Proceedings of Machine Learning Research vol 132:1–37, 2021 32nd International Conference on Algorithmic Learning Theory

Stochastic Top-K Subset Bandits with Linear Space and Non-Linear Feedback

Mridul Agarwal AGARW180@PURDUE.EDU
Purdue University, West Lafayette, IN, USA

Vaneet Aggarwal VANEET@PURDUE.EDU
Purdue University, West Lafayette, IN, USA

Christopher J. Quinn CJQUINN@IASTATE.EDU
Iowa State University, Ames, IA, USA

Abhishek K. Umrawal AUMRAWAL@PURDUE.EDU
Purdue University, West Lafayette, IN, USA

Editors: Vitaly Feldman, Katrina Ligett and Sivan Sabato

Abstract

Many real-world problems like Social Influence Maximization face the dilemma of choosing the best K out of N options at a given time instant. This setup can be modeled as a combinatorial bandit which chooses K out of N arms at each time, with an aim to achieve an efficient trade-off between exploration and exploitation. This is the first work for combinatorial bandits where the feedback received can be a non-linear function of the chosen K arms. The direct use of multi-armed bandit requires choosing among N-choose-K options making the state space large. In this paper, we present a novel algorithm which is computationally efficient and the storage is linear in N. The proposed algorithm is a divide-and-conquer based strategy, that we call CMAB-SM. Further, the proposed algorithm achieves a *regret bound* of $\tilde{O}(K^{\frac{1}{2}}N^{\frac{1}{3}}T^{\frac{2}{3}})$ for a time horizon T, which is *sub-linear* in all parameters T, N, and K.

1. Introduction

Multi-Armed Bandits (MAB) can be used to solve problems in domains where an agent chooses an arm to play at each time instant and receives a reward. The goal of the agent is to perform online learning to select the best arm as early as possible after some initial exploration. However, many real-world problems are combinatorial in nature, where the agent chooses K out of N arms at each time and receives an aggregate reward. For example, the problem of Social Influence Maximization where the aim is to select a subset of individuals in a social network to adopt a new product or innovation, and the target is to trigger a large cascade of further adoptions (Domingos and Richardson, 2001). We have studied it later in great detail. Other applications include daily advertising campaign characterized by a set of sub-campaigns (Zhang et al., 2012; Nuara et al., 2018), erasure-coded storage (Xiang et al., 2016) where K out of N servers are chosen to obtain the content for each request, and cross-selling item selection with K items in the bundle (Wong et al., 2003).

We consider the setting where the agent plays a complex action made by choosing K out of N arms at each time and only receives an aggregate reward. This reward is a function of the rewards of different arms. This is an instance of Combinatorial Multi-Armed Bandit

(CMAB) problem. Based on the history of actions played and rewards observed, the action at the next time is taken. This paper aims to find an efficient algorithm to minimize the gap between the reward obtained by choosing the best K arms at each time and the arms chosen by the CMAB algorithm, for a given time horizon T. Only an aggregate reward, a possibly non-linear function of individual arms' rewards, is observed. The non-linear function makes it hard to estimate the rewards of each arm individually. Thus, algorithms that either assume access to, or first estimate, the individual arm rewards do not directly work (Chen et al., 2013; Lin et al., 2014). To our knowledge, this is the first work that proposes an algorithm for the CMAB problem with non-linear aggregate rewards without any extra feedback.

One way for solving CMAB with non-linear feedback is to treat every action (consisting of K arms) as an arm and apply the classical MAB framework using Upper Confidence Bound (UCB) algorithm (Auer and Ortner, 2010). However, as the number of arms N and the subset size K increases, the total number of actions increases exponentially. Consider a case where $N = 30$, $K = 15$, and playing each action costs 1 second. The time to explore all possible actions a single time will take $\binom{30}{15}$ seconds or approximately 5 years. Moreover, to store all the possible actions and their rewards requires large memory, which might not be possible for applications with limited storage constraints (Liau et al., 2018). This paper proposes a novel algorithm, called CMAB-SM, which explores only $O(N)$ actions, and achieves a regret bound of $\tilde{O}(K^{\frac{1}{2}}N^{\frac{1}{3}}T^{\frac{2}{3}})$ [1]. We note that using the UCB approach as mentioned, the regret bound will be $O\left(\sqrt{\binom{N}{K}T}\right)$, which is better than the proposed bound of $O(T^{2/3})$ for small values of N and K. However, this is not the case when N and K are large. Again for the case of $N = 30$, and $K = 15$, the regret bounds of CMAB-SM will match that of UCB only when T is approximately 4×10^{26}. Further, CMAB-SM has a storage complexity of $O(N)$, and per round time complexity of $\tilde{O}(K)$, making it efficient.

1.1. Our Contributions

The main contributions of this paper can be summarized as follows:

1. We propose CMAB-SM, the first, efficient algorithm for the CMAB problem when the reward is non-linear and no additional feedback about individual arms is available.
2. We use the theory of stochastic dominance to obtain ordering of individual arms instead of estimating arm distributions directly, which alleviates the need of linear feedback. This approach may be of independent interest for MAB problems.
3. We prove that CMAB-SM achieves a regret of $\widetilde{O}(K^{\frac{1}{2}}N^{\frac{1}{3}}T^{\frac{2}{3}})$ when the exploration-exploitation procedure runs for time horizon T. Thus the regret is *sub-linear* with respect to each parameter.
4. We prove the algorithm is efficient, with time complexity of $O(TK \log K)$ and space complexity of $O(N)$.
5. We apply CMAB-SM to some synthetic problems and show that it outperforms the UCB algorithm.

1. $\tilde{O}()$ hides the logarithmic factors.

1.2. Key techniques

We now summarize the method and proof techniques used. CMAB-SM divides all N arms into groups of $K+1$ arms, such that each group contains only $K+1$ actions as there are K arms to choose from $K+1$ arms. Since choosing K out of $K+1$ is equivalent to removing 1 out of $K+1$, there is a one-to-one mapping between arms and actions in a group. We sort the actions (and the corresponding arms) in each group which requires time steps of polynomial order in K. We then merge those groups one by one and obtain the best K arms.

For the analysis, we assume certain properties on the reward distributions of the different arms, and on the function of rewards of individual arms. More precisely, we use the theory of stochastic dominance to differentiate between cumulative distribution functions. The reward distributions of different arms are assumed to dominate or be dominated by each other. Further, the non-linear function is assumed to be symmetric in the rewards obtained from each arm, and the mean of the function is assumed to be continuous (in terms of dominated inputs). These properties are satisfied in case of a few reward distributions (e.g., Bernoulli rewards), and few non-linear functions (e.g., maximum).

1.3. Related Work

Combinatorial Bandits have been studied where the agent chooses K of the N arms in each round (Dani et al., 2008b; Cesa-Bianchi and Lugosi, 2012; Audibert et al., 2014; Dani et al., 2008a; Abbasi-Yadkori et al., 2011). In these works, it is assumed that the reward function in each round is linear in the different arms. They consider a setting where at time t, the agent selects an arm $x_t \in D_t$ and observes a reward $\theta^T x_t$, where $D_t \subset \mathbb{R}^K$ is the decision set and $\theta \in \mathbb{R}^K$ is a constant vector. Due to this linear function, the problem is also called online linear optimization. The algorithms proposed in these works use the linearity of the reward function to estimate rewards of individual arms and achieve a regret of $\tilde{O}\left(\sqrt{T}\right)$. Weights are then assigned to each of the $\binom{N}{K}$ actions to decide the action in the next round; such approaches are not computationally efficient for large N. The work of (Liau et al., 2018) reduces the space complexity at the cost of regret bounds. However, they still loop over all the arms and the regret bound becomes exponential in K. For a linear function, such as sum of rewards, we can construct arms which is a binary K-sparse vector of length N. Such a setting has N unknown variables, and those unknowns could be obtained using least squares as done by (Dani et al., 2008a) or regularized least squares (Abbasi-Yadkori et al., 2011). We consider non-linear functions for which the expected rewards of individual arms could not be obtained using least squares solution.

(Filippi et al., 2010; Jun et al., 2017; Li et al., 2017) studied the problem of generalized linear models (GLM) where reward r_t is a function ($f : \mathbb{R} \to \mathbb{R}$) of $z = \theta^T x_t$. Generalized linear models assume the distribution of bandit reward r_t belongs to a cannonical exponential family. The exponential distribution allows the use of log-likelihood maximization to obtain estimates of arm parameters which increase the likelihood of observed rewards. Generalized Linear Models assume that the expected reward of the arm played is a non linear function of the linear combination of features of the action played with a fixed parameter. This is different than our setup where we assume the reward of the action played is a non linear function of individual realization of rewards of each arm.

GLM (and linear) models have long and rich history across many disciplines such as finding a target item among multiple options (Kveton et al., 2020). GLMs also have many interesting theoretical and statistical properties. But there are settings where GLMs do not accurately model rewards. For example, in the case where multiple arms are selected and the joint reward is the maximum of individual rewards, the joint rewards is not a linear combination of the individual arm rewards.

(Kveton et al., 2014) provides a UCB style algorithm for matroid bandits, where the agent selects a maximal independent set of rank K to maximize sum of rewards of each arm. They assume rewards of each of the K arms is also observed in each round. Such a setup, where the rewards of each of the K arms is also available to agent, is referred to as a semi-bandit problem. (Gai et al., 2010) also considered the problem of semi-bandits for the problem of maximum weighted matching for cognitive radio applications. (Kveton et al., 2015b) showed that the UCB algorithm provides a tight regret bound for semi-bandit combinatorial bandit problem with linear reward function. The authors of (Chen et al., 2013) considered combinatorial semi-bandit problem with non-linear rewards using a UCB style analysis. The authors of (Lin et al., 2014) assumed combinatorial bandit problem with non-linear reward function and feedback, where the feedback is a linear combination of rewards of the K arms. Such feedback of linear function of rewards allows for the recovery of individual rewards. In contrast to prior works, this paper does not consider the availability of individual arm rewards or a linear feedback. With only aggregate, non-linear feedback, it might not be possible to obtain the exact values of the rewards of base arms.

1.4. Organization

The rest of the paper is organized as follows. In Section 2, we provide the model under consideration and the assumptions that are taken for the analysis. Section 3 presents the proposed algorithm, CMAB-SM. The main result is provided in Section 4. Section 5 illustrates our results on a synthetic example where a continuous reward distribution and a non-linear reward function is chosen. Section 6 presents the conclusions with directions for future work.

2. Problem Formulation and Assumptions

2.1. Problem Setup

We now describe the stochastic combinatorial multi-armed bandit problem we consider. There are N "arms" labeled as $i \in [N] = \{1, 2, \cdots, N\}$. Each time an arm is chosen or "played," there is a reward. Let $X_{i,t} \in [0, 1]$ be a random variable denoting the reward of the i^{th} arm, at time-step t (also referred to as time t). We assume that $X_{i,t}$ are independent across time and arms, and for any arm the distribution is identical at all times. Also, if some analysis is independent of time variable, we will drop the subscript t and write only X_i. The rewards could be discrete valued, continuous valued, or mixed.

At each time instant, the agent chooses an action $\mathbf{a} = (a_1, a_2, \cdots, a_K)$ which is a K-tuple of arms. Let $\mathcal{A} = \{\mathbf{a} \in [N]^K \mid \mathbf{a}(i) \neq \mathbf{a}(j) \ \forall \ i, j : \ 1 \leq i < j \leq K\}$ be the set of all such actions which can be constructed using N arms. Thus the cardinality of $\mathcal{A}$ is $\binom{N}{K}$. We denote the action played at time t as $\mathbf{a}_t \in \mathcal{A}$. For an action $\mathbf{a} = (a_1, a_2, \cdots, a_K)$ let

$\mathbf{d}_{\mathbf{a},t} = \left(X_{a_1,t}, X_{a_2,t}, \ldots, X_{a_K,t}\right) \in [0,1]^K$ denote the column vector of arm rewards at time t from action $\mathbf{a}$. The reward $r_{\mathbf{a}}(t)$ of action $\mathbf{a}$ at time t is a bounded function $f : [0,1]^K \to [0,1]$ of the rewards from the arms chosen in that action,

$$r_{\mathbf{a}}(t) = f(\mathbf{d}_{\mathbf{a},t}).$$

Later in the text, we will skip index t for brevity, where it is unambiguous. If at time t, action $\mathbf{a}_t$ is played, $\mathbf{d}_{\mathbf{a}_t,t}$ will be simplified to $\mathbf{d}_{\mathbf{a}_t}$. Also if we analyze behavior for action $\mathbf{a}$ such that its reward vector is independent and identically distributed across time, we will drop the subscript and use only $\mathbf{d}_{\mathbf{a}}$ for brevity.

In many practical systems, the real reward is a non-linear function of noisy reward instead of a non-linear function of expected rewards plus noise. For example, consider a distributed system. A job may be forked into multiple parallel tasks. The completion time of the job depends is the maximum time taken to finish any of the task. Hence, the completion time of the job is a non-linear function of the completion time of the sub-tasks. Another example for non-linear function of individual rewards is item selection in cross selling. In cross-selling the total profit of the seller is sum of individual profits plus an additional advantage made from the combined transaction. The additional advantage can be modelled as a quadratic term of the individual items sold (Wong et al., 2003).

