^{(t)} \triangleq \frac{\sum_{s=1}^{t} \mathbb{1}_{\{D_{s,m}=d\}}}{t}.$$

We denote the probability mass matrix estimated from $\mathcal{H}_t$ as $\widehat{\boldsymbol{P}}^{(t)} \triangleq [\widehat{p}_{m,d}^{(t)} : m \in [M], d \in [d_{max}]]$. We also estimate the reward mean from $\mathcal{H}_t$ via the empirical average:

$$\widehat{\mu}_m^{(t)} \triangleq \frac{\sum_{s=1}^{t} \sum_{k=1}^{K} \mathbb{1}_{\{X_{s,k} \neq \text{null}\}} X_{s,k} \mathbb{1}_{\{a_{s,k}=m\}}}{\sum_{s=1}^{t} \sum_{k=1}^{K} \mathbb{1}_{\{X_{s,k} \neq \text{null}\}} \mathbb{1}_{\{a_{s,k}=m\}}}.$$

We denote the mean vector estimated from $\mathcal{H}_t$ as $\widehat{\boldsymbol{\mu}}^{(t)} \triangleq [\widehat{\mu}_m^{(t)} : \forall m \in [M]]$ The following lemma states a confidence interval sequence for the estimator $\widehat{\mu}_m^{(t)}$.

Lemma 6 *For each* m*, we have* $\mathbb{P}\left[\forall t, \mu_m - \widehat{\mu}_m^{(t)} \geq \epsilon_m^{(t)}\right] \leq \delta$*, where* $\delta \in (0,1)$ *and*

$$\epsilon_m^{(t)} = \begin{cases} \sqrt{2\sigma^2 (t'_m + 1) \ln(\sqrt{t'_m + 1}/\delta)/t'_m}, & \textit{if } t'_m \geq 1, \\ +\infty, & \textit{if } t'_m = 0, \end{cases}$$

$$t'_m = \sum_{s=1}^{t} \sum_{k=1}^{K} \mathbb{1}_{\{X_{s,k} \neq \textit{null}\}} \mathbb{1}_{\{a_{s,k}=m\}}.$$

For simplicity, we denote $\boldsymbol{\epsilon}^{(t)} = [\epsilon_m^{(t)} : \forall m \in [M]]$.

Online Learning Algorithm

Note that we have expert feedback on the stochastic resource or the probability mass matrix, i.e., in each round the number of units of resource in each arm is revealed to the decision maker. However, we only have bandit feedback on reward, i.e., we only obtain the reward of a unit of resource when it is assigned by a play, otherwise, its reward is not revealed. Thus we define a quasi-UCB index for each action profile as follows:

$$\text{Quasi-UCB}_t(\boldsymbol{a}) = \max_{\mu_m \leq \widehat{\mu}_m^{(t)} + \epsilon_m^{(t)}, \forall m \in [M]} U(\boldsymbol{a}, \boldsymbol{\mu}, \widehat{\boldsymbol{P}}^{(t)}).$$

The rationality of Quasi-UCB$(\boldsymbol{a})$ is that: (1) we input the $\widehat{\boldsymbol{P}}$ as we have expert feedback on $\boldsymbol{P}$ and there is no need to do exploration for $\boldsymbol{P}$; (2) we search over the confidence set of mean vector, i.e., $\mu_m \leq \widehat{\mu}_m^{(t)} + \epsilon_m^{(t)}, \forall m \in [M]$, to enable exploration as we only have bandit feedback on rewards. In each time slot we select the action profile with the largest index, i.e.,

$$\boldsymbol{a}_{t+1} \in \arg\max_{\boldsymbol{a} \in \mathcal{A}} \text{Quasi-UCB}_t(\boldsymbol{a}). \tag{4}$$

We next state a lemma to facilitate the computing of $\boldsymbol{a}_{t+1}$ in Equation (4).

