Algorithms for Adversarial Bandit Problems with Multiple Plays

Taishi Uchiya, Atsuyoshi Nakamura, and Mineichi Kudo

Graduate School of Information Science and Technology, Hokkaido University
Kita 14, Nishi 9, Kita-ku, Sapporo 060-0814, Hokkaido, Japan
{uchiya, atsu, mine}@main.ist.hokudai.ac.jp

Abstract. Adversarial bandit problems studied by Auer et al. [4] are multi-armed bandit problems in which no stochastic assumption is made on the nature of the process generating the rewards for actions. In this paper, we extend their theories to the case where $k(\geq 1)$ distinct actions are selected at each time step. As algorithms to solve our problem, we analyze an extension of **Exp3** [4] and an application of a bandit online linear optimization algorithm [1] in addition to two existing algorithms (**Exp3**,**ComBand** [5]) in terms of time and space efficiency and the regret for the best fixed action set. The extension of **Exp3**, called **Exp3.M**, performs best with respect to both the measures: it runs in $O(K(\log k + 1))$ time and $O(K)$ space, and suffers at most $O(\sqrt{kTK \log(K/k)})$ regret, where K is the number of possible actions and T is the number of iterations. The upper bound of the regret we proved for **Exp3.M** is an extension of that proved by Auer et al. for **Exp3**.

Keywords: multi-armed bandit problem, adversarial bandit problem, online learning

1 Introduction

Multi-armed bandit problems are a kind of sequential resource allocation problems in which one resource is allocated to one action among several alternative actions at each time step. Each allocation yields a reward and the objective of the problem is the maximization of the total reward. The problems are known as paradigms of the trade-off between exploration (for better future rewards) and exploitation (for high current rewards). These problems have been becoming more and more important in this Internet age because several problems such as server selection in network, Internet ad placement and market pricing at e-commerce sites [8] can all be formulated as multi-armed bandit problems.

Vast studies have been done so far on these problems [10], and the majority of them assumes that the bandit processes are stochastic. However, adversarial bandit problems studied by Auer et al. [4] make no stochastic assumption on the nature of the process generating the rewards for the actions, and they also have been becoming popular recently. There have been several extensions on this line such as the on-line shortest path problem [7], bandit online linear optimization [1] and combinatorial bandits [5].

In this paper, we study adversarial bandit problems extended in one direction, namely, those problems with multiple plays. In this extension, k resources are allocated at each time step. The multiple play setting is practically useful for such problems as multiple ad placement on one web page, which is studied as the problem of multi-impressions in [11]. As for stochastic bandit problems, there are already several studies along this direction [2, 3, 13, 14], but, to the best of our knowledge, only the study made so far in the adversarial setting is combinatorial bandits of Cesa-Bianchi and Lugosi [5]. They considered a general bandit problem in which a player select one binary vector from a fixed set $\mathcal{S} \subseteq \{0,1\}^K$ at each time step. The k-sized subset version of their algorithm **ComBand** is just an algorithm for the multiple play setting. The regret for the best fixed k-sized action subset is $O(k^{\frac{3}{2}}\sqrt{TK \ln K})$, where T is the number of iterations and K is the number of possible actions. The time and space complexities of this algorithm are $O(kK^3)$ and $O(K^3)$, respectively. Note that here we only consider the case where selected k actions must be distinct, different from the studies in [12, 15].

Time and space complexities of **ComBand** can be improved a little by algorithm **BOLOM**, which is made by applying the bandit online linear optimization algorithm [1] to our problem. **BOLOM** runs in $O(K^3)$ time per iteration and $O(K^2)$ space regardless of the value of k. The regret of **BOLOM** for the best fixed action set is bounded by $O(kK^{\frac{3}{2}}\sqrt{T \log T})$, which is worse than **ComBand**. The best algorithm for the multiple play setting is algorithm **Exp3.M**, which is an extension of **Exp3** [4]. The action space is the same as that of **Exp3**, namely, **Exp3.M** keeps just K weights. Using the efficient k-combination selection procedure developed by Gandhi et al. [6], which can select a set S of k distinct actions so as to satisfy that each action i is selected with given probability p_i, **Exp3.M** runs in $O(K(\log k + 1))$ time per iteration and $O(K)$ space, and achieves an upper bound $O(\sqrt{kTK \log(K/k)})$ on the regret for the best fixed action set. Note that this upper bound is an extension of that proved by Auer et al. [4] because they coincide when $k = 1$. We also show that a lower bound of the regret on the problem is $\Omega(((K-k)/K)^2\sqrt{KT})$, which is also an extension of that proved in [4] on the original problem.

2 Problem Setting

An *adversarial bandit problem* [4] is specified by a set of possible player's actions $[K](\doteq \{1, 2, \ldots, K\})$ and an assignment of rewards $\boldsymbol{x}(t)=(x_1(t), x_2(t), \ldots, x_K(t))$ at time step $t = 1, 2, \ldots, T$, where $x_i(t) \in [0, 1]$ denotes the reward obtained by the player. In *multiple play* setting, the player selects a set of k distinct actions $S(t) \in \mathbf{C}([K], k)$ at each time step t, and after that, the player gets rewards $x_i(t)$ for $i \in S(t)$, where $\mathbf{C}([K], k) = \{S \subseteq [K] : |S| = k\}$, namely, the set of all subsets of size k in $[K]$. Throughout the paper, we use $|S|$ as the number of elements in S for any set S. Note that we also use notation $\mathbf{C}(K, k)$ which denotes the number $|\mathbf{C}\left([K],k\right)|$ $\left(= \begin{pmatrix} K \\ k \end{pmatrix}\right)$. All the information the player can obtain at time step t is only the rewards for the actions the player has selected at that time step.

The *cumulative reward* G_A of a player algorithm A is defined as

$$G_A \doteq \sum_{t=1}^{T} \sum_{i \in S(t)} x_i(t)$$

if the algorithm A chooses an action sequence $S(1), S(2), \ldots, S(T)$. The problem is to design a player algorithm A that maximizes its cumulative reward under the condition that an adversary who knows the strategy of A decides an assignment of rewards.

We measure the performance of algorithm A by the regret of A for the best fixed set of actions (that is called weakly regret in [4]), which is defined by $G_{\textbf{max-k}} - G_A$, where

$$G_{\textbf{max-k}} \doteq \max_{S \in \mathbf{C}([K],k)} \sum_{t=1}^{T} \sum_{i \in S} x_i(t)$$

is the cumulative reward for the best fixed set of k distinct actions.