For a linear case, the two cases are equivalent as

$$r_t = f(\mathbf{d}_{\mathbf{a}_t}) = \theta^T \mathbf{d}_{\mathbf{a}_t} = \theta^T(\mathbb{E}\left[\mathbf{d}_{\mathbf{a}_t}\right] + \eta_t) = \theta^T \mathbb{E}\left[\mathbf{d}_{\mathbf{a}_t}\right] + \theta^T \eta_t = f\left(\mathbb{E}\left[\mathbf{d}_{\mathbf{a}_t}\right]\right) + \epsilon_t \tag{1}$$

For non-linear case, this formulation does not simply reduce to function of expected rewards plus noise. However, we can still write the reward as the expected bandit reward plus noise, or

$$r_t = \mathbb{E}[f(\mathbf{d}_{\mathbf{a}_t})] + \epsilon_t \tag{2}$$

Similarly to the work of (Auer and Ortner, 2010), we aim to maximize the expected reward for the actions selected over time. Further, we intend to improve the finite time regret bounds for the same. We denote the expected reward of any action $\mathbf{a} \in \mathcal{A}$ as $\mu_{\mathbf{a}}$, or $\mu_{\mathbf{a}} = \mathbb{E}_{\mathbf{d}_{\mathbf{a}_t}|\mathbf{a}_t=\mathbf{a}}[r_{\mathbf{a}}]$.

We assume that there is an action $\mathbf{a}^*$ for which the expected reward $\mu_{\mathbf{a}^*}$ is highest among all actions $\mathbf{a} \in \mathcal{A}$,

$$\mathbf{a}^* = \arg\max_{\mathbf{a}\in\mathcal{A}} \mu_{\mathbf{a}}$$

We refer to this action $\mathbf{a}^*$ as "optimal." Given an optimal action, regret for an action at time t can be defined as follows.

Definition 1 (Regret) *The regret of an action $\mathbf{a}_t$ at time t is defined as the difference between the reward obtained by the optimal action and the reward obtained by $\mathbf{a}_t$, or*

$$R(t) = r_{\mathbf{a}^*}(t) - r_{\mathbf{a}_t}(t) \tag{3}$$

The objective is to minimize the expected regret (also known as pseudo-regret) accumulated during the entire time horizon,

$$W(T) = \mathbb{E}_{\mathbf{a}_1, r_{\mathbf{a}_1}(1), \cdots, \mathbf{a}_T, r_{\mathbf{a}_T}(T)} \left[\sum_{t=1}^{T} R(t) \right] \tag{4}$$

2.2. Assumptions

We now discuss the technical assumptions which are required to prove the regret bounds for the CMAB-SM algorithm. We note that many of the assumptions are not required for the algorithm to work, only to prove the guarantees.

The function f is assumed to be symmetric, so that the ordering within the tuple does not matter. In other words, the rewards for an action is symmetric in its constituent arms. This assumption is true for certain problem settings where the ordering among the individual arms is not important, like the maximum of rewards, or the sum of rewards of the individual arms. This assumption is given as follows.

Assumption 1 (Symmetry) *f is a symmetric function of the rewards obtained by the constituent arms. More precisely, let $\Pi(\mathbf{d})$, be an arbitrary permutation of $\mathbf{d}$. Then, the reward observed will be identical for both $\Pi(\mathbf{d})$ and $\mathbf{d}$, or*

$$f(\mathbf{d}) \quad = \quad f(\Pi(\mathbf{d})) \tag{5}$$

In the rest of the text, we denote $\Pi(\cdot)$ as a permutation, where $\Pi(\mathbf{y})$ is one of the possible permutations of the vector $\mathbf{y}$. We now define gap Δ between two actions as follows.

Definition 2 (Gap) *The Gap $\Delta_{\mathbf{a}_1,\mathbf{a}_2}$ between any two actions $\mathbf{a}_1, \mathbf{a}_2 \in \mathcal{A}$ is defined as the difference of expected rewards of the actions, or*

$$\Delta_{\mathbf{a}_1,\mathbf{a}_2} = \mu_{\mathbf{a}_1} - \mu_{\mathbf{a}_2} = \mathbb{E}[f(\mathbf{d}_{\mathbf{a}_1})] - \mathbb{E}[f(\mathbf{d}_{\mathbf{a}_2})] \tag{6}$$

We assume that there is an optimal action $\mathbf{a}^*$ for which the expected reward is highest among all actions $\mathbf{a} \in \mathcal{A}$. We denote the reward of the optimal action by $\mu_{\mathbf{a}^*}$. The Gap of an action $\mathbf{a} \in \mathcal{A}$ with respect to the optimal action is simply written as $\Delta_{\mathbf{a}}$, or, $\Delta_{\mathbf{a}} = \Delta_{\mathbf{a}^*,\mathbf{a}}$

Remark 3 *Using linearity of expectation it can be seen that, $\mathbb{E}[R(t)] = \Delta_{\mathbf{a}_t}$*

From remark 3, the expected regret accumulated during the entire time horizon can be written as,

$$W(T) = \mathbb{E}\left[\sum_{t=1}^{T} R(t) \right] = \sum_{t=1}^{T} \Delta_{\mathbf{a}_t} \tag{7}$$

We define the maximum regret of all possible actions as $R_{max} = \max_{\mathbf{a} \in \mathcal{A}} \Delta_{\mathbf{a}}$.

We now use the concept of stochastic dominance to order two arms. Assume that there exists a first-order stochastic dominance between any two arms which is defined as follows.

Definition 4 (First-Order Stochastic Dominance) *A random variable X has first-order stochastic dominance (FSD) over another random variable Y (or $X \succ Y$), if for any outcome x, X gives at least as high a probability of receiving at least x as does Y, and for some x, X gives a higher probability of receiving at least x,*

$$\begin{aligned} X \succ Y \Leftrightarrow P(X \geq x) \geq P(Y \geq x) \, \forall x \in \mathbb{R}, \\ P(X \geq x) > P(Y \geq x) \;\; \text{for some } x \in \mathbb{R}. \end{aligned} \tag{8}$$

Assumption 2 (FSD between arms) *There exists a dominance ordering between all the arms, which is defined using FSD. In other words, for each pair of arms i and j, either $X_i \succ X_j$ or $X_j \succ X_j$.*

The FSD implies second order stochastic dominance, which indicates that the mean of the dominating random variable is at least as much as the mean of the dominated random variable (Hadar and Russell, 1969; Bawa, 1975). This is summarized in the following lemma.

Lemma 5 ((Hadar and Russell, 1969; Bawa, 1975)) *If a random variable X has FSD over another random variable Y (or, $X \succ Y$), then the expected value of X is at least the expected value of Y, or*

$$\mathbb{E}[X] > \mathbb{E}[Y] \tag{9}$$

Remark 6 *From Lemma 5, we note that if arm i dominates arm j, then the mean reward for arm i is strictly greater than that of arm j. Thus, under Assumption 2,*

$$\mathbb{E}[X_i] \neq \mathbb{E}[X_j], \forall i, j \in \{1, \cdots, N\} \tag{10}$$

Such strict dominance exists for Bernoulli and exponential reward distribution functions.

Since we can construct a new action by changing the arms of an existing action, we define the replacement function $h(\cdot)$ which changes an element i of a given reward vector $\mathbf{d}$ (where each entry in the reward vector is a random variable with the distribution of the corresponding arms).

Definition 7 (Replacement function) *The replacement function $h(\cdot)$ is defined as a function on $\mathbb{R}^{K+2}$, which replaces the i^{th} element of vector $\mathbf{d}$ with x, or*

$$h(\mathbf{d}, i, x) = (\mathbf{d}(1), \; \ldots, \; \mathbf{d}(i-1), \; x, \; \mathbf{d}(i+1), \; \ldots, \; \mathbf{d}(K)) \, .$$

For a random variable X, $h(\mathbf{d}, i, X)$ is also a random variable. We also assume that the expected reward of an action is strictly increasing function of the rewards obtained by the individual arms.

Assumption 3 (Strictly Increasing) *$f(\cdot)$ is element-wise, strictly increasing function of the individual rewards obtained by the constituent arms. More precisely,*

$$f(h(\mathbf{d}, i, x)) \; > f(h(\mathbf{d}, i, y)) \;\; \forall x > y \; ; \, x, y \in [0, 1] \; \forall \mathbf{d} \in \mathbb{R}^K \tag{11}$$

Even though we assume strictly increasing function, the analysis also holds for strictly decreasing function f by transforming the reward function as $f_n(\mathbf{d}) = 1 - f(\mathbf{d})$. In order to compare the distance between individual reward vectors from two different actions, we need to find the difference in the two individual reward vectors up to a permutation, since the reward function is permutation invariant. With this distance metric in mind, we assume that $f(\cdot)$ is Lipschitz continuous (in an expected sense), which is formally described in the following.

Assumption 4 (Continuity of expected rewards) *The expected value of $f(\cdot)$ is Lipschitz continuous with respect to the expected value of the rewards obtained by the individual arms, meaning*

$$\left| \mathbb{E}\left[f(\mathbf{d}_1)\right] - \mathbb{E}\left[f(\mathbf{d}_2)\right] \right| \leq U_1 \min_{\Pi} \left|\left|\mathbb{E}[\mathbf{d}_1] - \Pi(\mathbb{E}[\mathbf{d}_2])\right|\right|_2 \tag{12}$$

for any given random vectors $\mathbf{d}_1$ and $\mathbf{d}_2$ and for some $U_1 < \infty$, where Π is minimized over all permutations of $\{1, \cdots, K\}$.

Corollary 8 *Assumption 4 also implies*

$$\left|\mathbb{E}\left[f(h\left(\mathbf{d}, i, X\right))\right] - \mathbb{E}\left[f(h\left(\mathbf{d}, i, Y\right))\right] \right| \leq U_1 \left|\mathbb{E}[X] - \mathbb{E}[Y]\right| \tag{13}$$

for any given random vector $\mathbf{d}$ and any $i \in \{1, \cdots, K\}$.

We further assume a lower bound in (13) as formally stated in the following assumption.

Assumption 5 (Continuity of individual expected rewards) *We also assume that the continuity given in (13) also has a similar lower bound. More precisely, there is a $U_2 < \infty$ such that*

$$\left|\mathbb{E}[X] - \mathbb{E}[Y]\right| \leq U_2 \left| \left(\mathbb{E}\left[f(h\left(\mathbf{d}, i, X\right))\right] - \mathbb{E}\left[f(h\left(\mathbf{d}, i, Y\right))\right]\right) \right| \tag{14}$$

for any given random vectors $\mathbf{d}$ and any $i \in \{1, \cdots, K\}$.

Assumption 5 holds for many well behaved functions in practical scenarios. For example, $f(\cdot) = \max(\cdot)$ with Bernoulli rewards for individual arms[2], $f(\cdot) =$ sum of individual rewards, or $f(\cdot) =$ concave utility function of sum of individual rewards.

Corollary 9 *Combining Corollary 8 and Assumption 5 and defining $U = \max(U_1, U_2)$, we have*

$$\frac{1}{U}\left|\mathbb{E}[X] - \mathbb{E}[Y]\right| \leq \left| \left(\mathbb{E}\left[f(h\left(\mathbf{d}, i, X\right))\right] - \mathbb{E}\left[f(h\left(\mathbf{d}, i, Y\right))\right]\right) \right| \tag{15}$$

$$\leq U\left|\mathbb{E}[X] - \mathbb{E}[Y]\right|, \tag{16}$$

for any given random vector $\mathbf{d}$ and any $i \in \{1, \cdots, K\}$.

We note that linear bandits become a special case of the assumptions we considered.

2. We note that maximum function, in general, does not satisfy Assumption 5. However, if an arm exists such that its rewards are always higher compared to other arms, the agent can place any other arm among the $K - 1$ arms, without incurring any regret.

3. Proposed Algorithm

The proposed algorithm, called CMAB-SM, is an explore then exploit strategy which aims to minimize the expected regret, be computationally efficient, and have a storage complexity which is linear with N and independent of K. CMAB-SM, described in Algorithm 1, utilizes the fact that for CMAB problem, choosing K arms from a set of $K+1$ arms has $K+1$ actions thus making the problem solvable using the standard MAB approach. In other words, if $N = K+1$, then the complexity is $\binom{K+1}{K} = K+1$, and only $K+1$ actions needs to be optimized.

We construct a group G which is a vector of length $K+1$ consisting of arm indices, or $G \in [N]^{K+1}$. Then, we can construct $K+1$ actions, each action using all but one entries in the group G.

Let $\mathbf{a}^G_{-i}$ be an action in group G with $G(i)^{th}$ arm left out, where $G(i), i \in \{1, \cdots, K+1\}$, is the i^{th} entry of the group.

With $X_{G(i)}$ denoting the reward of arm $G(i)$, the individual reward vector $\mathbf{d}_{\mathbf{a}^G_{-i}}$ with the action $\mathbf{a}^G_{-i}$ is

$$\mathbf{d}_{\mathbf{a}^G_{-i}} = (X_{G(1)}, \ldots, X_{G(i-1)}, X_{G(i+1)}, \ldots, X_{G(K+1)}). \tag{17}$$

The (random) reward obtained at any time with this action is $r_{\mathbf{a}^G_{-i}} = f\left(\mathbf{d}_{\mathbf{a}^G_{-i}}\right)$, with a mean reward of $\mu_{\mathbf{a}^G_{-i}} = \mathbb{E}\left[f\left(\mathbf{d}_{\mathbf{a}^G_{-i}}\right)\right]$. The next result shows that an ordering on $\binom{K+1}{K}$ actions made using $K+1$ arms gives an ordering on $K+1$ arms under the considered assumptions.