Algorithm 2: `OnLinActPrf` ($\mathcal{H}_t$)

```
$\widehat{p}_{m,d}^{(t)} \leftarrow 0, \forall m \in [M], d \in [d_{max}]$
$\widehat{\mu}_m^{(t)} \leftarrow 0, \forall m \in [M], \quad t'_m \leftarrow 0, \forall m \in [M]$
$\epsilon_m^{(t)} \leftarrow +\infty, \forall m \in [M]$
$\widehat{p}_{m,d}^{(t)} \leftarrow \sum_{s=1}^{t} \mathbb{1}_{\{D_{s,m}=d\}}/t, \forall m \in [M], d \in [D]$
$t'_m \leftarrow \sum_{s=1}^{t} \sum_{k=1}^{K} \mathbb{1}_{\{X_{s,k} \neq \text{null}\}} \mathbb{1}_{\{a_{s,k}=m\}}, \forall m \in [M]$
$\widehat{\mu}_m^{(t)} \leftarrow \frac{\sum_{s=1}^{t} \sum_{k=1}^{K} \mathbb{1}_{\{X_{s,k} \neq \text{null}\}} X_{s,k} \mathbb{1}_{\{a_{s,k}=m\}}}{t'_m}, \forall m \in [M]$
for $m \in [M]$ do
    if $t'_m \neq 0$ then
        $\epsilon_m^{(t)} \leftarrow \sqrt{2\sigma^2 (t'_m + 1) \ln(\sqrt{t'_m + 1}/\delta)/t'_m}$
    end if
end for
$\boldsymbol{a}_{t+1} \leftarrow$ OffOptActPrf$(\widehat{\boldsymbol{\mu}}^{(t)} + \boldsymbol{\epsilon}^{(t)}, \widehat{\boldsymbol{P}}^{(t)}, \boldsymbol{C})$
return $\boldsymbol{a}_{t+1}$
```

Lemma 7 *The* Quasi-UCB$_t(\boldsymbol{a})$ *can be derived as* Quasi-UCB$_t(\boldsymbol{a}) = U(\boldsymbol{a}, \widehat{\boldsymbol{\mu}}^{(t)} + \boldsymbol{\epsilon}^{(t)}, \widehat{\boldsymbol{P}}^{(t)})$.

Lemma 7 states a closed form formula for Quasi-UCB$_t(\boldsymbol{a})$. Based on this formula, one can observe that the locating action profile $\boldsymbol{a}_{t+1}$ in Equation (4) is boiled down to:

$$\boldsymbol{a}_{t+1} \in \arg\max_{\boldsymbol{a} \in \mathcal{A}} U(\boldsymbol{a}, \widehat{\boldsymbol{\mu}}^{(t)} + \boldsymbol{\epsilon}^{(t)}, \widehat{\boldsymbol{P}}^{(t)}). \tag{5}$$

From Equation (5), one can observe that Algorithm 1 can be applied to searching the $\boldsymbol{a}_{t+1}$. Summarize the above ideas together, Algorithm 2 outlines an algorithm to selection action profiles relying on $\mathcal{H}_t$.

Due to page limit, we state the regret upper bound of Algorithm 2 in our supplementary file (Chen and Xie 2021).

6 Experiments

Experiment Setting

Parameter setting. We consider a generic sequential user-centric selection problem characterized by $M = 15$ arms and $K = 30$ plays by default. Note that we also vary M and K to evaluate our proposed algorithm. We set the probability mass function as:

$$p_{m,d} = \begin{cases} \alpha d, & \text{if } d \leq \lceil m/2 \rceil, \\ \alpha(m+1-d), & \text{if } \lceil m/2 \rceil < d \leq m, \\ 0, & \text{otherwise} \end{cases}$$

where $\alpha = 1/(\sum_{d=1}^{\lceil m/2 \rceil} d + \sum_{d=\lceil m/2 \rceil + 1}^{m} m + 1 - d)$ is the normalizing factor. This probability function has a normal distribution like shape. Roughly, the expected number of units of resource of an arm increases in its index m, because as the index m increases, more probability masses shift to the larger value of d.

Each arm's rewards are sampled from Guassian distributions. i.e., $\boldsymbol{R}_m \sim N(\mu_m, \sigma^2)$, where $\mu_m \in [1,2]$ and $\sigma > 0$. We consider three cases of the reward mean. (1) **Inc-Shape**: $\mu_m = 1 + m/M$, i.e., reward mean increases in the index of arm m. (2) **Dec-Shape**: $\mu_m = 2 - m/M$, i.e., the reward mean decreases in the index of arm m. (3) **U-Shape**: $\mu_m = 1 + |M/2 - m|/M$, i.e., the reward mean first decrease and then increase in the index of arm m. For the standard deviation σ we consider two cases. (1) **Exact** σ, i.e., input the exact σ to the algorithm. This case reveals the best possible learning speed of the algorithm. (2) **Over specifying** σ, i.e., we input an upper bound of σ, in particular, 2σ and 4σ to the algorithm. This case captures that in practice the σ is unknown, and one needs to over specify it to avoid divergence of the algorithm. We set the movement cost as $c_{k,m} = \eta|(k \mod M) - m|/\max\{K, M\}$, where $\eta \in \mathbb{R}_+$ is a hyper-parameter that controls the scale of the cost.