3 Previous Works

First, an adversarial multi-armed bandit problem with multiple plays can be solved by using **Exp3** developed by Auer et. al [4]. Regarding a set of k actions as one action makes **Exp3** applicable to our multiple play setting[1]. However, this arises a problem that the size of action space becomes large, namely, $\mathbf{C}\left(K,k\right) = \Omega(K^k)$. As a result, the regret upper bound obtained by Corollary 3.2 in [4] becomes

$$G_{\textbf{max-k}} - E[G_{\textbf{Exp3}}] \leq 2k\sqrt{e-1}\sqrt{T\mathbf{C}\left(K,k\right)\ln \mathbf{C}\left(K,k\right)} \leq 2.63\sqrt{k^3 T K^k \ln K},$$

and the time and space complexities becomes $\Omega(K^k)$.

The regret upper bound can be improved significantly even using $\mathbf{C}\left(K,k\right)$ weights like **Exp3**. Algorithm **ComBand** shown in Fig. 1 is the k-sized subset version of the algorithm developed by Cesa-Bianchi and Lugosi [5], which is just the algorithm that solves our problem as it is. Like **Exp3**, **ComBand** has one weight for each k-sized subset. At each time step, it randomly selects one k-sized subset according to a distribution calculated by the weights, and updates the weights depending on the obtained reward. The randomized selection method of **ComBand** is the same as that of **Exp3**, but **ComBand** uses more sophisticated method of weight update than **Exp3**. **ComBand** calculates a $K \times K$ matrix P_t that is defined as $E_{p_U}[\mathbf{1}_U \mathbf{1}_U^\top]$, where $\mathbf{1}_U$ is a K-dimensional vector whose ith

[1] A reward for a set of k actions must be divided by k in the application.

```
**ComBand**(The k-sized subset version of the algorithm for Combinatorial Bandits)
Parameters:    $\gamma \in (0,1]$
Initialization: $w_U(1) = 1$ for $U \in \mathbf{C}([K],k)$
**For each** $t = 1, 2, \ldots, T$,
    1. For $U \in \mathbf{C}([K],k)$ set
           $p_U(t) = (1-\gamma)\frac{w_U(t)}{\sum_{U' \in \mathbf{C}([K],k)} w_{U'}(t)} + \frac{\gamma}{\mathbf{C}(K,k)}.$
    2. Choose $S(t)$ randomly according to the distribution $p_U(t)$ for $U \in \mathbf{C}([K],k)$.
    3. Receive a reward $\sum_{i \in S(t)} x_i(t) \in [0,k]$.
    4. Set
           $P_t = \sum_{U \in \mathbf{C}([K],k)} p_U \mathbf{1}_U \mathbf{1}_U^\top$ and
           $\hat{\mathbf{l}}(t) = \left(k - \sum_{i \in S(t)} x_i(t)\right) P_t^+ \mathbf{1}_{S(t)}.$
    5. For $U \in \mathbf{C}([K],k)$ set
           $w_U(t+1) = w_U(t) \exp\left(-\frac{\gamma(K-k)\sum_{i \in U} \hat{l}_i(t)}{kK(K-1)}\right).$
```

Fig. 1. Pseudocode of algorithm **ComBand** [5] (k-sized subset version).

component is 1 if $i \in U$ and 0 otherwise, and $\top$ denotes transpose. Then, the K-dimensional vector of pseudo-losses $\hat{\mathbf{l}}(t)$ is calculated as $P_t^+ \mathbf{1}_{S(t)}$ multiplied by the loss $k - \sum_{i \in S(t)} x_i(t)$, where P_t^+ is the pseudo-inverse of P_t. For each k-sized subset U, the weight $w_U(t+1)$ is $w_U(t)$ multiplied by $exp\left(-\eta \sum_{i \in U} \hat{l}_i(t)\right)$, where $\hat{l}_i(t)$ is the ith component of $\hat{\mathbf{l}}(t)$ and $\eta = \gamma(K-k)/kK(K-1)$. By Theorem 1 and Proposition 15 in [5], for $k \le K/2$, we obtain

$$G_{\mathbf{max\text{-}k}} - E[G_{\mathbf{ComBand}}] \le \left(2 + \frac{K-1}{K-k}\right) k\sqrt{TK \ln \mathbf{C}(K,k)} \le 4\sqrt{k^3 TK \ln K}.$$

As commented in [5], there is an implementation of **ComBand** in which the time and space complexity is also significantly improved. In the efficient implementation, one weight w_i for each original action i is enough because weight w_U for a k-sized subset can be represented by $\prod_{i \in U} w_i$ and Step 5 in **ComBand** can be replaced with

$$5'. \text{ For } i \in [K] \text{ set } w_i(t+1) = w_i(t) \exp\left(-\frac{\gamma(K-k)\hat{l}_i(t)}{kK(K-1)}\right).$$

Since each k-sized subset can be represented by a path from the source to

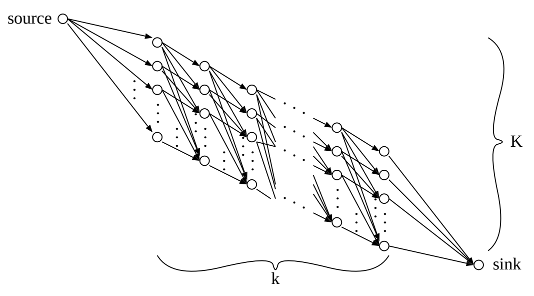

Fig. 2. The directed graph G such that a path in G has one-to-one correspondence to a k-sized subset of $[K]$.

the sink in G of Fig. 2, by dynamic programming technique of Takimoto and Warmuth [16], $S(t)$ can be chosen without calculating $p_U(t)$ for all $U \in \mathbf{C}\left([K], k\right)$ in Step 2, and P_t can be calculated without taking a sum over all $U \in \mathbf{C}\left([K], k\right)$ in Step 4. (See also the proof of Theorem 3 in [7].) From the fact that the number of nodes in G is $O(kK)$ and the weights of the edges incoming into the same vertex are the same, $S(t)$ can be chosen in $O(kK)$ time and $O(kK)$ space, and P_t can be calculated in $O(kK^3)$ time and $O(K^3)$ space. Note that we do not have to keep edge information in memory because of its regularity. The pseudo-inverse P_t^+ can be calculated in $O(K^3)$ time and $O(K^2)$ space. In total, the time and space complexities of **ComBand** is $O(kK^3)$ and $O(K^3)$, respectively.