Lemma 10 *An ordering on $\binom{K+1}{K}$ actions, in group G, made using $K+1$ arms gives an ordering on $K+1$ arms. In other words, if an ordering exists between actions $\mathbf{a}^G_{-i}$ and $\mathbf{a}^G_{-j}$, then an ordering exists between arms $G(i)$ and $G(j)$. More precisely,*

$$\mu_{\mathbf{a}^G_{-i}} > \mu_{\mathbf{a}^G_{-j}} \Rightarrow \mathbb{E}\left[X_{G(i)}\right] < \mathbb{E}\left[X_{G(j)}\right] \tag{18}$$

Proof (Outline): If we have two actions made from group G by excluding arm $G(i)$ and arm $G(j)$ respectively, then the arm with higher reward will increase the joint reward f as f is an increasing function. The detailed proof is provided in Appendix B. ■

CMAB-SM divides all N arms into groups of $K+1$ arms arbitrarily. Each group now contains only $K+1$ actions. If the last group contains less than $K+1$ arms (if N mod $(K+1) > 0$), arms from other groups are added (repeated) to have $K+1$ arms in the last group. CMAB-SM then picks the first group of $K+1$ arms and orders the arms in the group using SORT subroutine. Using this subroutine, the $K+1$ arms in the group are ordered with respect to expected individual rewards. We also consider G^* as the best K arms seen so far, which are the top K arms in G_1. It later proceeds in $k \in \left\{2, \cdots, \frac{N}{K+1}\right\}$ rounds. In k^{th} round it performs SORT on G_k and merge G_k and G^* using MERGE subroutine to obtain a new G^*. The SORT subroutine orders the $K+1$ arms in G_k. The MERGE subroutine takes the best K arms before this round, G^*; and the best K arms from the

SORT subroutine on G_k and merges them to find the best K arms seen so far and saves them as G^*. This is then inputted to the next value of k to merge with other groups.

At the end of $\left(\frac{N}{K+1}\right)^{th}$ round, we would have played all arms in each group and merged them, thus resulting in an optimal action which maximizes the expected reward for the remaining time slots. Apart from the sort and merge scheme, we also use a hyperparameter λ in our algorithm. λ denotes the minimum gap the agent can resolve between any two arms. If any the gap between arm i and arm j, the algorithm cannot determing which arm is better with high probability and selects the arm with higher sample mean as the better arm. This behaviour is common in both SORT and MERGE subroutine. We now describe the algorithms used in CMAB-SM which are SORT and MERGE subroutines in detail.

3.1. SORT

The SORT subroutine is given in Algorithm 2. In this subroutine, we play $K+1$ actions formed from $K+1$ arms in a group G, each action corresponding to one left out arm. The subroutine proceeds in rounds similar to UCB algorithm by (Auer and Ortner, 2010). By the end of round r, each action is played n_r times so that the expected reward of each action can be estimated within $\pm\Delta_r$. At the end of each round, the estimates are used to sort the arms, where the arms $G(i), G(j)$ are considered sorted when the upper bound on reward estimate of action $\mathbf{a}^G_{-i}$ is less than lower bound of action $\mathbf{a}^G_{-j}$. When an arm is placed at its true sorted location in the group, its corresponding action is not sampled again. The procedure ends when $\Delta_r < \lambda$ or when all $K+1$ arms are sorted. At the end of the algorithm, only top K arms are provided as output.

3.2. MERGE

The MERGE subroutine is given in Algorithm 3. The MERGE subroutine aims to merge two groups, each with K sorted arms to give sorted best K arms. Since we only want the best K arms from the merged $2K$ arms to be sorted, it can be done with only $K+1$ arm comparisons.

Starting with two K-sized sorted groups G_1, and G_2 and an optimal group which is empty at the start of the subroutine, we identify the best K out of $2 \times K$ arms by figuring the best arm one by one. Starting with both i and j as 1, we construct a new action by replacing the i^{th} arm of group G_1 by the j^{th} arm of group G_2. Note that if after replacement, the reward is bigger, it implies that the added j^{th} arm of G_2 is the next arm in the sorted final list else the i^{th} arm of G_1 is the next arm in the sorted final list. In order to differentiate between the two actions, the procedure similar to the SORT subroutine is used. Based on whether the i^{th} arm of G_1 or j^{th} arm of G_2 made in the optimal set, i or j is incremented and the procedure is repeated till the K best arms in the merger of the two groups are obtained.

3.3. Complexity of CMAB-SM

We now analyze the complexity of CMAB-SM for both storage and computation at each time step. Detailed subroutines are provided in Appendix G, with the key pseudo-codes in Algorithm 1-3.

The algorithm while running SORT, or MERGE subroutine stores the reward of each action in the group, and sorts all the actions. The total storage at any step is no more than $O(K)$. Even when the groups are being merged, $O(K)$ temporary storage is used for the merged rewards. This merged group is then used to decide the action in the exploiting phase. Thus, the maximum storage at any time is $O(K)$ for the subroutines and $O(N)$ for CMAB-SM. To evaluate the computational complexity at each time-step, we consider the three cases of what the algorithm may be doing at a time step.

Algorithm 1 CMAB-SM(T, N, K)

Initialize separation threshold for mean estimates

$$\lambda = \left(\frac{256N \log 2NT}{T}\right)^{\frac{1}{3}} \tag{19}$$

Partition N arms arbitrarily into $\frac{N}{K+1}$ equal sized groups. $G_k : \ 1 \leq k \leq \frac{N}{K+1}$
$G^* = \text{SORT}(G_1, \lambda, T, N, K)$ ▷ Sort 1^{st} group. For a single group, returned set is optimal
for $k = 2 : \frac{N}{K+1}$ **do** ▷ Sort and Merge subsequent groups
    $G_k = \text{SORT}(G_k, \lambda, T, N, K)$
    $G^* = \text{MERGE}(G^*, G_k, \lambda, T, N, K)$
end for
return G^*

Algorithm 2 SORT(G, λ, T, N, K)

Initialize $\hat{\mu}_i = 0$ for $i \in \{1, 2, \cdots, K+1\}$ ▷ $\hat{\mu}_i$ is the sample mean of $\mathbf{a}^G_{-i}$
$S \leftarrow \phi$ ▷ list of sorted actions
Initialize $r \leftarrow 1$; $\Delta_r \leftarrow \frac{1}{2^r}$; $n_r \leftarrow \frac{2 \log (TNK)}{\Delta_r^2}$ ▷ Initial confidence bounds
while $\Delta_r > \lambda$ **and** $|S| < K+1$ **do**
    for all $i \in \{1, 2, \cdots, K+1\}$ **do**
        if $|\hat{\mu}_i - \hat{\mu}_j| \leq 2\Delta_r$ for some $j \in \{1, 2, \cdots, K+1\} \setminus \{i\}$ **then**
            Sample action $\mathbf{a}^G_{-i}$ for total of n_r times and update $\hat{\mu}_i$
        else if $i \notin S$ **then**
            $S = S \cup \{i\}$
        end if
    end for
    Update $r = r + 1$; $\Delta_r = \frac{\Delta_r}{2}$; $n_r = \frac{2 \log (TNK)}{\Delta_r^2}$
end while
Make S of length $K+1$ (if not already) by adding remaining indices from $G \setminus S$
Order S inverse sorting on $\{\hat{\mu}\}_{1:K+1}$
return $S[1:K]$

1) In SORT subroutine, at the end of each iteration of the while loop, arms in the group are sorted requiring $O(K \log K)$ computations. It then loops over all the arms to place them in the correct order which takes $O(K)$ steps. Thus, the computational complexity in the worst case time-step in sort is $O(K \log K)$. 2) MERGE subroutine at any time step either run action and saves the result, or perform comparisons which are all $O(1)$ at each time. 3) After the MERGE is complete, the best action is available which is then exploited thus

Algorithm 3 MERGE$(G_1, G_2, \lambda, T, N, K)$

Initialize $G^*[1:K] \leftarrow \mathbf{0}$ ▷ Array of size K initialized with 0 to store the optimal arms
$\mathbf{a}_{G_1} = G_1$ ▷ Construct action from arms in Group 1
Initialize $\hat{\mu}_{\mathbf{a}_{G_1}} \leftarrow 0$; $i \leftarrow 1$; $j \leftarrow 1$
Initialize $r_1 \leftarrow 1$; $\Delta_{r_1} \leftarrow \frac{1}{2}$; $n_{r_1} = \frac{2\log(TNK)}{\Delta_{r_1}^2}$
for $k = 1 : K$ **do**
  $\mathbf{a}_{i,j} = (G_1 \setminus \{G_1(i)\}) \cup \{G_2(j)\}$ ▷ Replace i^{th} arm of G_1 by j^{th} arm of G_2
  Initialize $r_2 \leftarrow 1$; $\Delta_{r_2} \leftarrow \frac{1}{2}$; $n_{r_2} = \frac{2\log(TNK)}{\Delta_{r_2}^2}$; $\hat{\mu}_{\mathbf{a}_{i,j}} \leftarrow 0$ ▷ For every new action constructed
  while $(\Delta_{r_2} > \lambda)$ **and** $(G^*[k] == 0)$ **do**
    While total samples of action $\mathbf{a}_{G_1}$ ¡ n_{r_1}, Sample action $\mathbf{a}_{G_1}$ to update $\hat{\mu}_{\mathbf{a}_{G_1}}$
    While total samples of action $\mathbf{a}_{i,j}$ ¡ n_{r_2}, Sample action $\mathbf{a}_{i,j}$ to update $\hat{\mu}_{\mathbf{a}_{i,j}}$
    if confidence bounds of $\mathbf{a}_{G_1}$ and $\mathbf{a}_{i,j}$ do not overlap **then**
      Update $G^*[k], i, j$
    else
      Update $r_2 = r_2 + 1$; $\Delta_{r_2} = \frac{\Delta_{r_2}}{2}$; $n_{r_2} = \frac{2\log(TNK)}{\Delta_{r_2}^2}$
      If $r_2 > r_1$, update $r_1 = r_1 + 1$; $\Delta_{r_1} = \frac{\Delta_{r_1}}{2}$; $n_{r_1} = \frac{2\log(TNK)}{\Delta_{r_1}^2}$
    end if
  end while
  Update $G^*[k], i, j$ if not updated
end for
return G^* ▷ Merged set of G_1, and G_2

making the complexity in the exploit phase as $O(1)$. Thus, the overall complexity at any time is $O(K \log K)$ which happens due to sorting the actions for the removal of sub-optimal arms after every round in SORT subroutine.

In each call to SORT, the actions are sorted with respect to their mean observed rewards. From Lemma 10, an ordering is also obtained for the corresponding arms. In MERGE subroutine, a new action is constructed from an old action by replacing exactly one arm. The ordering between the old action and the new action gives the ordering between the replaced arm and the new arm. Note that the inequality conditions work in different directions for SORT and MERGE algorithms.

3.4. Other design options

We note that an algorithm can be constructed by keeping the first $K-1$ arms fixed. The algorithm will now select the best arm from remaining $N-K+1$ arms. This arm will now always be kept in first $K-1$ arms. The process is repeated until all K places are filled. However, this algorithm has two issues, 1) **Higher complexity**: The algorithm will need to sort $N-K+1$ arms into place which increases the sorting complexity from $\tilde{O}(K)$ to $\tilde{O}(N)$. **More exploration steps**. Since, this algorithm will now perform exploration after placing every an arm among the first K Group. This increases the time required for exploration by a factor of K, which would increase the order of regret bound. This makes the CMAB-SM a better choice compared to a naïve implementation of UCB by fixing $K-1$ arms.

$$\delta_{G(i)} = \begin{cases} \mathbb{E}\left[X_{G(i)}\right] - \mathbb{E}\left[X_{G(i+1)}\right], & i = 1 \\ \min\left\{\mathbb{E}\left[X_{G(i-1)}\right] - \mathbb{E}\left[X_{G(i)}\right], \mathbb{E}\left[X_{G(i)}\right] - \mathbb{E}\left[X_{G(i+1)}\right]\right\}, & i \in 2, ..., K \\ \mathbb{E}\left[X_{G(i-1)}\right] - \mathbb{E}\left[X_{G(i)}\right], & i = K+1 \end{cases} \tag{20}$$

4. Regret Analysis of the proposed algorithm

In this section, we will present the main result of the paper, related to the regret analysis of the proposed algorithm.

4.1. Bounds on exploitation regret and exploration time

In this subsection, we will bound the regret in the exploitation phase, which indicates the loss in reward due to choosing an incorrect action at the end of the MERGE algorithm. We will also bound the time spent in the SORT and the MERGE subroutines, which is the exploration phase.

We first find the time taken in the sort and merge subroutines.

Lemma 11 (Sort time requirement) *SORT subroutine 2 gives correct ordering on $K+1$ actions in a group G with probability $1 - \frac{K}{2N^2T^2}$, up to precision λ defined in equation (19), where the actions are chosen for at most*

$$\left(\sum_{i=1}^{K+1} \frac{64U^2 \log 2NT}{\max(\delta_{G(i)}^2, \lambda^2)}\right)$$

time steps, where $\delta_{G(i)}$ is given in (20).