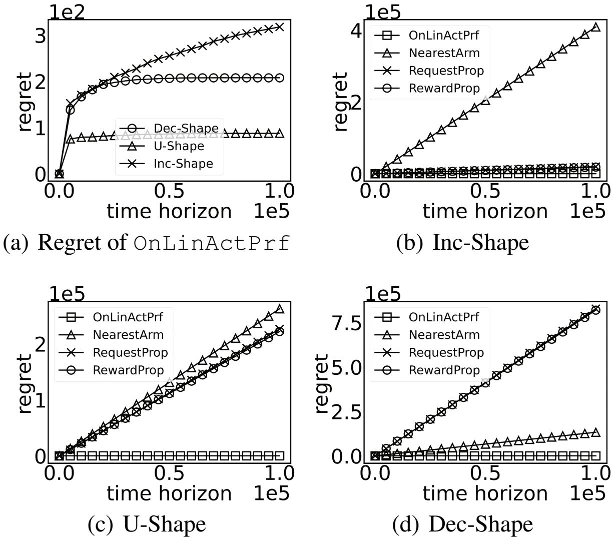
Unless we vary them explicitly, we consider the following default parameters: $T = 10^5$, $\delta = 1/T$, $K = 30$ plays, $M = 15$ arms, $\eta = 1$, the exact σ case with $\sigma = 0.2$ and the U-Shape reward.

Performance metric and baseline. To evaluate the efficiency of `OnLinActPrf`, we compare it with the following three baselines. (1) **RequestProp**, which assigns each play to an arm with a probability proportional to empirical average number of units of resources of an arm, i.e., $\mathbb{P}[a_{t+1,k} = m] = \bar{D}_m^{(t)}/(\bar{D}_1^{(t)} + \cdots + \bar{D}_M^{(t)})$, where $\bar{D}_m^{(t)} = \sum_{d=1}^{d_{max}} d\widehat{p}_{m,d}^{(t)}$. (2) **NearestArm**, which assigns a play to the arm with $a_{t,k} \in \arg\min_{m \in [M]} c_{k,m}$. (3) **RewardProp**, which assigns each play to an arm with a probability proportional to empirical average reward of an arm, i.e., $\mathbb{P}[a_{t+1,k} = m] = \widehat{\mu}_m^{(t)} \bar{D}_m^{(t)}/(\widehat{\mu}_1^{(t)} \bar{D}_1^{(t)} + \cdots + \widehat{\mu}_M^{(t)} \bar{D}_M^{(t)})$. We use the regret as the performance metric. We use Monte Carlo simulation to compute the average regret of each algorithm with 120 rounds of simulation.

Experiment Results

Impact of resource-reward correlation. To study the impact of resource-reward correlation, we fix the probability mass function of resource and three cases of the reward means, i.e., Inc-Shape (positive correlation), Dec-Shape (negative correlation) and U-Shape (weak correlation). Figure 1(a) shows the corresponding regret of `OnLinActPrf` and three comparison baselines. From Figure 1(a), one can observe that the regret curves of `OnLinActPrf` under three types of reward mean first increase sharply, and then become flat. This validates that `OnLinActPrf` has a sub-linear regret. Furthermore, the regret curve of `OnLinActPrf` corresponds to the U-Shape reward mean lie in the bottom. Namely, `OnLinActPrf` has the smallest regret when the correlation between reward and resource is not strong. From Figure 1(b), one can observe that under the Inc-Shape (positive correlation) reward mean, the regret curve corresponds to `OnLinActPrf` lies in the bottom. Namely, `OnLinActPrf` has the smallest regret compared to three baselines. This statement also holds when the reward mean is of U-Shape (weak correlation) and Dec-Shape (negative correlation) as shown in Figure 1(c) and 1(d).

Impact of standard deviation of reward, standard deviation, movement cost, number of arms and number plays. We also conduct experiments to study the impact standard deviation of reward, standard deviation, movement cost, number of arms and number plays. Experiment results show similar conclusion as Figure 1 and they validates the efficiency of our algorithm. Due to page limit, we present them in supplementary file.



(a) Regret of `OnLinActPrf`

(b) Inc-Shape

(c) U-Shape

(d) Dec-Shape

Figure 1: Impact of resource-reward correlation.

7 Conclusion

This paper formulates a new variant of multi-play MAB model to capture important factors of the sequential user-centric selection problem arise from mobile edge computing systems and ride sharing applications, etc. We design `OffOptActPrf` algorithm to locate the optimal action profile given model parameters. The `OffOptActPrf` serves as a subroutine of our `OnLinActPrf`, which estimates the optimal action profile from historical feedbacks when some model parameters are unknown. The core idea of `OffOptActPrf` is that we formulate a completed weighted bipartite graph to capture factors of the offline decision problem. We identify a dominance structure of $\mathcal{U}$-saturated matchings. This correspondence and dominance structure enables us to design an algorithm named `OffOptActPrf` to locate the optimal action. We design estimators for model parameters and use these estimators to design a Quasi-UCB index for each action profile. The `OnLinActPrf` selects the action profile with the largest Quasi-UCB index based on the `OffOptActPrf`. We conduct extensive experiments to validate the efficiency of `OnLinActPrf`.