4 Application of Bandit Online Linear Optimization

A more efficient algorithm for large k can be obtained by applying an algorithm developed in the context of bandit online linear optimization [1]. In the bandit online linear optimization problem, the space corresponding to the action space is a compact closed convex set $\mathcal{K}$ in $\mathbb{R}^n$. At each time step t, a player chooses $\boldsymbol{q}_t \in \mathcal{K}$, then an adversary returns $\boldsymbol{x}_t^\top \boldsymbol{q}_t$ to the player. The player's goal is to minimize his regret defined as

$$\max_{\boldsymbol{q}^* \in \mathcal{K}} \sum_{i=1}^{T} \boldsymbol{x}_t^\top \boldsymbol{q}^* - \sum_{i=1}^{T} \boldsymbol{x}_t^\top \boldsymbol{q}_t.$$

To consider our multiple play setting bandit problem in this framework, set $\mathcal{K}$ to the convex hull of $\left\{\mathbf{1}_U \in \mathbb{R}^K : U \in \mathbf{C}\left([K], k\right)\right\}$, where $\mathbf{1}_U$ is a vector whose ith component is 1 if $i \in U$ and 0 otherwise. The following proposition holds.

Proposition 1. *The convex hull of* $\left\{\mathbf{1}_U \in \mathbb{R}^K : U \in \mathbf{C}\left([K], k\right)\right\}$ *is equal to* $\left\{\boldsymbol{q} : 0 \le q_i \le 1 \text{ for } i = 1, 2, \ldots, K, \sum_{i=1}^{K} q_i = k\right\}$.

Proof. Let $A = \left\{\boldsymbol{q} : 0 \le q_i \le 1 \;\text{ for }\; i = 1, 2, \ldots, K, \sum_{i=1}^{K} q_i = k\right\}$ and $B = \left\{\mathbf{1}_U \in \mathbb{R}^K : U \in \mathbf{C}\left([K], k\right)\right\}$. Then, B is a compact convex subset of $\mathbb{R}^K$ and the set of its extreme points is A. Thus, the convex hull of A is B by Krein-Milman theorem [9]. □

By the above proposition, any $\boldsymbol{q} \in \mathcal{K}$ satisfies $\sum_{i=1}^{K} q_i = k$, so $\mathcal{K}$ can be regarded as a $(K-1)$-dimensional subspace $\mathcal{K}_{K-1}$ defined by

$$\mathcal{K}_{K-1} \doteq \left\{\boldsymbol{q} : 0 \le q_i \le 1 \;\text{ for }\; i = 1, 2, \ldots, K-1, 0 \le k - \sum_{i=1}^{K-1} q_i \le 1\right\}.$$

Therefore, a linear optimization problem under the constraint of range $\mathcal{K}_{K-1}$ can be solved as the unconstrained linear optimization problem with the θ-self concordant barrier

$$R(\boldsymbol{q}) \doteq -\ln\left[\left\{\prod_{i=1}^{K-1} q_i(1-q_i)\right\}\left(k - \sum_{i=1}^{K-1} q_i\right)\left(1 - k + \sum_{i=1}^{K-1} q_i\right)\right]$$

on $\mathcal{K}_{K-1}$, where $\theta = 2K$ for this barrier R. By applying Algorithm 1 in [1] to our multiple play setting bandit problem, we can obtain algorithm **BOLOM** shown in Fig. 3. In the application, there are two things we have to take care of. One is the difference of the problem settings: in linear optimization setting, the player can select any element $\boldsymbol{p}$ in $\mathcal{K}$, but in the multiple play setting, the player must select a set U from $\mathbf{C}\left([K], k\right)$, which corresponds to the original set of $\mathbf{C}\left(K, k\right)$ vectors before taking its convex hull.

This can be overcome by selecting a set $U \in \mathbf{C}\left([K], k\right)$ at random so as to satisfy the condition that each action i is selected with probability p_i. This selection guarantees that $E(\mathbf{1}_U) = \boldsymbol{p}$ holds, and the bound of Theorem 1 in [1] is still valid for the algorithm of the above modification by their Proposition 1 in Sec. 7 in [1]. Function **DepRound** [6] used in our algorithm is an efficient algorithm that makes such a selection, whose details are described in Sec. 7.

The other problem is that the reward $\boldsymbol{x}(t)^\top \boldsymbol{p}(t)$ expectedly received by the player at each time step t cannot be expressed linearly in $\mathcal{K}_{K-1}$: it is expressed as

$$(x_1(t) - x_K(t), \ldots, x_{K-1}(t) - x_K(t))\begin{pmatrix} p_1(t) \\ \vdots \\ p_{K-1}(t)\end{pmatrix} + kx_K(t),$$

that is, it is expressed as a combination of a linear part with a new reward vector $(x_1(t) - x_K(t), \cdots, x_{K-1}(t) - x_K(t))$ and a bias $kx_K(t)$.
Let $\boldsymbol{y}(t) = \frac{1}{k}\left(x_1(t) - x_K(t), \cdots, x_{K-1}(t) - x_K(t)\right)^\top$. Then,