Proof (Outline) We sample each action till the confidence intervals around estimated means of any two actions are separated. The confidence intervals reduces as number of samples increases from concentration bounds from Lemma 19 with a lower limit of λ. Using Corollary 9 and Hoeffding's Inequality (Lemma 19) we bound the number of samples required for separation with high confidence. We then use union bounds to bound the total numbers of samples required for each action. The detailed proof is provided in Appendix C. ■

Lemma 12 (Merge time requirement) *MERGE subroutine 5 merges arms in two groups G_1 and G_2 to G correctly with probability $1 - \frac{K}{2N^2T^2}$, up to precision λ as defined in equation (19), where the total number of time steps needed to merge is at most*

$$\left(\sum_{i=1}^{K+1} \frac{64U^2 \log 2NT}{\max(\delta_{G(i)}^2, \lambda^2)}\right),$$

where $\delta_{G(i)}$ is given in (20).

Proof (Outline) While merging two groups G_1 and G_2, we sample two actions till the confidence intervals around the estimated means of both actions are separated by twice

the confidence intervals around the estimates. Again using Lemma 19 and Corollary 9, we bound the number of samples required reduce the confidence intervals sufficiently enough to order two arms. The detailed proof is provided in Appendix C. ■

In order to bound the regret in the exploitation phase, we first characterize the probability that the action decided by CMAB-SM is not the best action.

Lemma 13 (Total error probability) *The probability that the action selected by* CMAB-SM *during the exploitation phase is not the best action (up to precision defined in Equation 19) is at most* $\frac{1}{NT^2}$.

Proof (Outline) We use union bounds to calculate total error probability of the algorithm. We use Lemma 11 and Lemma 12 to calculate failure probability of each sort and merge event respectively. There are total $\frac{N}{K+1}$ groups to be sorted, and $\frac{N}{K+1} - 1$ groups to be merged. Taking union bound over the total number of failure events, and probability of each failure event gives an upper bound on total error probability of the algorithm. The detailed proof is provided in Appendix D. ■

In the next result, we bound the time spent in exploring, including the SORT and MERGE subroutines for all groups.

Lemma 14 (Bound on exploration time steps) *Total time-steps used to SORT all* $\frac{N}{K+1}$ *groups, and merge these sorted groups one after the other is bounded as*

$$T_{exp} \leq \frac{128NU^2 \log 2NT}{\lambda^2} \tag{21}$$

Proof (Outline) Lemma 11 gives the maximum number of samples required to Sort one group. Similarly, Lemma 12 gives the number of samples required to merge two groups. Since there are $\frac{N}{K+1}$ groups to be sorted, and $\frac{N}{K+1} - 1$ groups to be merged. Summing over total number of samples for each groups gives an upper bound on total samples required for exploration. The detailed proof is provided in Appendix E. ■

In the following result, we bound the expected regret in the exploitation phase, caused by CMAB-SM selecting incorrect action.

Lemma 15 (Bounded exploitation regret) *The expected regret when a sub-optimal action* $\hat{\mathbf{a}}^*$ *is returned by* CMAB-SM *is bounded as*

$$\mathbb{E}\left[\Delta_{\hat{\mathbf{a}}^*}\right] \leq U\lambda\sqrt{K} + \frac{U\sqrt{K}}{N^2T^2} \tag{22}$$

Proof (Outline) Regret can arise in Exploitation phase when either SORT algorithm or MERGE algorithm had a failure event. Regret can also come in exploitation phase if two arms are have expected rewards close enough that SORT or MERGE algorithm cannot distinguish between them with high confidence. Combining these two sources of regret and using Assumption 4 gives the upper bound on the expected regret during the exploitation phase. The detailed proof is provided in Appendix F. ■

4.2. Main Result

Our main result is presented in Theorem 16, which states that CMAB-SM algorithm achieves a sub-linear expected regret.

Theorem 16 CMAB-SM *algorithm described in Algorithm 1 has an expected regret accumulated during the entire time horizon upper bounded as*

$$W(T) = \tilde{O}\left(N^{\frac{1}{3}} K^{\frac{1}{2}} T^{\frac{2}{3}}\right) \tag{23}$$

Proof (Outline) We first note that regret of the algorithm for playing sub-optimal action can come during the exploration phase, or during exploitation phase if the exploration resulted in a suboptimal action. Time steps CMAB-SM uses for exploration is the total time spend in SORT and MERGE subroutines. We bound the time steps in both subroutines by $\frac{128NU^2 \log 2NT}{\lambda^2}$ using Lemma 14. Further, if the algorithm results in an sub-optimal arm, the expected regret in a single time step of exploitation phase is $\left(U\lambda\sqrt{K} + \frac{U\sqrt{K}}{NT^2}\right)$ by using Lemma 15. By choosing an optimal value of λ as defined in (19) we obtain the required bound. Having described the outline, we next give the detailed steps of the proof.

(Detailed Steps) We note that the expected regret till time T is sum of expected regret accumulated at each round. We can rewrite it as sum of two phases of the algorithm which are exploration and exploitation as,

$$\begin{aligned}
W(T) &= \sum_{t=1}^{T} \mathbb{E}\left[R(t)\right] && (24)\\
&= \sum_{t=1}^{T_{exp}} \mathbb{E}\left[R(t)\right] + (T - T_{exp}) \times \mathbb{E}\left[\Delta_{\hat{\mathbf{a}}^*}\right] && (25)\\
&\leq \sum_{t=1}^{T_{exp}} \mathbb{E}\left[R(t)\right] + T\mathbb{E}\left[\Delta_{\hat{\mathbf{a}}^*}\right] && (26)\\
&\leq \sum_{t=1}^{T_{exp}} \max\left(R(t)\right) + T\mathbb{E}\left[\Delta_{\hat{\mathbf{a}}^*}\right] && (27)\\
&\leq T_{exp} \max\left(R(t)\right) + T\mathbb{E}\left[\Delta_{\hat{\mathbf{a}}^*}\right], && (28)
\end{aligned}$$

where (25) follows from splitting the regret into exploration-exploitation phase, (26) follows since $T - T_{exp} \leq T$, and (27) follows since mean is at most the maximum.

Using the values for maximum regret in any round, Lemma 15, inequality (68), maximum exploration time from Lemma 14, and maximum exploitation regret from Lemma 15, we have,

$$\begin{aligned}
& W(T) \\
\leq\ & U\sqrt{K}\frac{128NU^2 \log 2NT}{\lambda^2} + T\left(U\lambda\sqrt{K} + \frac{U\sqrt{K}}{NT^2}\right) \\
=\ & \left(\frac{128NU^3\sqrt{K} \log 2NT}{\lambda^2} + TU\lambda\sqrt{K}\right) + \frac{U\sqrt{K}}{NT}. && (29)
\end{aligned}$$

We now choose a value of λ to optimize $W(T)$. Since during the implementation of algorithm U is most likely an unknown quantity, we use the following value of λ

$$\lambda = \left(\frac{256N \log 2NT}{T}\right)^{\frac{1}{3}} \tag{30}$$

Choosing the value of λ as defined in (30), we have the total regret of the algorithm as

$$W(T) \quad \leq \quad (U^3 + 2U)\sqrt{K}\,(32N \log 2NT)^{\frac{1}{3}}\, T^{\frac{2}{3}} + \frac{U\sqrt{K}}{NT}$$

This proves the result as in the statement of the Theorem. ■

This trick where we tune λ after we define the precision in each SORT/MERGE round allows us to eliminate the dependence on potentially hard to order sequences of items. The intuition behind this is, in a finite time horizon, any agent wants to work out the best possible it can get however can do so only up to a certain precision permitted by finite the time available.

4.3. Handling insufficient exploration time

We note that there can be a instance where the algorithm is run with insufficient time for exploration. We first characterize what value of T would result in the algorithm to run with insufficient exploration time. Then, we evaluate the regret in such a scenario indeed occurs.

Note that we assumed that rewards of each arm lies between $[0, 1]$ in Section 2. This results in gap between any two arms is less than 1. For the optimal λ defined in Equation (19), any $T \leq 256N \log(2NT)$ will make $\lambda \geq 1$ which serves no practical purpose based on our assumption. In that case, we arbitrarily select one of the $\binom{N}{K}$ actions and still suffer a maximum regret of $TU\sqrt{K} \leq \lambda TU\sqrt{K}$. We have,

$$\begin{aligned} W(T) &\leq U\sqrt{KT} && (31)\\ &\leq TU\sqrt{K}\lambda && (32)\\ &= U\sqrt{K}(256N \log(2NT))^{1/3}T^{2/3} && (33) \end{aligned}$$

We note that the sub-linear regret in Equation 33 grows as $\tilde{O}(T^{2/3})$ but with a large multiplicative constant. Hence the linear regret in Equation 31 provides a better bound because of limited time to explore all arms.

4.4. Handling unknown time horizon using doubling trick

We now analyze the case where the time horizon T is unknown and the algorithm requires to optimize actions without the knowledge of T to tune λ. We use the standard doubling trick from Multi-Armed Bandit literature (Auer and Ortner, 2010; Besson and Kaufmann, 2018). To use doubling trick we start the algorithm from $T_0 = 0$. We the restart the algorithm after every $T_l = 2^l,\ l = 1, 2, \cdots$ time steps, till the algorithm reaches the unknown T. Each restart of the algorithm runs for $T_l - T_{l-1}$ steps with $T_0 = 0$ with $\lambda_l = \left(\frac{256N \log 2N(T_l - T_{l-1})}{T_l - T_{l-1}}\right)^{1/3}$

To show that the regret is bounded by $T^{2/3}$ for the doubling algorithm, we use Theorem 4 from (Besson and Kaufmann, 2018) which we state in the following lemma.

Lemma 17 *If an algorithm $\mathcal{A}$ satisfies $R_T(\mathcal{A}_T) \leq cT^{\gamma}(\log T)^{\delta} + f(T)$, for $0 < \gamma < 1$, $\delta \geq 0$ and for $c > 0$, and an increasing function $f(t) = o(t^{\gamma}(\log t)^{\delta}$ (at $t \to \infty$), then anytime version $\mathcal{A}' := \mathcal{DT}(\mathcal{A}, (T_i)_{i\in\mathbb{N}})$ with geometric sequence $(T_i)_{i\in\mathbb{N}}$ of parameters $T_0 \in \mathbb{N}^*$, $b > 1$, (i.e., $T_i = \lfloor T_0 b^i \rfloor$) with the condition $T_0(b-1) > 1$ if $\delta > 0$ satisfies,*

$$R_T(\mathcal{A}') \leq l(\gamma, \delta, T_0, b)cT^{\gamma}(\log T)^{\delta} + g(T), \tag{34}$$

with a increasing function $g(t) = o(t^{\gamma}(\log t)^{\delta})$ and a constant loss $l(\gamma, \delta, T_0, b) > 1$,

$$l(\gamma, \delta, T_0, b) := \left(\left(\frac{\log(T_0(b-1)+1)}{\log(T_0(b-1))}\right)^{\delta}\right) \times \frac{b^{\gamma}(b-1)^{\gamma}}{b^{\gamma}-1} \tag{35}$$

Using Lemma 17 for $b = 2, \gamma = 2/3, \delta = 1/3$, we can convert our algorithm to an anytime algorithm.

4.5. Comparison between CMAB-SM and UCB algorithm

We now compare the regret bound with the one that would be achieved by using the UCB approach on each of the $\binom{N}{K}$ actions.

Lemma 18 (UCB Regret, (Auer and Ortner, 2010)) *For a Multi Armed Bandit setting with action space $\mathcal{A}$, time horizon T, and precision $\lambda \approx \sqrt{\frac{|\mathcal{A}|\log|\mathcal{A}|}{T}}$, the expected regret accumulated during entire time horizon T using improved UCB algorithm is upper bounded as,*

$$W(T) \leq \sqrt{|\mathcal{A}|T}\frac{\log\left(|\mathcal{A}|\log|\mathcal{A}|\right)}{\sqrt{\log|\mathcal{A}|}}$$

Bounding the size of action space by using Stirling's approximation (Slomson, 1997), we get expected regret accumulated regret at time T of UCB algorithm as,

$$W_{UCB}(T) = \tilde{O}\left(\left(\frac{eN}{K}\right)^{\frac{K}{2}} T^{\frac{1}{2}}\right)$$

For UCB approach to outperform CMAB-SM, T has to be very large. More formally,

$$W(T) > W_{UCB}(T), \text{ when } T = \tilde{\Omega}\left(\frac{e^{3K}N^{3K-2}}{K^{3K+3}}\right)$$

Even for an agent which can play 10^{12} actions per second, this will take about 10 million years to outperform CMAB-SM for a setup with $N = 30$ and $K = 15$. Hence, for all practical problems when an agent has a large number of arms to play simultaneously, the CMAB-SM algorithm will outperform the UCB algorithm.

4.6. Discussion of $\tilde{O}(T^{2/3})$ regret bound

Lower bound of $\Omega(\sqrt{NT})$ for Linear Bandits was proven in (Dani et al., 2008a), and lower bound of $\Omega(\sqrt{KNT})$ is proven for semi-bandits by (Kveton et al., 2015b). Note that any $\tilde{O}(\sqrt{T})$ regret bound algorithm compares all the actions with best possible action either by getting individual regret for semi-bandits or by estimating individual regret as in linear bandits (Dani et al., 2008a; Abbasi-Yadkori et al., 2011; Kveton et al., 2015b).