Acknowledgment

The work of Hong Xie was supported by National Natural Science Foundation of China (61902042), Chongqing Natural Science Foundation (cstc2020jcyj-msxmX0652) and Chongqing Talents: Exceptional Young Talents Project (cstc2021ycjh-bgzxm0195) and the Fundamental Research Funds for the Central Universities (2020CDJ-LHZZ-057).

References

Agrawal, R.; Hegde, M.; Teneketzis, D.; et al. 1990. Multi-armed bandit problems with multiple plays and switching cost. *Stochastics and Stochastic reports*, 29(4): 437–459.

Anantharam, V.; Varaiya, P.; and Walrand, J. 1987a. 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.

Anantharam, V.; Varaiya, P.; and Walrand, J. 1987b. Asymptotically efficient allocation rules for the multiarmed bandit problem with multiple plays-Part II: Markovian rewards. *IEEE Transactions on Automatic Control*, 32(11): 977–982.

Cesa-Bianchi, N.; and Lugosi, G. 2012. Combinatorial bandits. *Journal of Computer and System Sciences*, 78(5): 1404–1422.

Chen, J.; and Xie, H. 2021. *A Robust Algorithm to Unifying Offline Causal Inference and Online Multi-armed Bandit Learning. https://1drv.ms/b/s!AkqQNKuLPUbEi1V7tOKuSgaofsaC?e=jGf07Q.*

Chen, W.; Wang, Y.; and Yuan, Y. 2013. Combinatorial multi-armed bandit: General framework and applications. In *International Conference on Machine Learning*, 151–159. PMLR.

Chen, W.; Wang, Y.; Yuan, Y.; and Wang, Q. 2016. Combinatorial multi-armed bandit and its extension to probabilistically triggered arms. *The Journal of Machine Learning Research*, 17(1): 1746–1778.

Combes, R.; Magureanu, S.; Proutiere, A.; and Laroche, C. 2015a. Learning to rank: Regret lower bounds and efficient algorithms. In *Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, 231–244.

Combes, R.; Talebi, S.; Proutière, A.; and Lelarge, M. 2015b. Combinatorial Bandits Revisited. In *NIPS 2015-Twenty-ninth Conference on Neural Information Processing Systems*.

Crouse, D. F. 2016. On implementing 2D rectangular assignment algorithms. *IEEE Transactions on Aerospace and Electronic Systems*, 52(4): 1679–1696.

Gai, Y.; Krishnamachari, B.; and Jain, R. 2012. Combinatorial Network Optimization With Unknown Variables: Multi-Armed Bandits With Linear Rewards and Individual Observations. *IEEE/ACM Transactions on Networking*, 20(5): 1466–1478.

Jun, T. 2004. A survey on the bandit problem with switching costs. *de Economist*, 152(4): 513–541.

Komiyama, J.; Honda, J.; and Nakagawa, H. 2015. Optimal regret analysis of Thompson sampling in stochastic multi-armed bandit problem with multiple plays. In *International Conference on Machine Learning*, 1152–1161. PMLR.

Komiyama, J.; Honda, J.; and Takeda, A. 2017. Position-based multiple-play bandit problem with unknown position bias. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 5005–5015.

Kveton, B.; Wen, Z.; Ashkan, A.; Eydgahi, H.; and Eriksson, B. 2014. Matroid bandits: fast combinatorial optimization with learning. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, 420–429.

Kveton, B.; Wen, Z.; Ashkan, A.; and Szepesvári, C. 2015. Combinatorial cascading bandits. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 1*, 1450–1458.

Lagrée, P.; Vernade, C.; and Cappé, O. 2016. Multiple-play bandits in the position-based model. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 1605–1613.

Lesage-Landry, A.; and Taylor, J. A. 2017. The multi-armed bandit with stochastic plays. *IEEE Transactions on Automatic Control*, 63(7): 2280–2286.

Wen, Z.; Kveton, B.; Valko, M.; and Vaswani, S. 2017. Online influence maximization under independent cascade model with semi-bandit feedback. In *Neural Information Processing Systems*, 1–24.

Xia, Y.; Qin, T.; Ma, W.; Yu, N.; and Liu, T.-Y. 2016. Budgeted Multi-Armed Bandits with Multiple Plays. In *IJCAI*, 2210–2216.

Zhou, D.; and Tomlin, C. 2018. Budget-constrained multi-armed bandits with multiple plays. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.