$$\begin{aligned} & E_{S(t)}\frac{1}{k}\boldsymbol{x}(t)^\top \mathbf{1}_{S(t)} = \frac{1}{k}\boldsymbol{x}(t)^\top E_{S(t)}\mathbf{1}_{S(t)} = \frac{1}{k}\boldsymbol{x}(t)^\top \boldsymbol{p}(t) \\ & = \boldsymbol{y}(t)^\top \begin{pmatrix} p_1(t) \\ \vdots \\ p_{K-1}(t)\end{pmatrix} + x_K(t) = \boldsymbol{y}(t)^\top(\boldsymbol{q}(t) + \varepsilon_t \lambda^{-\frac{1}{2}}\boldsymbol{e}_{i_t}) + x_K(t) \end{aligned}$$

```
BOLOM(Bandit Online Linear Optimization with Multiple plays)
Parameters:   $\eta > 0$
Initialization:
  $R(\boldsymbol{q}) = -\ln \left[ \left\{ \prod_{i=1}^{K-1} q_i(1-q_i) \right\} \left( k - \sum_{i=1}^{k-1} q_i \right) \left( 1 - k + \sum_{i=1}^{K-1} q_i \right) \right]$  for $\boldsymbol{q} \in \mathcal{K}_{K-1}$,
  $q_i(1) = \frac{k}{K}$  for $i = 1, 2, \ldots, K-1$
For each $t = 1, 2, \ldots, T$
  1. Calculate the set of eigenvectors $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_{K-1}\}$ and
     eigenvalues $\{\lambda_1, \ldots, \lambda_{K-1}\}$ of $\nabla^2 R(\boldsymbol{q}(t))$.
  2. Choose $i_t$ uniformly at random from $\{1, \ldots, K-1\}$ and
     $\varepsilon_t = \pm 1$ with probability 1/2.
  3. Set
       $p_j(t) = q_j(t) + \varepsilon_t \lambda^{-\frac{1}{2}} e_{i_t,j}$  for $j = 1, 2, \ldots, K-1$,
       $p_K(t) = k - \sum_{i=1}^{K-1} p_i(t)$.
  4. Set
       $S(t) = \mathbf{DepRound}(k, (p_1(t), p_2(t), \ldots, p_K(t)))$.
  5. Receive rewards $x_i(t) \in [0,1]$ for $i \in S(t)$.
  6. Set
       $\hat{\boldsymbol{y}}(t) = (K-1) \left( \frac{1}{k} \sum_{i \in S(t)} x_i(t) \right) \varepsilon_t \lambda^{\frac{1}{2}} \boldsymbol{e}_{i_t}$,
       $\boldsymbol{q}(t+1) = \arg\min_{\boldsymbol{q} \in \mathcal{K}_{K-1}} \left[ -\eta \sum_{s=1}^{t} \hat{\boldsymbol{y}}(s)^\top \boldsymbol{q} + R(\boldsymbol{q}) \right]$.
```

Fig. 3. Pseudocode of algorithm **BOLOM**.

holds. Thus, by virtue of random choice of ε_t,

$$\begin{aligned}
E_{\varepsilon_t} E_{S(t)} \hat{\boldsymbol{y}}(t) &= \frac{1}{2}(K-1)(\boldsymbol{y}(t)^\top (\boldsymbol{q}(t) + \lambda^{-\frac{1}{2}} \boldsymbol{e}_{i_t}) + x_K(t)) \lambda_{i_t}^{\frac{1}{2}} \boldsymbol{e}_{i_t} \\
&\quad - \frac{1}{2}(K-1)(\boldsymbol{y}(t)^\top (\boldsymbol{q}(t) - \lambda^{-\frac{1}{2}} \boldsymbol{e}_{i_t}) + x_K(t)) \lambda_{i_t}^{\frac{1}{2}} \boldsymbol{e}_{i_t} \\
&= (K-1)(\boldsymbol{y}(t)^\top \boldsymbol{e}_{i_t}) \boldsymbol{e}_{i_t}
\end{aligned}$$

holds, so $E\hat{\boldsymbol{y}}(t) = E_{i_t} E_{\varepsilon_t} E_{S(t)} \hat{\boldsymbol{y}}(t) = \boldsymbol{y}(t)$ is still implied.

Therefore, by Theorem 1 in [1], we obtain the following theorem.

Theorem 1. *Set $\eta = \sqrt{2K \log T} / (4(K-1)\sqrt{T})$. Then*

$$G_{\textbf{max-k}} - E[G_{\textbf{BOLOM}}] \leq 16k(K-1)\sqrt{2KT \log T} + \sqrt{(K-1)T}$$

holds for any $T > 16K \log T$ and for any assignment of rewards.

The regret upper bound of **BOLOM** is worse than that of **ComBand** in both T and K. Algorithm **BOLOM** runs in $O(K^3)$ time per iteration and needs $O(K^2)$ space. Thus, **BOLOM** is more efficient than **ComBand**.

5 Multiple Play Version of Exp3

In Sec. 3, we saw that a direct application of **Exp3** for action space $\mathbf{C}([K], k)$ have to deal with $\Omega(K^k)$ weights, which caused a large regret upper bound and large time and space complexities. Can we apply algorithm **Exp3** for the original action space $[K]$ to solve the multiple play setting of a bandit problem? The answer is yes, and we develop such an algorithm in this section.

As **Exp3** does, our algorithm selects action i with probability $p_i(t)$ and estimates $x_i(t)$ by $\hat{x}_i(t) = x_i(t)/p_i(t)$ when action i is selected and by $\hat{x}_i(t) = 0$ otherwise. This calculation guarantees our algorithm to satisfy $E[\hat{x}_i(t)] = x_i(t)$ if action i is selected randomly with probability $p_i(t)$. Then, the problem is reduced to how to select k distinct actions under the condition that each action i is selected randomly with probability $p_i(t)$. This can be done in $O(K)$ time per iteration by using function **DepRound** [6]. (See Sec. 7.) Note that $\sum_{i=1}^K p_i(t)$ must be k in this problem setting because the expected total number of selection is $\sum_{i=1}^K p_i(t)$.

However, a new problem arises using this selection: probability $p_i(t)$ possibly becomes more than 1 if it is set to a value proportional to weight $w_i(t)$ that is more than $\frac{1}{k}\sum_{j=1}^K w_j(t)$. Our countermeasure for this situation is to let $p_i(t)$ linearly depend on modified weight $w'_i(t)$ that is made from $w_i(t)$ by cutting off at some threshold α_t.