Individual sub-optimal action $\mathbf{a}$ is eliminated in $O(\frac{1}{\Delta_{\mathbf{a}}^2})$ number of samples. The regret from this action then becomes $\Delta_{\mathbf{a}} \times \frac{1}{\Delta_{\mathbf{a}}^2} = \frac{1}{\Delta_{\mathbf{a}}}$ and hence, the cumulative regret is of the form of $\frac{1}{\lambda}T + \lambda T$. This gives $O(\sqrt{T})$ regret for $\lambda = \frac{1}{\sqrt{T}}$.

Due to the bandit feedback, we do not directly obtain the rewards of the individual arms but we only get an ordering on any two arms which can be compared. To eliminate arms early we need some estimator of individual arms similar to linear bandits. Also, the regret accumulated to eliminate arm i is not of the form $O(1/\Delta_i)$. This also hinders the development of an $\tilde{O}(\sqrt{T})$ bound with linear space and time complexity.

5. Numerical Evaluation

In this section, we evaluate CMAB-SM under a synthetic problem setting. We compare the result with improved UCB algorithm as described in (Auer and Ortner, 2010). Since this paper provides the first result with non-linear reward functions for CMAB problem with bandit feedback, we compare with the UCB algorithm (Auer and Ortner, 2010) which is optimal for small N and K while having the regret scale with $\binom{N}{K}$.

For evaluations, we ran the algorithm for $T = 10^6$ time steps and averaged over 30 runs. We compare cumulative regret at each t starting from $t = 0$, which is defined as,

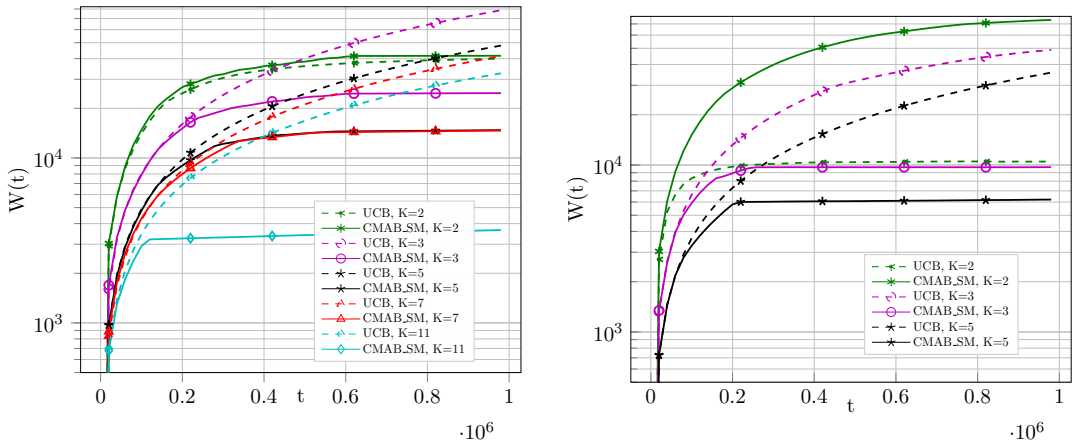
$$W(t) = \sum_{t'=0}^{t} R(t')$$

We consider two values of $N \in \{12, 24\}$. For $N = 12$, we choose $K \in \{2, 3, 5\}$, while for $N = 24$, we choose $K \in \{2, 3, 5, 7, 11\}$. Since the arms must have FSD over each other, we describe one example single parameter distributions for the reward of each arm that have this property. We consider a random variable Y_i which follows an exponential distribution with parameter λ_i, $Y_i \sim exp(\lambda_i)$. Since this random variable can take values in $[0, \infty)$, we transform the variable using arctan function to limit it to the set $[0, \pi/2)$ as

$$X_i \quad = \quad \frac{2}{\pi} \arctan(Y_i). \tag{36}$$

We note that arm i has FSD over arm j if $\lambda_i > \lambda_j$. Thus, this reward distribution satisfies Assumption 2 as long as no two arms have same parameter, or $\lambda_i \neq \lambda_j$ for any $i \neq j$. Figure 2b in Appendix plots $P(X \geq x)$ of the reward function for different values of λ_i. We see that $P(X \geq x)$ is larger for the distribution with larger value of λ, and for any two different values of λ, there is x (e.g., any $x \in (0, 1]$) such that $P(X \geq x)$ are not the same thus showing that the reward distributions satisfy Assumption 2.

We consider an online portal that can display K products because of certain limitations. Assume that the reward, which indicates the profit from the sale of a product, from each arm



(a) $N = 24$ (b) $N = 12$

Figure 1: Empirical regret of UCB and CMAB-SM algorithm for the case when the reward of actions is a non linear function of rewards of individual arms for various values of N and K. As it can be seen from the plots, except for the case of $K = 2$, CMAB-SM incurs significantly lower regret than UCB algorithm.

follows the distribution as defined in (36). However, there is an additional benefit received when multiple products are sold together, e.g., reduced overhead/shipping costs. We define a non linear reward r as a function f of individual arms as follows:

$$f\left(X_1, X_2, \cdots, X_K\right) = \frac{2}{K(K+1)} \sum_{i=1}^{K} \sum_{j \geq i}^{K} X_i X_j. \tag{37}$$

The expected value of the reward in terms of expected value of rewards of individual arms is

$$\mathbb{E}\left[f\left(X_1, X_2, \cdots, X_K\right)\right] = \frac{2}{K(K+1)} \left(\sum_{i=1}^{K} \mathbb{E}\left[X_i^2\right] + \sum_{i=1}^{K} \sum_{j>i}^{K} \mathbb{E}\left[X_i\right] \mathbb{E}\left[X_j\right] \right),$$

This result follows from linearity of expectation.We note that the expected reward is strictly increasing with respect to the expected values of individual rewards. Figure 1 shows the evaluation results for setting where the reward of an action is the function described in Equation (37) of the rewards of individual rewards of the arms. We see that for both values of N, CMAB-SM outperforms UCB for $K > 3$ in the time step range considered. Further, for $N = 24$ and $K = 2$, the gap between the proposed algorithm and UCB is small. In summary, when the value of $\binom{N}{K}$ is moderately large, and T is not significantly large $(T < \tilde{O}\left(\frac{e^{3K} N^{3K-2}}{K^{3K+3}}\right))$, CMAB-SM outperforms the baseline. Further, the computation and storage complexity of the proposed algorithm are much better as compared to the baseline, as seen in Section 3.3.

In Appendix H, we further consider two more examples for Bernoulli distribution, where the sum and maximum of rewards are considered as the reward functions.

6. Conclusions and Future Work

This paper considers the problem of combinatorial multi-armed bandits with non-linear rewards, where agent chooses K out of N arms in each time-step and receives an aggregate reward. A novel algorithm, called CMAB-SM, is proposed which is shown to be computationally efficient and has a space complexity which is linear in number of base arms. The algorithm is analyzed in terms of a regret bound, and is shown to outperform the approach of considering the combinatorial action as arm for limited time horizon T. CMAB-SM provides a way to resolve two challenges in combinatorial bandits problem, the first is that the feedback is non-linear in the individual arms, and the second is that the space complexity in the previous approaches could be large due to exploding action space. CMAB-SM works efficiently for large N and K.

Followed by this work, we provided an algorithm that achieves a regret bound of $\tilde{O}(K\sqrt{NKT})$ using intuitions from this paper for the setup with non-linear Top-K subset bandits with potentially correlated rewards of individual arms Agarwal et al. (2021).

Considering non-symmetric functions of individual rewards and studying the applications of such settings, such as Social Influence Maximization, provides interesting directions for future works.

References

Yasin Abbasi-Yadkori, David Pal, and Csaba Szepesvari. Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems 24*, pages 2312–2320, 2011.

Mridul Agarwal, Vaneet Aggarwal, Christopher J Quinn, and Abhishek Umrawal. Dart: Adaptive accept reject algorithm for non-linear combinatorial bandits. In *Thirty-Fifth AAAI Conference on Artificial Intelligence*, 2021.

Jean-Yves Audibert, Sébastien Bubeck, and Gábor Lugosi. Regret in online combinatorial optimization. *Math. Oper. Res.*, 39(1):31–45, February 2014.

Peter Auer and Ronald Ortner. UCB revisited: Improved regret bounds for the stochastic multi-armed bandit problem. *Periodica Mathematica Hungarica*, 61(1-2):55–65, 2010.

Vijay S Bawa. Optimal rules for ordering uncertain prospects. *Journal of Financial Economics*, 2(1):95–121, 1975.

Lilian Besson and Emilie Kaufmann. What doubling tricks can and can't do for multi-armed bandits. *arXiv.org*, 2018.

Nicolò Cesa-Bianchi and Gábor Lugosi. Combinatorial bandits. *J. Comput. Syst. Sci.*, 78 (5):1404–1422, September 2012.

Wei Chen, Yajun Wang, and Yang Yuan. Combinatorial multi-armed bandit: General framework and applications. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 151–159, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

Varsha Dani, Thomas Hayes, and Sham Kakade. Stochastic linear optimization under bandit feedback. In *Proceedings of the 21st Annual Conference on Learning Theory*, pages 355–366, 2008a.

Varsha Dani, Sham M Kakade, and Thomas P. Hayes. The price of bandit information for online optimization. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 345–352. Curran Associates, Inc., 2008b.

Pedro Domingos and Matt Richardson. Mining the network value of customers. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 57–66. ACM, 2001.

Sarah Filippi, Olivier Cappe, Aurelien Garivier, and Csaba Szepesvari. Parametric bandits: The generalized linear case. In *Advances in Neural Information Processing Systems 23*, pages 586–594, 2010.

Yi Gai, Bhaskar Krishnamachari, and Rahul Jain. Learning multiuser channel allocations in cognitive radio networks: A combinatorial multi-armed bandit formulation. In *New Frontiers in Dynamic Spectrum, 2010 IEEE Symposium on*, pages 1–9. IEEE, 2010.

Aditya Gopalan, Shie Mannor, and Yishay Mansour. Thompson sampling for complex online problems. In *Proceedings of the 31st International Conference on Machine Learning*, pages 100–108, 2014.

Josef Hadar and William R Russell. Rules for ordering uncertain prospects. *The American Economic Review*, 59(1):25–34, 1969.

Kwang-Sung Jun, Aniruddha Bhargava, Robert Nowak, and Rebecca Willett. Scalable generalized linear bandits: Online computation and hashing. In *Advances in Neural Information Processing Systems 30*, pages 98–108, 2017.

O Krafft and N Schmitz. A note on Hoeffding's inequality. *Journal of the American Statistical Association*, 64(327):907–912, 1969.

Branislav Kveton, Zheng Wen, Azin Ashkan, Hoda Eydgahi, and Brian Eriksson. Matroid bandits: fast combinatorial optimization with learning. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, pages 420–429, 2014.

Branislav Kveton, Csaba Szepesvari, Zheng Wen, and Azin Ashkan. Cascading bandits: Learning to rank in the cascade model. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 767–776, Lille, France, 07–09 Jul 2015a. PMLR.

Branislav Kveton, Zheng Wen, Azin Ashkan, and Csaba Szepesvari. Tight regret bounds for stochastic combinatorial semi-bandits. In *Artificial Intelligence and Statistics*, pages 535–543, 2015b.

Branislav Kveton, Manzil Zaheer, Csaba Szepesvari, Lihong Li, Mohammad Ghavamzadeh, and Craig Boutilier. Randomized exploration in generalized linear bandits. In *23rd International Conference on Artificial Intelligence and Statistics*, 2020.

Chang Li, Branislav Kveton, Tor Lattimore, Ilya Markov, Maarten de Rijke, Csaba Szepesvári, and Masrour Zoghi. Bubblerank: Safe online learning to re-rank via implicit click feedback. In *Uncertainty in Artificial Intelligence*, pages 196–206. PMLR, 2020.

Lihong Li, Yu Lu, and Dengyong Zhou. Provably optimal algorithms for generalized linear contextual bandits. In *Proceedings of the 34th International Conference on Machine Learning*, pages 2071–2080, 2017.

David Liau, Zhao Song, Eric Price, and Ger Yang. Stochastic multi-armed bandits in constant space. In Amos Storkey and Fernando Perez-Cruz, editors, *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 386–394, Playa Blanca, Lanzarote, Canary Islands, 09–11 Apr 2018. PMLR.

Tian Lin, Bruno Abrahao, Robert Kleinberg, John Lui, and Wei Chen. Combinatorial partial monitoring game with linear feedback and its applications. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 901–909, Bejing, China, 22–24 Jun 2014. PMLR.

Alessandro Nuara, Francesco Trovo, Nicola Gatti, Marcello Restelli, et al. A combinatorial-bandit algorithm for the online joint bid/budget optimization of pay-per-click advertising campaigns. In *Thirty-Second AAAI Conference on Artificial Intelligence*, pages 1840–1846, 2018.

Idan Rejwan and Yishay Mansour. Top-k combinatorial bandits with full-bandit feedback. In *Algorithmic Learning Theory*, pages 752–776, 2020.

Alan B Slomson. *Introduction to Combinatorics*. CRC Press, 1997.

Raymond Chi-Wing Wong, Ada Wai-Chee Fu, and K. Wang. MPIS: Maximal-profit item selection with cross-selling considerations. In *Third IEEE International Conference on Data Mining*, pages 371–378, 2003.

Yu Xiang, Tian Lan, Vaneet Aggarwal, and Yih-Farn R Chen. Joint latency and cost optimization for erasure-coded data center storage. *IEEE/ACM Transactions on Networking (TON)*, 24(4):2443–2457, 2016.

Weinan Zhang, Ying Zhang, Bin Gao, Yong Yu, Xiaojie Yuan, and Tie-Yan Liu. Joint optimization of bid and budget allocation in sponsored search. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1177–1185. ACM, 2012.

Appendix A. Fundamental Lemmas

In this subsection, we will describe some lemmas that will be later used to prove the regret bound in Theorem 1. The first lemma is Hoeffding's Inequality, which will be used in the results in this paper.

Lemma 19 (Hoeffding's Inequality (Krafft and Schmitz, 1969)) *Let X be a random variable bounded in $[a, b]$, and let $\bar{X}$ denote the expected value of X. Further, let $\hat{\bar{X}}$ denote the average of n i.i.d. samples of X. Then, for any $\epsilon > 0$, we have*

$$P\left(\left|\bar{X} - \hat{\bar{X}}\right| \geq \epsilon\right) \quad \leq \quad 2e^{-\frac{2n\epsilon^2}{(b-a)^2}} \tag{38}$$

The next result shows that the property of FSD is preserved by strictly increasing functions.

Lemma 20 *Suppose that a random variable X has FSD over another random variable Y (or $X \succ Y$). Further, let $g'(\cdot)$ be a strictly increasing function on $\mathbb{R}$. Then, $g'(X)$ has FSD over $g'(Y)$, or*

$$g'(X) \succ g'(Y).$$

Proof Since $X \succ Y$, we have

$$P\left(X \geq x\right) \quad \geq \quad P\left(Y \geq x\right)$$

Transforming X and Y using the function g', and using the strict monotonicity of g', we have

$$P\left(g'(X) \geq g'(x)\right) \quad \geq \quad P\left(g'(Y) \geq g'(x)\right)$$

Taking $t \triangleq g'(x)$, we have,

$$P\left(g'(X) \geq t\right) \quad \geq \quad P\left(g'(Y) \geq t\right)$$

By the same arguments, if there is an x where $P\left(X \geq x\right) > P\left(Y \geq x\right)$, for $t = g'(x)$, $P\left(g'(X) \geq t\right) > P\left(g'(Y) \geq t\right)$. Thus, we have $g'(X) \succ g'(Y)$.

■

Appendix B. Proof of Lemma 10

Note we can write the actions using replacement function h to replace reward of $G(j)$ arm by the reward of $G(i)$ arm, and obtain

$$\mathbf{d}_{\mathbf{a}^G_{-j}} \quad = \quad h\left(\mathbf{d}_{\mathbf{a}^G_{-i}}, j, X_{G(i)}\right) \ \forall\, j \tag{39}$$

The expectation of the reward of an action formed using replacement can be written as expected value of the conditional expectation of the replaced reward value of arm $G(i)$. More precisely,

$$\mathbb{E}\left[f\left(\mathbf{d}_{\mathbf{a}^G_{-j}}\right)\right] = \mathbb{E}\left[\mathbb{E}\left[f\left(h\left(\mathbf{d}_{\mathbf{a}^G_{-i}}, j, X_{G(i)}\right)\right)\Big| X_{G(i)}\right]\right] \tag{40}$$

In right hand side of Equation (40), the outer expectation is taken over $X_{G(i)}$ while the inner expectation is over $\mathbf{d}_{\mathbf{a}^G_{-i}}$. In addition, we can replace the reward of an arm $G(j)$ in $\mathbf{d}_{\mathbf{a}^G_{-i}}$ by itself to obtain

$$\mathbf{d}_{\mathbf{a}^G_{-i}} = h\left(\mathbf{d}_{\mathbf{a}^G_{-i}}, j, X_{G(j)}\right) \tag{41}$$

Similarly to (40), we can write the expected reward of the action as the expectation of conditional expectation of reward of arm $G(j)$. Thus, we have

$$\mathbb{E}\left[f\left(\mathbf{d}_{\mathbf{a}^G_{-i}}\right)\right] = \mathbb{E}\left[\mathbb{E}\left[f\left(h\left(\mathbf{d}_{\mathbf{a}^G_{-i}}, j, X_{G(j)}\right)\right)\Big| X_{G(j)}\right]\right] \tag{42}$$

The ordering between actions $\mathbf{a}^G_{-i}$ and $\mathbf{a}^G_{-j}$ is defined as an order between the expected rewards of the respective actions. We assume that $\mathbf{a}^G_{-i}$ provides a higher expected reward than $\mathbf{a}^G_{-j}$, hence we have $\mu_{\mathbf{a}^G_{-i}} > \mu_{\mathbf{a}^G_{-j}}$. This further implies

$$\mathbb{E}\left[f\left(\mathbf{d}_{\mathbf{a}^G_{-i}}\right)\right] > \mathbb{E}\left[f\left(\mathbf{d}_{\mathbf{a}^G_{-j}}\right)\right] \tag{43}$$

Replacing the right hand side of (43) by (40) and the left hand side of (43) by (42), we have

$$\begin{aligned} &\mathbb{E}\left[\mathbb{E}\left[f\left(h\left(\mathbf{d}_{\mathbf{a}^G_{-i}}, j, X_{G(j)}\right)\right)\Big| X_{G(j)}\right]\right] \\ > \;& \mathbb{E}\left[\mathbb{E}\left[f\left(h\left(\mathbf{d}_{\mathbf{a}^G_{-i}}, j, X_{G(i)}\right)\right)\Big| X_{G(i)}\right]\right] \end{aligned} \tag{44}$$

We define a function $g(x)$ as the conditional expectation of reward function $f(\cdot)$ with respect to the reward of the j^{th} arm being x. More precisely,

$$g(x) \triangleq \mathbb{E}\left[f\left(h\left(\mathbf{d}_{\mathbf{a}^G_{-i}}, j, X\right)\right)\Big| X = x\right] \tag{45}$$

Replacing the conditional expectations in (44), by g as defined in (45), we get

$$\mathbb{E}_{X_{G(j)}}\left[g\left(X_{G(j)}\right)\right] > \mathbb{E}_{X_{G(i)}}\left[g\left(X_{G(i)}\right)\right] \tag{46}$$

From Assumption 3, $g(x)$ is a strictly increasing function of x, and from Assumption 2, rewards of individual arms have FSD relationships. Using Assumption 3 and Assumption 2 along with equation (46), we want to prove that arm $G(j)$ has FSD over arm $G(i)$. Let us assume the converse where arm $G(i)$ has FSD over arm $G(j)$, or $X_{G(i)} \succ X_{G(j)}$. Due to the strict increasing nature of $g(\cdot)$ and Lemma 20, we have $g\left(X_{G(i)}\right) \succ g\left(X_{G(j)}\right)$. Using Lemma 5, we further have

$$\mathbb{E}\left[g\left(X_{G(i)}\right)\right] \geq \mathbb{E}\left[g\left(X_{G(j)}\right)\right] \tag{47}$$

This inequality (47) does not agree with the inequality obtained from the original assumption (46) thus disproving $X_{G(i)} \succ X_{G(j)}$. Since every two arms have FSD relation and $X_{G(i)} \succ X_{G(j)}$ does not hold, we have

$$X_{G(j)} \succ X_{G(i)} \tag{48}$$

Hence, an ordering on the expected rewards of the $K+1$ actions constructed by leaving one arm aside, gives an ordering on the $K+1$ arms of the group.

Appendix C. Number of time steps in SORT and MERGE subroutines

In this subsection, we will bound the number of exploration time steps that are spent in SORT and MERGE subroutines. The next result bounds the number of time steps spent in a group G in the SORT subroutine to obtain an ordering on the actions and thus on arms (by Lemma 10).

Proof [Proof of Lemma 11] Let $G(i)$ for all $i \in 1, 2, \cdots, K+1$ be the different arms of group G that we aim to sort using the SORT subroutine. Let $\hat{\mu}_{\mathbf{a}^G_{-i}}$ be the estimate of action made using all arms in G except arm $G(i)$. We want to identify the number of time steps spent in SORT subroutine, and the error probability in the ordering.

The algorithm proceeds in rounds starting from $r = 1$. We define $\Delta_r \triangleq 2^{-r}$, and each unsorted action in round r is played for $n_r \triangleq \frac{\log 2NT}{\Delta_r^2}$ time steps. Using Hoeffding's Inequality, the expected reward estimate of each action lies in the range $\left[\hat{\mu}_{\mathbf{a}^G_{-i}} - \Delta_r, \hat{\mu}_{\mathbf{a}^G_{-i}} + \Delta_r\right]$ with probability bound given as

$$\begin{aligned} P\left(\left|\mu_{\mathbf{a}^G_{-i}} - \hat{\mu}_{\mathbf{a}^G_{-i}}\right| \geq \Delta_r\right) &\leq 2e^{-2\frac{\log 2NT}{\Delta_r^2}\Delta_r^2} \\ &= \frac{1}{2N^2T^2} \end{aligned} \tag{49}$$

Thus, the expected reward estimate of each action lies in the range $\left[\hat{\mu}_{\mathbf{a}^G_{-i}} - \Delta_r, \hat{\mu}_{\mathbf{a}^G_{-i}} + \Delta_r\right]$ with probability at least $1 - \frac{1}{2N^2T^2}$. Let it take r_i rounds to be able to sort $\mathbf{a}^G_{-i}$ in its correct position, which implies that all other actions can be well separated from this action. When two actions are separated, the upper confidence bound for the action with a lower estimated reward is less than the lower confidence bound of action with higher estimated reward, which gives the following inequality

$$\hat{\mu}_{\mathbf{a}^G_{-i}} + \Delta_{r_i} < \hat{\mu}_{\mathbf{a}^G_{-j}} - \Delta_{r_i}.$$

This further means that the actions $\mathbf{a}^G_{-i}$ and $\mathbf{a}^G_{-j}$ were inseparable in round $r_i - 1$, thus the following holds.

$$\hat{\mu}_{\mathbf{a}^G_{-i}} + \Delta_{r_i-1} \geq \hat{\mu}_{\mathbf{a}^G_{-j}} - \Delta_{r_i-1} \tag{50}$$

However, using Lemma 19, $\hat{\mu}_{\mathbf{a}^G_{-i}}$ lies between the confidence region around the true mean as $\hat{\mu}_{\mathbf{a}^G_{-i}} \in \left[\mu_{\mathbf{a}^G_{-i}} - \Delta_{r_i-1}, \mu_{\mathbf{a}^G_{-i}} + \Delta_{r_i-1}\right]$. The same is true for $\hat{\mu}_{\mathbf{a}^G_{-j}}$. Using the upper limits of estimated mean rewards, we can rewrite inequality (50) as

$$\mu_{\mathbf{a}^G_{-i}} + 2\Delta_{r_i-1} \geq \mu_{\mathbf{a}^G_{-j}} - 2\Delta_{r_i-1} \tag{51}$$

From inequality (51), we get Δ_{r_i} in terms of the difference of expected rewards of the two arms as follows.

$$\begin{aligned} 4\Delta_{r_i-1} &\geq \mu_{\mathbf{a}^G_{-j}} - \mu_{\mathbf{a}^G_{-i}} \\ &\geq \frac{1}{U}\left(\mathbb{E}\left[X_{G(i)}\right] - \mathbb{E}\left[X_{G(j)}\right]\right), \end{aligned} \tag{52}$$

where the last step follows from Corollary 9. Using the definition of Δ_r and the upper bound obtained in inequality (52), we have

$$\begin{aligned} 2^{-(r_i-1)} &= \Delta_{r_i-1} \\ &\geq \frac{\left(\mathbb{E}\left[X_{G(i)}\right] - \mathbb{E}\left[X_{G(j)}\right]\right)}{4U} \end{aligned}$$

Taking logarithm with base 2, we obtain a lower bound on the number of rounds needed as

$$r_i - 1 \leq \log_2\left(\frac{4U}{\left(\mathbb{E}\left[X_{G(i)}\right] - \mathbb{E}\left[X_{G(j)}\right]\right)}\right).$$

The term $\left(\mathbb{E}\left[X_{G(i)}\right] - \mathbb{E}\left[X_{G(j)}\right]\right)$ will be lowest for $j = i+1$ or $i-1$ since the arms closest to i will result in the lowest gap. Thus, we define δ_i for arm $G(i)$ as

$$\begin{aligned} \delta_i \triangleq \min\big\{&\mathbb{E}\left[X_{G(i)}\right] - \mathbb{E}\left[X_{G(i+1)}\right], \\ &\mathbb{E}\left[X_{G(i-1)}\right] - \mathbb{E}\left[X_{G(i)}\right]\big\} \end{aligned} \tag{53}$$

The number of rounds to correctly rank arm $G(i)$ can thus be upper bounded as follows.

$$r_i \leq \log_2\left(\frac{8U}{\delta_i}\right).$$

Having bounded the number of rounds to rank $\mathbf{a}^G_{-i}$, we find the number of time steps each action is played till round r_i as follows.

$$\begin{aligned} n_{r_i} &= \frac{\log 2NT}{\Delta^2_{r_i}} \\ &= 2^{2r_i}\log 2NT \\ &\leq 2^{2\log_2\left(\frac{8U}{\delta_i}\right)}\log 2NT \\ &= \frac{64U^2\log 2NT}{\delta_i^2} \end{aligned}$$

This provides the number of times action $\mathbf{a}^G_{-i}$ is played. Let λ be chosen as a threshold, which is the precision level below which we cannot correctly order two actions. Using λ as a lower bound for δ_i, each of the $K+1$ action is chosen for n_{r_i} times, where δ_i is replaced by $\max(\delta_i, \lambda)$. Thus, the total time steps any of the action is selected is given as

$$n' \leq \sum_{i=1}^{K+1}\frac{64U^2\log 2NT}{\max(\delta_i^2, \lambda^2)}$$

which proves the number of time steps as in the statement of the Lemma.