Our extension of algorithm **Exp3** for multiple play setting is algorithm **Exp3.M** shown in Fig. 4. If all $w_j(t)$ are less than $\left(\frac{1}{k} - \frac{\gamma}{K}\right)\sum_{i=1}^K w_i(t)/(1-\gamma)$, which is checked at Step 1, $p_j(t)$ calculated at Step 3 is less than 1 for all $i = 1, 2, \ldots, K$ without weight modification. In this case, $S_0(t)$ is set to $\emptyset$ at Step 1. Otherwise, threshold α_t is set to an appropriate value, and all the actions i with $w_i(t) \geq \alpha_t$ are classified into $S_0(t)$. The temporal weight $w'_i(t)$ is set to α_t for $i \in S_0(t)$ and $w_i(t)$ for $i \notin S_0(t)$. Since

$$\frac{w'_i(t)}{\sum_{j=1}^K w'_j(t)} = \frac{\alpha_t}{\sum_{w_j(t)\geq\alpha_t}\alpha_t + \sum_{w_j(t)<\alpha_t} w_j(t)}$$

holds for all $i \in S_0(t)$, $p_i(t)$ is set to 1 for all $i \in S_0(t)$ in Step 3 if α_t is decided as in Step 1, namely, α_t is decided so as to satisfy

$$\frac{\alpha_t}{\sum_{w_i(t)\geq\alpha_t}\alpha_t + \sum_{w_i(t)<\alpha_t} w_i(t)} = \left(\frac{1}{k} - \frac{\gamma}{K}\right)/(1-\gamma).$$

Note that $S_0(t) \subseteq S(t)$ since $p_i(t) = 1$ for all $i \in S_0(t)$. Another point of our algorithm is that weights $w_i(t)$ are not updated for $i \in S_0(t)$ in Step 6, namely,

```
**Exp3.M**(The extended version of **Exp3** for bandit problems with Multiple plays)
Parameters:    $\gamma \in (0,1]$
Initialization:  $w_i(1) = 1$ for $i = 1, 2, ..., K$
**For each** $t = 1, 2, ...$
  1. **if** $\arg\max_{j\in[K]} w_j(t) \geq \left(\frac{1}{k} - \frac{\gamma}{K}\right) \sum_{i=1}^{K} w_i(t)/(1-\gamma)$ **then**
       Decide $\alpha_t$ so as to satisfy
         $$\frac{\alpha_t}{\sum_{w_i(t)\geq\alpha_t} \alpha_t + \sum_{w_i(t)<\alpha_t} w_i(t)} = \left(\frac{1}{k} - \frac{\gamma}{K}\right)/(1-\gamma).$$
       Set $S_0(t) = \{i : w_i(t) \geq \alpha_t\}$ and $w'_i(t) = \alpha_t$ for $i \in S_0(t)$.
     **else**
       Set $S_0(t) = \emptyset$.
  2. Set
         $$w'_i(t) = w_i(t) \text{ for } i \in \{1, 2, ..., K\} - S_0(t).$$
  3. Set
         $$p_i(t) = k\left((1-\gamma)\frac{w'_i(t)}{\sum_{j=1}^{K} w'_j(t)} + \frac{\gamma}{K}\right) \text{ for } i = 1, 2, ..., K.$$
  4. Set
         $$S(t) = \textbf{DepRound}(k, (p_1, p_2, \ldots, p_n)).$$
  5. Receive rewards $x_i(t) \in [0,1]$ for $i \in S(t)$.
  6. For $i = 1, 2, ..., K$ set
         $$\hat{x}_i(t) = \begin{cases} x_i(t)/p_i(t) & \text{if } i \in S(t), \\ 0 & \text{otherwise.} \end{cases}$$
         $$w_i(t+1) = \begin{cases} w_i(t)\exp(k\gamma\hat{x}_i(t)/K) & \text{if } i \notin S_0(t), \\ w_i(t) & \text{otherwise.} \end{cases}$$
```

Fig. 4. Pseudocode of algorithm **Exp3.M**

$w_i(t+1) = w_i(t)$ for actions i with relatively too large weights. The bottleneck of the algorithm is the calculation of α_t, which can be calculated in $O(K(\log k + 1))$ time by finding the largest k weights. Therefore, **Exp3.M** runs in $O(K(\log k + 1))$ time at each time step and needs $O(K)$ space.

The following theorem is an extension of Theorem 3.1 in [4].

Theorem 2. *For any $K > 0$ and for any $\gamma \in (0,1]$,*

$$G_{\textit{\textbf{max-k}}} - E[G_{\textit{\textbf{Exp3.M}}}] \leq (e-1)\gamma G_{\textit{\textbf{max-k}}} + \frac{K}{\gamma} \ln \frac{K}{k}$$

holds for any assignment of rewards and for any $T > 0$.

Proof. Let W_t, W'_t denote $\sum_{i=1}^K w_i(t), \sum_{i=1}^K w'_i(t)$, respectively. Then, for any $t = 1, 2, ..., T$,

$$\begin{aligned}
\frac{W_{t+1}}{W_t} &= \sum_{i\in[K]-S_0(t)} \frac{w_i(t+1)}{W_t} + \sum_{i\in S_0(t)} \frac{w_i(t+1)}{W_t} \\
&= \sum_{i\in[K]-S_0(t)} \frac{w_i(t)}{W_t} \exp\left(\frac{k\gamma}{K}\hat{x}_i(t)\right) + \sum_{i\in S_0(t)} \frac{w_i(t)}{W_t} \\
&\le \sum_{i\in[K]-S_0(t)} \frac{w_i(t)}{W_t}\left[1+\frac{k\gamma}{K}\hat{x}_i(t)+(e-2)\left(\frac{k\gamma}{K}\hat{x}_i(t)\right)^2\right]+\sum_{i\in S_0(t)}\frac{w_i(t)}{W_t} \qquad (1)\\
&= 1 + \frac{W'_t}{W_t} \sum_{i\in[K]-S_0(t)} \frac{w_i(t)}{W'_t}\left[\frac{k\gamma}{K}\hat{x}_i(t) + (e-2)\left(\frac{k\gamma}{K}\hat{x}_i(t)\right)^2\right] \\
&= 1 + \frac{W'_t}{W_t} \sum_{i\in[K]-S_0(t)} \frac{\frac{p_i(t)}{k} - \frac{\gamma}{K}}{1-\gamma}\left[\frac{k\gamma}{K}\hat{x}_i(t) + (e-2)\left(\frac{k\gamma}{K}\hat{x}_i(t)\right)^2\right] \\
&\le 1+\frac{\gamma}{K(1-\gamma)}\sum_{i\in[K]-S_0(t)} p_i(t)\hat{x}_i(t) + \frac{(e-2)k\gamma^2}{K^2(1-\gamma)}\sum_{i\in[K]-S_0(t)} p_i(t)\hat{x}_i(t)^2 \qquad (2)\\
&\le 1 + \frac{\gamma}{K(1-\gamma)} \sum_{i\in S(t)-S_0(t)} x_i(t) + \frac{(e-2)k\gamma^2}{K^2(1-\gamma)} \sum_{i\in[K]} \hat{x}_i(t). \qquad (3)
\end{aligned}$$