For $K+1$ arms, there can be $\binom{K+1}{2}$ ordered pairs denoting the edges of a complete graph, where each edge is the ordering of the two vertices. However, the ordering is preserved by a sub-graph which is a chain by just keeping the edges which connect two immediately ordered

vertices. The number of edges thus becomes $(K+1)-1=K$. Thus, there are K orderings in a group, and all of them should be correct to make the algorithm work. The probability of error in sorting group G can then be written as

$$\begin{aligned} & P\left(\{\text{any arm is incorrectly sorted}\}\right) \\ = \; & 1-P\left(\{\text{all arms are correctly sorted}\}\right) \\ \leq \; & 1-\left(1-\frac{1}{2N^2T^2}\right)^K \\ < \; & 1-\left(1-\frac{K}{2N^2T^2}\right) \\ = \; & \frac{K}{2N^2T^2}, \end{aligned}$$

where the probability that any two arms are incorrectly sorted is bounded by $\frac{1}{2N^2T^2}$ as given in (49). This proves the probability of correct ordering as in the statement of the Lemma. ■

Having understood the number of time steps spent in the SORT subroutine, and the error probability of ranking arms in each group, we now consider the MERGE subroutine. In the following lemma, we find the number of time steps it takes to merge two groups, with an error probability on the ordering in the combined group.

Proof [Proof of Lemma 12] In the MERGE subroutine, we use the sorted groups G_1 and G_2 to construct a group G of K elements such that arms in group G are sorted. We maintain two counters i and j for the groups G_1 and G_2 respectively. We replace arm $G_2(j)$ by $G_1(i)$ in group G_1 to create a new action. We play both actions in rounds starting from $r=1$. At the end of round r, the deviation of estimated mean reward and true expected rewards is $\Delta_r \triangleq 2^{-r}$. By round r, each action has been played for $n_r \triangleq \frac{\log 2NT}{\Delta_r^2}$ time steps similar to Algorithm 2. Two actions are separated when the upper confidence bound of worse action is lesser than the lower confidence bound of better action. We then add the arm corresponding to better action to group G and increment the counter for the corresponding group from which arm was picked and continue comparing new actions. This is continued till we have K arms in G.

Since the fundamental concept of comparison of two actions in Algorithm 2 is the same as that in Algorithm 3, similar analysis follows and the number of time steps required to merge the groups is

$$n' \;\leq\; \sum_{i=1}^{K+1} \frac{64U^2\log 2NT}{\max\left(\delta^2_{G(i)},\lambda^2\right)},$$

where $G(K+1)$ is the last arm with which comparison was made but not added to group G. The same argument as in the proof of Lemma 11 can be used to bound the error probability of a single run of Algorithm 3 by $\frac{K}{2N^2T^2}$. ■

Appendix D. Proof of Lemma 13

To bound the probability of error of Algorithm 1, we define the event $\mathcal{E}$ which is the event when the algorithm makes an error. The algorithm makes an error when either the SORT

subroutine, the MERGE subroutine, or both make an error. So, we can write the error event as a union of error events in sorting and error events in merging. Let $\mathcal{E}_s$ the event where at least one of the $\frac{N}{K+1}$ calls made to Algorithm 2 resulted in an incorrect list, and $\mathcal{E}_m$ be the event where at least one of the $\frac{N}{K+1} - 1$ merges is in error.

$$\mathcal{E} \quad = \quad \mathcal{E}_s \cup \mathcal{E}_m$$

Since the probability of the union of events is upper bounded by the sum of probabilities of individual events, we get

$$P(\mathcal{E}) \quad \leq \quad P(\mathcal{E}_s) + P(\mathcal{E}_m) \tag{54}$$

We now identify upper bounds for $P(\mathcal{E}_s)$ and $P(\mathcal{E}_m)$ by breaking down the error events into error in each call to SORT and MERGE subroutine.

Let us define an event $\mathcal{E}_{s,i}$ which denotes that there was an error in the sorted list given by Algorithm 2 for i^{th} group, or

$$\mathcal{E}_{s,i} \quad \triangleq \quad \Big\{\text{at least two arms are incorrectly placed in sorting } i^{th} \text{ group}\Big\} \tag{55}$$

Hence, $\mathcal{E}_s$ is a union of $\mathcal{E}_{s,i}$ for all $i \in \{1, 2, \cdots, \frac{N}{K+1}\}$, or $\mathcal{E}_s = \bigcup_{i=1}^{\frac{N}{K+1}} \mathcal{E}_{s,i}$. Probability of error in any of the sorting operations is thus given as

$$\begin{aligned}
P(\mathcal{E}_s) \quad &= \quad P\left(\bigcup_{i=1}^{\frac{N}{K+1}} \mathcal{E}_{s,i}\right) && (56)\\
&\leq \quad \sum_{i=1}^{\frac{N}{K+1}} P(\mathcal{E}_{s,i}) && (57)\\
&< \quad \sum_{i=1}^{\frac{N}{K+1}} \frac{K}{2NT^2} && (58)\\
&= \quad \frac{N}{K+1}\frac{K}{2N^2T^2} && (59)\\
&< \quad \frac{1}{2NT^2} && (60)
\end{aligned}$$

where (56), and (57) follow from the definition of events and the union bound, respectively, and (58) follows from Lemma 11.

Similarly we define event $\mathcal{E}_{m,i}$ representing that Algorithm 2 incorrectly merges the merged group up to group i and $(i+1)^{th}$ group. Let G_M^i be the merged sorted group formed by merging sorted groups G_M^{i-1} and G_i for $i > 1$, with $G_M^1 = G_1$. Then, we have

$$\mathcal{E}_{m,i} = \left\{\text{error occurred while merging } G_M^i \text{ and } G_{i+1}\right\}.$$

We note that $\mathcal{E}_m$ is a union of $\mathcal{E}_{m,i}$ for all $i \in \{1, 2, \cdots, \frac{N}{K+1} - 1\}$, or $\mathcal{E}_m = \bigcup_{i=1}^{\frac{N}{K+1}-1} \mathcal{E}_{m,i}$. Probability of error in the MERGE subroutine is given as

$$\begin{aligned} P(\mathcal{E}_m) &= P\left(\bigcup_{i=1}^{\frac{N}{K+1}-1} \mathcal{E}_{m,i}\right) && (61)\\ &\leq \sum_{i=1}^{\frac{N}{K+1}-1} P(\mathcal{E}_{m,i}) && (62)\\ &< \sum_{i=1}^{\frac{N}{K+1}-1} \frac{K}{2N^2T^2} && (63)\\ &< \frac{N}{K+1}\frac{K}{2N^2T^2} && (64)\\ &< \frac{1}{2NT^2} && (65) \end{aligned}$$

where (61) and (62) follow from the definition of events and union bound, respectively, and (63) follows from Lemma 12.

Substituting bounds obtained on $P(\mathcal{E}_m)$ and $P(\mathcal{E}_s)$ in (54), we have

$$\begin{aligned} P(\mathcal{E}) &< \frac{1}{2NT^2} + \frac{1}{2NT^2} && (66)\\ &= \frac{1}{NT^2} && (67) \end{aligned}$$

The total error probability of the algorithm is thus bounded by $\frac{1}{NT^2}$, proving the statement of the Lemma.

Appendix E. Proof of Lemma 14

Exploration in Algorithm 1 is done using SORT and MERGE subroutines, so we will analyze the maximum time taken by the two subroutines. To sort all groups, Algorithm 1 runs SORT $\frac{N}{K+1}$ times, and to merge the groups, Algorithm 1 runs $\frac{N}{K+1} - 1$ times. Let i^{th} run of SORT uses $T_{SORT,i}$ time steps, and j^{th} run of MERGE uses $T_{MERGE,j}$ time steps. The total number of time-steps used for exploration can be written as

$$\begin{aligned} T_{exp} &= \sum_{s=1}^{\frac{N}{K+1}} T_{SORT,s} + \sum_{m=1}^{\frac{N}{K+1}-1} T_{MERGE,m}\\ &\leq \sum_{s=1}^{\frac{N}{K+1}} \max_s (T_{SORT,s}) + \sum_{m=1}^{\frac{N}{K+1}-1} \max_m (T_{MERGE,m})\\ &= \frac{N}{K+1}\max_s (T_{SORT,s})\\ &\quad + \left(\frac{N}{K+1} - 1\right)\max_m (T_{MERGE,m}) \end{aligned}$$

We use Lemma 11 to find $\max_s (T_{SORT,s})$ as follows.

$$\begin{aligned}
& T_{SORT,s} \\
= & \sum_{j=1}^{K+1} \frac{64U^2 \log 2NT}{(\max(\lambda, \delta_{s,j}))^2} \\
\leq & \sum_{j,\delta_{s,j}>\lambda}^{K+1} \frac{64U^2 \log 2NT}{\lambda^2} + \sum_{j,\delta_{s,j}<\lambda}^{K+1} \frac{64U^2 \log 2NT}{\lambda^2} \\
= & \sum_{j=1}^{K+1} \frac{64U^2 \log 2NT}{\lambda^2} \\
= & \frac{64(K+1)U^2 \log 2NT}{\lambda^2}.
\end{aligned}$$

Similarly, we use Lemma 12 to find $\max_s (T_{MERGE_s})$ as follows.

$$\begin{aligned}
& T_{MERGE,m} \\
= & \sum_{j=1}^{K+1} \frac{64U^2 \log 2NT}{(\max(\lambda, \delta_{m,j}))^2} \\
\leq & \sum_{j,\delta_{m,j}>\lambda}^{K+1} \frac{64U^2 \log 2NT}{\lambda^2} + \sum_{j,\delta_{m,j}<\lambda}^{K+1} \frac{64U^2 \log 2NT}{\lambda^2} \\
= & \sum_{j=1}^{K+1} \frac{64U^2 \log 2NT}{\lambda^2} \\
= & \frac{64(K+1)U^2 \log 2NT}{\lambda^2}.
\end{aligned}$$

Total exploration time can now be bounded by using the values for maximum time taken for SORT and MERGE as,

$$\begin{aligned}
& T_{exp} \\
\leq & \frac{N}{K+1} \max_s (T_{SORT,s}) + \left(\frac{N}{K+1} - 1\right) \max_s (T_{SORT,s}) \\
\leq & \frac{N}{K+1} \frac{64(K+1)U^2 \log 2NT}{\lambda^2} \\
& + \left(\frac{N}{K+1} - 1\right) \frac{64(K+1)U^2 \log 2NT}{\lambda^2} \\
< & \frac{N}{K+1} \frac{64(K+1)U^2 \log 2NT}{\lambda^2} \\
& + \left(\frac{N}{K+1}\right) \frac{64(K+1)U^2 \log 2NT}{\lambda^2} \\
\leq & \frac{64NU^2 \log 2NT}{\lambda^2} + \frac{64NU^2 \log 2NT}{\lambda^2}
\end{aligned}$$

$$\leq \frac{128NU^2 \log 2NT}{\lambda^2}.$$

Appendix F. Proof of Lemma 15

We will bound the expected regret where the suboptimal action is returned by CMAB-SM, which includes an error from Algorithm 2 or from Algorithm 3. Let the chosen action be $\hat{\mathbf{a}}^* = (\hat{a}_1^*, \cdots, \hat{a}_K^*)$ and optimal action be $\mathbf{a}^* = (a_1^*, \cdots, a_K^*)$. Then, the gap in the actions is $P = \sqrt{\min_\Pi \{\sum_{i=1}^K (\mathbf{d}_{\hat{a}_i^*} - \mathbf{d}_{a^*_{\Pi(i)}})^2\}}$, where Π is any possible permutation of $\{1, \cdots, K\}$. Also, let Π^* be the permutation that optimizes the above minimization. Using Assumption 4, and U defined in Corollary 9, we have

$$\mathbb{E}\left[r_{\mathbf{a}^*} - r_{\hat{\mathbf{a}}^*}\right] \leq UP \tag{68}$$

We consider two possible cases. Case 1 corresponds to the scenario where for some i, $\mathbb{E}\left[\mathbf{d}_{\hat{a}_i^*}\right] - \mathbb{E}\left[\mathbf{d}_{a^*_{\Pi^*(i)}}\right] > \lambda$. The second case is when for all i, $\mathbb{E}\left[\mathbf{d}_{\hat{a}_i^*}\right] - \mathbb{E}\left[\mathbf{d}_{a^*_{\Pi^*(i)}}\right] \leq \lambda$.