Inequality (1) uses $e^a \le 1 + a + a^2$ for $a \le 1$, inequality (2) holds because $W'_t/W_t \le 1$, and inequality (3) uses the fact that $p_i(t)\hat{x}_i(t) = x_i(t) \le 1$ for $i \in S(t)$ and $p_i(t)\hat{x}_i(t) = 0$ for $i \notin S(t)$. Since $1 + x \le e^x$, we have

$$\ln \frac{W_{t+1}}{W_t} \le \frac{\gamma}{K(1-\gamma)} \sum_{i\in S(t)-S_0(t)} x_i(t) + \frac{(e-2)k\gamma^2}{K^2(1-\gamma)} \sum_{i\in[K]} \hat{x}_i(t).$$

By summing over t, we obtain

$$\ln \frac{W_{T+1}}{W_1} \le \frac{\gamma}{K(1-\gamma)} \sum_{t=1}^T \sum_{i\in S(t)-S_0(t)} x_i(t) + \frac{(e-2)k\gamma^2}{K^2(1-\gamma)} \sum_{t=1}^T \sum_{i\in[K]} \hat{x}_i(t). \qquad (4)$$

On the other hand, for the set $A^* \subset [K]$ of k elements with the maximum total reward $\sum_{j\in A}\sum_{t=1}^T x_j(t)$ among all subsets A containing k elements,

$$\begin{aligned}
\ln \frac{W_{T+1}}{W_1} \ge \ln \frac{\sum_{j\in A^*} w_j(T+1)}{W_1} &\ge \frac{\sum_{j\in A^*} \ln w_j(T+1)}{k} + \ln \frac{k}{K} \qquad (5)\\
&= \frac{\gamma}{K} \sum_{j\in A^*} \sum_{t:j\notin S_0(t)} \hat{x}_j(t) + \ln \frac{k}{K}. \qquad (6)
\end{aligned}$$

The second inequality in (5) uses the fact that

$$\sum_{j\in A^*} w_j(T+1) \ge k(\prod_{j\in A^*} w_j(T+1))^{1/k}$$

and equation (6) uses $w_j(T+1) = \exp((k\gamma/K)\sum_{t:j\notin S_0(t)} \hat{x}_j(t))$.

From (4) and (6), we get

$$\sum_{j\in A^*}\sum_{t:j\notin S_0(t)} \hat{x}_j(t) + \frac{K}{\gamma}\ln\frac{k}{K} \leq \frac{1}{(1-\gamma)}\sum_{t=1}^{T}\sum_{i\in S(t)-S_0(t)} x_i(t) + \frac{(e-2)k\gamma}{K(1-\gamma)}\sum_{t=1}^{T}\sum_{i\in[K]}\hat{x}_i(t).$$

Since $\sum_{j\in A^*}\sum_{t:j\in S_0(t)} x_j(t) \leq \frac{1}{1-\gamma}\sum_{t=1}^{T}\sum_{i\in S_0(t)} x_i(t)$ trivially holds, we have

$$\begin{aligned}&\sum_{j\in A^*}\sum_{t:j\notin S_0(t)} \hat{x}_j(t) + \sum_{j\in A^*}\sum_{t:j\in S_0(t)} x_j(t) + \frac{K}{\gamma}\ln\frac{k}{K}\\ &\leq \frac{1}{(1-\gamma)}G_{\mathbf{Exp3.M}} + \frac{(e-2)k\gamma}{K(1-\gamma)}\sum_{t=1}^{T}\sum_{i\in[K]}\hat{x}_i(t).\end{aligned}$$

Taking expectation of both sides of this inequality, we obtain

$$G_{\mathbf{max\text{-}k}} + \frac{K}{\gamma}\ln\frac{k}{K} \leq \frac{1}{(1-\gamma)}E[G_{\mathbf{Exp3.M}}] + \frac{(e-2)k\gamma}{K(1-\gamma)}\sum_{t=1}^{T}\sum_{i\in[K]} x_i(t)$$

because equation $E[\hat{x}_i(t)|S(1), S(2), \ldots, S(i-1)] = x_i(t)$ holds from the fact that **DepRound** selects action i with probability $p_i(t)$.

From the fact that

$$\sum_{t=1}^{T}\sum_{i=1}^{K} x_i(t) \leq \frac{K}{k}G_{\mathbf{max\text{-}k}},$$

we obtain the inequality in the statement of the theorem. □

The following corollary can be obtained by an appropriate choice of parameter γ. The proof is the same as that of Corollary 3.2 in [4].

Corollary 1. *Set $\gamma = min\left\{1, \sqrt{K\ln(K/k)/((e-1)kT)}\right\}$. Then*

$$G_{\textit{max-k}} - E[G_{\textit{Exp3.M}}] \leq 2\sqrt{(e-1)}\sqrt{kTK\ln\frac{K}{k}} \leq 2.63\sqrt{kTK\ln\frac{K}{k}}$$

holds for any $T > 0$ and for any assignment of rewards.

6 Lower Bounds on the Regret

Auer et al. showed a lower bound $\Omega(\sqrt{KT})$ on the regret of any player for adversarial bandit problem with single play ($k = 1$). Their theorem can be extended easily to the multiple play setting.

Theorem 3. *For any number of actions K, for any time horizon T and for any $k \in [K]$, there exists a distribution over the assignment of rewards such that*

$$E[G_{\textbf{max-k}} - G_A] \geq \min\left\{ \frac{1}{5}\left(\frac{K-k}{K}\right)^2 \sqrt{KT}, \frac{K-k}{8K}kT \right\},$$

holds for any algorithm A.

Proof. This theorem can be proved by modifying the proof of Theorem 5.1 [4] a little. The reward assignment by the distribution whose existence is insisted by the theorem is made as follows. First, select a set of k actions I according to uniform distribution over $\mathbf{C}([K], k)$. For each $(i,t) \in [K] \times [T]$, independently assign 1 to reward $x_i(t)$ with probability $\frac{1}{2} + \varepsilon$ when $i \in I$ and with probability $\frac{1}{2}$ otherwise, where ε is a small constant value belonging to $(0, \frac{1}{2})$. Value 0 is assigned to $x_i(t)$ with the rest of the probability. Note that this distribution over reward assignment coincides with the one used to prove Theorem 5.1 in [4] when $k = 1$. Let $E_*[\cdot]$ denote expectation of some random variable with respect to this distribution. Then, we can prove

$$E_*[G_{\textbf{max-k}} - G_A] \geq k\varepsilon \left(T - \frac{kT}{K} - 2\sqrt{\ln\frac{4}{3}}kT\sqrt{\frac{T}{K}}\varepsilon \right), \tag{7}$$

when $0 \leq \varepsilon \leq 1/4$. By choosing $\varepsilon = (1/4)\min\{(K-k)/(k\sqrt{\ln(4/3)}\sqrt{KT}), 1\}$, the lower bound of this theorem can be obtained.

The proof of Inequality (7) can be done by evaluating a random variable $N_{\mathbf{i}}$ which denotes the number of times action $i \in \mathbf{i}$ $(\in \mathbf{C}([K], k))$ is chosen, namely, $N_{\mathbf{i}} = \sum_{t=1}^{T} |S(t) \cap \mathbf{i}|$. Let $E_{\mathbf{i}}[\cdot]$ denote conditional expectation $E_*[\cdot | I = \mathbf{i}]$. Then,

$$E_*[G_A] = \frac{kT}{2} + \frac{\varepsilon}{\mathbf{C}(K,k)} \sum_{\mathbf{i} \in \mathbf{C}([K],k)} E_{\mathbf{i}}[N_{\mathbf{i}}] \tag{8}$$

holds. We use the following lemma, which is a straightforward extension of Lemma A.1 in [4].

Lemma 1. *Let $f : \{0,1\}^{kT} \to [0, M]$ be any function defined on reward sequences $\mathbf{r}$. Then for any set of actions $\mathbf{i} \in \mathbf{C}([K], k)$,*

$$E_{\mathbf{i}}[f(\mathbf{r})] \leq E_{\textit{unif}}[f(\mathbf{r})] + \frac{M}{2}\sqrt{-E_{\textit{unif}}[N_{\mathbf{i}}]\ln(1-4\varepsilon^2)},$$

where $E_{\textit{unif}}[\cdot]$ is the uniform distribution over assignment of rewards.

By Lemma 1, we obtain

$$E_{\mathbf{i}}[N_{\mathbf{i}}] \leq E_{\mathbf{unif}}[N_{\mathbf{i}}] + \frac{kT}{2}\sqrt{-E_{\mathbf{unif}}[N_{\mathbf{i}}]\ln(1-4\varepsilon^2)}. \tag{9}$$

Combining (8) and (9), and using Jensen's Inequality and the fact that $\sum_{\mathbf{i}\in\mathbf{C}([K],k)} E_{\mathbf{unif}}[N_{\mathbf{i}}] = \mathbf{C}\,(K-1,k-1)\,kT$, we can prove

$$E_*[G_A] \le \frac{kT}{2} + k\varepsilon\left(\frac{kT}{K} + \frac{kT}{2}\sqrt{-\frac{T}{K}\ln(1-4\varepsilon^2)}\right).$$

Inequality (7) can be derived from this inequality using the fact that $E_*[G_{\mathbf{max\text{-}k}}] \ge kT(1/2+\varepsilon)$ and the inequality $-\ln(1-x) \le (4\ln(4/3))x$ for $x \in [0,1/4]$. □

Remark 1. When $k = K$, there exists only one strategy, which means 0-regret for all algorithms. In this case, our upper bound shown in Corollary 1 and our lower bound shown in Theorem 3 become 0.

7 Efficient k-Combination Selection

In this section, we describe how to select efficiently a set of k distinct actions from $[K]$ so as to satisfy that each action i is selected with probability p_i for any given distribution $(p_1, p_2, \dots, p_K)$ with $0 \le p_i \le 1$ for $i = 1, 2, \dots, K$ and $\sum_{i=1}^K p_i = k$. How to realize this selection plays an important role for efficient implementations of both algorithms **BOLOM** and **Exp3.M**. One solution is to select a set S according to a distribution q_S for $S \in \mathbf{C}\,([K],k)$ after solving the following problem.

Problem 1. For a given $(p_1, p_2, \dots, p_K) \in [0,1]^K$ with $\sum_{i=1}^K p_i = k$, find q_S for $S \in \mathbf{C}\,([K],k)$ satisfying

$$\begin{aligned} &\sum_{i\in S} q_S = p_i(t) \quad \text{for all} \quad i = 1,2,\dots,K, \\ &0 \le q_S \le 1 \qquad \text{for all} \quad S \in \mathbf{C}\,([K],k) \end{aligned}$$

Note that Problem 1 always has a solution by Proposition 1. However, solving Problem 1 is not a good idea because it takes at least $\mathbf{C}\,(K,k) = \Omega(K^k)$ time for fixed k using a general solver. Note that $\sum_{i\in S} q_S = p_i$ for $i = 1, 2, \dots, K$ can be expressed as $A\boldsymbol{q} = \boldsymbol{p}$ using $K \times \mathbf{C}\,(K,k)$ matrix A, $\mathbf{C}\,(K,k)$-dimensional vector $\boldsymbol{q} = (\cdots, q_S, \cdots)^\top$ and K-dimensional vector $\boldsymbol{p} = (p_1, \cdots, p_K)^\top$ and for any mutually independent K column vectors $\boldsymbol{a}_{S_1}, \boldsymbol{a}_{S_2}, \dots, \boldsymbol{a}_{S_K}$ of A, there is a solution of $A\boldsymbol{q} = \boldsymbol{p}$ whose variables are zero except $q_{S_1}, q_{S_2}, \dots, q_{S_K}$, but it is not guaranteed that $0 \le q_{S_i} \le 1$ holds for all $i = 1, 2, \dots, K$.

Fortunately, there is a nice technique called *dependent rounding* [6], which can efficiently select a set of k distinct actions from $[K]$ while satisfying the condition that each action i is selected with probability p_i. Dependent rounding developed by Gandhi et al. [6] is a kind of technique that randomly selects a set of edges from a bipartite graph under some cardinality constraints. Our selection problem is a special case of the problems they considered, the case that the bipartite graph is a star. In such case, the algorithm can be described as shown in Fig. 5, which we call **DepRound** here. In the algorithm, $(p_1, p_2, ..., p_K)$ is

```
DepRound % Dependent Rounding
Inputs: Natural number $k(< K)$, $(p_1, p_2, ..., p_K)$ with $\sum_{i=1}^{K} p_i = k$
Output: Subset of $[K]$ with $k$ elements
  while there is an $i$ with $0 < p_i < 1$ do
    Choose distinct $i$ and $j$ with $0 < p_i < 1$ and $0 < p_j < 1$
    Set $\alpha = \min\{1 - p_i, p_j\}$ and $\beta = \min\{p_i, 1 - p_j\}$
    Update $p_i$ and $p_j$ as
      $(p_i, p_j) = \begin{cases} (p_i + \alpha, p_j - \alpha) \text{ with probability} \frac{\beta}{\alpha+\beta} \\ (p_i - \beta, p_j + \beta) \text{ with probability} \frac{\alpha}{\alpha+\beta} \end{cases}$
  end while
  return $\{i : p_i = 1, 1 \leq i \leq K\}$
```

Fig. 5. Pseudocode of algorithm **DepRound** [6]

Table 1. Performance Comparison of the algorithms for multiple play setting

Algorithm	Base Algorithm	regret	time comp.	space comp.
Exp3 [4]	-	$O(k^{\frac{3}{2}} T^{\frac{1}{2}} K^{\frac{k}{2}} \sqrt{\log K})$	$\Omega(K^k)$	$\Omega(K^k)$
ComBand [5]	-	$O(k^{\frac{3}{2}} \sqrt{TK \ln K})$	$O(kK^3)$	$O(K^3)$
BOLOM	**Algorithm 1** [1]	$O(kK^{\frac{3}{2}} \sqrt{T \log T})$	$O(K^3)$	$O(K^2)$
Exp3.M	**Exp3** [4]	$O(\sqrt{kTK \log(K/k)})$	$O(K(\log k + 1))$	$O(K)$

probabilistically updated until all the components are 0 or 1 while keeping the condition that $\sum_{i=1}^{K} p_i = k$. The inside of the while-loop is executed at most K times because at least one of p_i and p_j becomes 0 or 1 in each time of the execution. The nice property of the update is to keep the expectation values of p_i, namely, $E[p_i^{t+1}] = E[p_i^t]$ for every $i \in [K]$, where p_i^t denotes p_i after the tth execution of the inside of the while-loop. This is trivial when i is not chosen at the tth execution, and holds even when i is chosen since

$$(p_i + \alpha) \times \frac{\beta}{\alpha + \beta} + (p_i - \beta) \times \frac{\alpha}{\alpha + \beta} = (p_i - \alpha) \times \frac{\beta}{\alpha + \beta} + (p_i + \beta) \times \frac{\alpha}{\alpha + \beta} = p_i.$$

Each execution of the inside of the while-loop needs a constant time, so **DepRound** runs in $O(K)$ time and $O(K)$ space.

8 Concluding Remarks

We have extended adversarial bandit problems studied by Auer et al. [4] to those with multiple plays, and analyzed algorithms for the problem.

From the result shown in Table 1, we can know that **Exp3.M** is the best algorithm for this problem among the four algorithms analyzed here. We are now interested in applying our algorithms to real problems and demonstrating their practical usefulness.

Acknowledgements

We would like to thank anonymous referees for a lot of their helpful comments. This work was partially supported by JSPS KAKENHI 21500128.

References

1. Abernethy, J., Hazan, E., Rakhlin, A.: Competing in the dark: An efficient algorithm for bandit linear optimization. In: Proceedings of the 21st Annual Conference on Learning Theory (COLT08) (2008)
2. Agrawal, R., Hegde, M.V., Teneketzis, D.: Multi-armed bandits with multiple plays and switching cost. Stochastic and Stochastic Reports 29, 437–459 (1990)
3. Anantharam, V., Varaiya, P., Walrand, J.: Asymptotically efficient allocation rules for the multiarmed bandit problem with multiple plays –part i: I.i.d. rewards. IEEE Transactions on Automatic Control 32, 968–976 (1986)
4. Auer, P., Cesa-bianchi, N., Freund, Y., Schapire, R.E.: The nonstochastic multiarmed bandit problem. SIAM Journal on Computing 32, 48–77 (2002)
5. Cesa-bianchi, N., Lugosi, G.: Combinatorial bandits. In: Proceedings of the 22nd Annual Conference on Learning Theory (COLT09) (2009)
6. Gandhi, R., Khuller, S., Parthasarathy, S., Srinivasan, A.: Dependent rounding and its applications to approximation algorithms. Journal of the ACM 53(3) pp. 320–360 (2006)
7. György, A., Linder, T., Lugosi, G., Ottucsák, G.: The on-line shortest path problem under partial monitoring. Journal of Machine Learning Research 8, 2369–2403 (2007)
8. Kleinberg, R.: Notes from week 8: Multi-armed bandit problems. CS 683–Learning, Games, and Electronic Markets, http://www.cs.cornell.edu/courses/cs683/2007sp/lecnotes/week8.pdf (2007)
9. Krein, M., Milman, D.: On extreme points of regular convex sets. Studia Mathematica pp. 133–138 (1940)
10. Mahajan, A., Teneketzis, D.: Multi-armed bandit problems. In: Foundations and Applications of Sensor Management, pp. 121–151. Springer (2007)
11. Nakamura, A., Abe, N.: Improvements to the linear programming based scheduling of web advertisements. Electronic Commerce Research 5, 75–98 (2005)
12. Niculescu-Mizil, A.: Multi-armed bandits with betting. In: COLT09 Workshop. pp. 133–138 (2009)
13. Pandelis, D.G., Tenekezis, D.: On the optimality of the gittins index rule in multiarmed bandits with multiple plays. Mathematical Methods of Operations Research 50, 449–461 (1999)
14. Song, N.O., Teneketzis, D.: Discrete search with multiple sensors. Mathematical Methods of Operations Research 60, 1–14 (2004)
15. Uchiya, T., Nakamura, A., Kudo, M.: Adversarial bandit problems with multiple plays. In: The IEICE Technical Report (2009)
16. Warmuth, M.K., Takimoto, E.: Path kernels and multiplicative updates. Journal of Machine Learning Research pp. 773–818 (2003)