Case 1: For some i, $\mathbb{E}\left[\mathbf{d}_{\hat{a}_i^*}\right] - \mathbb{E}\left[\mathbf{d}_{a^*_{\Pi^*(i)}}\right] > \lambda$. In this case, the incorrect action has arms which could have been separated without hitting the threshold λ, but did not do so because of an error in the SORT or MERGE subroutine. We note that since $\mathbb{E}\left[\mathbf{d}_{\hat{a}_i^*}\right]$ and $\mathbb{E}\left[\mathbf{d}_{a^*_{\Pi^*(i)}}\right]$ are both in $[0, 1]$, $P \leq \sqrt{K}$. The expected regret at time t can be written as

$$\begin{aligned}
\mathbb{E}\left[R(t)\right] &= \mathbb{E}\left[r_{\mathbf{a}^*} - r_{\hat{\mathbf{a}}^*} \middle| \mathbf{a}_t \neq \mathbf{a}^*\right] \times P\left(\{\mathbf{a}_t \neq \mathbf{a}^*\}\right) && (69)\\
&\leq U\sqrt{K} \times P\left(\{\mathbf{a}_t \neq \mathbf{a}^*\}\right) && (70)\\
&\leq \frac{U\sqrt{K}}{NT^2}, && (71)
\end{aligned}$$

where (70) follows from (68) with $P \leq \sqrt{K}$, and (71) follows from Theorem 13.

Case 2: For all i, $\mathbb{E}\left[\mathbf{d}_{\hat{a}_i^*}\right] - \mathbb{E}\left[\mathbf{d}_{a^*_{\Pi^*(i)}}\right] \leq \lambda$. In this case, the SORT or MERGE subroutines will not be able to differentiate between the two actions. Thus, $P\left(\{\mathbf{a}_t \neq \mathbf{a}^*\}\right)$ will no longer be bounded by $1/NT^2$ in this case, since the individual rewards are bounded by λ, we have $P \leq \lambda\sqrt{K}$. Thus, we have

$$\begin{aligned}
\mathbb{E}\left[R(t)\right] &= \mathbb{E}\left[r_{\mathbf{a}^*} - r_{\hat{\mathbf{a}}^*} \middle| \mathbf{a}_t \neq \mathbf{a}^*\right] \times P\left(\{\mathbf{a}_t \neq \mathbf{a}^*\}\right) && (72)\\
&\leq U\lambda\sqrt{K} \times P\left(\{\mathbf{a}_t \neq \mathbf{a}^*\}\right) && (73)\\
&\leq U\lambda\sqrt{K}, && (74)
\end{aligned}$$

where (73) follows from (68) and $P \leq \lambda\sqrt{K}$.

Combining both the cases, the maximum regret the algorithm can incur is bounded by $U\lambda\sqrt{K} + \frac{U\sqrt{K}}{NT^2}$, thus proving the result.

Appendix G. Complete Implementation of SORT and MERGE subroutines

The detailed SORT and MERGE subroutines can be seen in Algorithm 4-6.

Algorithm 4 SORT

procedure SORT(G, λ, T, N, K) ▷ Group of $K+1$ arms, separation threshold
  Initialize $g_i := \mathbf{a}^G_{-i} \ \ \forall\ 1 \le i \le K+1$ ▷ Rename to reduce notation clutter
  Initialize $unsorted \leftarrow G$ ▷ Set of unsorted arms
  Initialize $G^*[1:K+1] \leftarrow 0$ ▷ Ranked list of arms initialized with zeros
  Initialize $t[1:K+1] \leftarrow 0$ ▷ Array to store number of times each action is played
  Initialize $\hat{\mu}_{g_i} \leftarrow 0$; $r \leftarrow 1$; $\Delta_r \leftarrow \frac{1}{2}$ $n_r = \frac{2\log(TNK)}{\Delta^2}$
  while $(\Delta_r > \lambda)$ **and** $unsorted \neq \phi$ **do**
    for $i \in unsorted$ **do**
      $\hat{\mu}_{g_i}, t[i] =$ UPDATE_MEAN$(\hat{\mu}_{g_i}, g_i, t[i], n_r)$ ▷ Exploration happens here
    end for
    $j = \{\arg\text{ sort}(\hat{\mu}_{g_i})\}$
    for $1 \le k \le K+1$ **do**
      if $\hat{\mu}_{g_{j[k]}} < \hat{\mu}_{g_{j[k-1]}} - 2\Delta_r$ **and** $\hat{\mu}_{g_{j[k]}} > \hat{\mu}_{g_{j[k+1]}} + 2\Delta_r$ **then**
        $G^*[K+1-(k-1)] = G[j[k]]$; $unsorted = unsorted \setminus \{j[k]\}$
      end if ▷ Perform only relevant checks for $k \in \{1, K+1\}$
    end for
    $r = r+1$; $\Delta_r = \frac{\Delta_{r-1}}{2}$; $n_r = \frac{2\log(TNK)}{\Delta^2}$ ▷ Update round parameters
  end while
  for $k \in unsorted$ **do**
    $G^*[K+1-k] = G[j[k]]$; $unsorted = unsorted \setminus \{j[k]\}$
  end for
  return $G^*[1:K]$ ▷ Optimal K arms increasing sorted increasingly in expected individual rewards
end procedure

Appendix H. Synthetic Evaluation Results for Bernoulli Rewards

In this section, we evaluate CMAB-SM under multiple synthetic problem settings. We compare the result with improved UCB algorithm as described in (Auer and Ortner, 2010). Since this paper provides the first result with non-linear reward functions for CMAB problem with bandit feedback, we compare with the UCB algorithm (Auer and Ortner, 2010) which is optimal for small N and K while having the regret scale with $\binom{N}{K}$.

For evaluations, we ran the algorithm for $T = 10^6$ time steps and averaged over 30 runs. We compare cumulative regret at each t starting from $t = 0$, which is defined as,

$$W(t) = \sum_{t'=0}^{t} R(t')$$

We consider two values of $N \in \{12, 24\}$. For $N = 12$, we choose $K \in \{2, 3, 5\}$, while for $N = 24$, we choose $K \in \{2, 3, 5, 7, 11\}$. Since the arms must have FSD over each other, we describe a example single parameter distributions for the reward of each arm that have this property. Reward of arm i comes from the set $\{0, 1\}$ and follows a Bernoulli distribution with parameter p_i, or

$$X_i \ \sim \ Bern(p_i).$$

We note that arm i has FSD over arm j if $p_i > p_j$. Thus, this reward distribution satisfies Assumption 2 as long as no two arms have same parameter, or $p_i \neq p_j$ for any $i \neq j$. Figure

Algorithm 5 MERGE

procedure MERGE(G_1, G_2, λ, T, N, K) ▷ Groups of K arms each, and precision
  Initialize $G^*[1:K] \leftarrow 0$ ▷ Array of size K initialized with 0 to store the optimal arms
  Initialize $\hat{\mu}_{G_1} \leftarrow 0$; $\hat{\mu}_A \leftarrow 0$; $i \leftarrow 1$; $j \leftarrow 1$
  Initialize $t_1 \leftarrow 0$; $r_1 \leftarrow 1$; $\Delta_{r_1} \leftarrow \frac{1}{2}$; $n_{r_1} = \frac{2\log(TNK)}{\Delta_{r_1}^2}$
  for $k = 1 : K$ **do**
    Construct new action by replacing i^{th} arm of G_1 by j^{th} arm of G_2

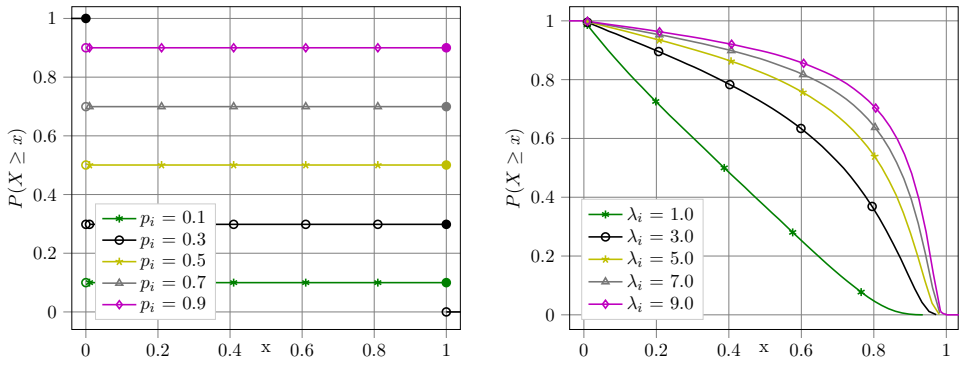
$$\mathbf{a}_{i,j} = (G_1 \setminus \{G_1(i)\}) \cup \{G_2(j)\} \tag{75}$$

    Initialize $t_2 \leftarrow 0$; $r_2 \leftarrow 1$; $\Delta_{r_2} \leftarrow \frac{1}{2}$; $n_{r_2} = \frac{2\log(TNK)}{\Delta_{r_2}^2}$ ▷ For every new action constructed
    while $(\Delta_{r_2} > \lambda)$ **and** $(G^*[k] == 0)$ **do**
      $\hat{\mu}_{\mathbf{a}^{G_1}}, t_1 =$ UPDATE_MEAN$(\hat{\mu}_{\mathbf{a}^{G_1}}, \mathbf{a}^{G_1}, t_1, n_{r_1})$ ▷ Exploration happens here
      $\hat{\mu}_{\mathbf{a}_{i,j}}, t_2 =$ UPDATE_MEAN$(\hat{\mu}_{\mathbf{a}_{i,j}}, \mathbf{a}_{i,j}, t_2, n_{r_2})$ ▷ Exploration happens here
      if $\hat{\mu}_{\mathbf{a}^{G_1}} < \hat{\mu}_{\mathbf{a}_{i,j}} - 2\Delta_{r_1}$ **then**
        $G^*[k] = G_2(j)$; $j = j + 1$
      else if $\hat{\mu}_{\mathbf{a}^{G_1}} > \hat{\mu}_{\mathbf{a}_{i,j}} + 2\Delta_{r_1}$ **then**
        $G^*[k] = G_1(i)$; $i = i + 1$
      end if
      $r_2 = r_2 + 1$; $\Delta_{r_2} = \frac{\Delta_{r_2-1}}{2}$; $n_{r_2} = \frac{2\log(TNK)}{\Delta_{r_2}^2}$
      if $r_2 > r_1$ **then**
        $r_1 = r_1 + 1$; $\Delta_{r_1} = \frac{\Delta_{r_1-1}}{2}$; $n_{r_1} = \frac{2\log(TNK)}{\Delta_{r_1}^2}$
      end if
    end while
    if $G*[k] == 0$ **then** ▷ If the arm have less separation than λ
      if $\hat{\mu}_{\mathbf{a}^{G_1}} < \hat{\mu}_{\mathbf{a}_{i,j}}$ **then**
        $G^*[k] = G_2(j)$; $j = j + 1$
      else if $\hat{\mu}_{\mathbf{a}^{G_1}} > \hat{\mu}_{\mathbf{a}_{i,j}}$ **then**
        $G^*[k] = G_1(i)$; $i = i + 1$
      end if
    end if
    $k = k + 1$
  end for
  return G^* ▷ Merged set of G_1, and G_2
end procedure

Algorithm 6 UPDATE_MEAN

procedure UPDATE_MEAN(μ, $\mathbf{a}$, t, n) ▷ Explore action $\mathbf{a}$ by playing it $n - t$ times
  while $t \leq n$ **do**
    $r_{\mathbf{a},t}$ = Reward collected by playing $\mathbf{a}$
    $\hat{\mu}_{\mathbf{a}} = \frac{t \times \hat{\mu}_{\mathbf{a}} + r_{\mathbf{a},t}}{t+1}$
    $t = t + 1$
  end while
  return $\hat{\mu}_{\mathbf{a}}, t$
end procedure

2a plots $P(X \geq x)$ of the reward function for different values of p_i. We see that $P(X \geq x)$



(a) Bernoulli Rewards (b) Transformed Rewards

Figure 2: First Order Stochastic Dominance on reward distribution

is larger for the distribution with larger value of p, and for any two different values of p, there is x (e.g., any $x \in (0, 1]$) such that $P(X \geq x)$ are not the same thus showing that the reward distributions satisfy Assumption 2.

We consider two different types of reward function to evaluate the performance of CMAB-SM. The two reward functions are the sum of rewards of each arm, and the maximum of rewards of each arm, In the following, we will describe the functions and evaluate them for one of the distributions of the rewards from each arm.

H.1. Sum of Bernoulli arm rewards

Assume that a company wishes to do a campaign of the product and chooses K out of available N sub-campaigns each day. Let the reward of sub-campaign i, X_i, be Bernoulli with parameter p_i, which is unknown. Further, let the reward that the company receives is how the overall company sales progressed, thus receiving the aggregate reward as $\sum_{i=1}^{K} X_i$. To normalize the received reward, we let $r = \frac{1}{K} \sum_{i=1}^{K} X_i$. We assume that the individual arm rewards are not observed by the company, while the overall progress of the campaign can be seen. Another application is showing K out of N advertisements to the user webpage, where the reward is in a form of whether user clicks the ad to go to the product page. The aggregate reward is the total number of clicks, and we assume that individual click information is not available.

Thus, we have the expected reward as

$$\begin{aligned} \mathbb{E}[r] &= \mathbb{E}\left[\frac{1}{K}\sum_{i=1}^{K} X_i\right] \\ &= \frac{1}{K}\sum_{i=1}^{K} \mathbb{E}\left[X_i\right] \\ &= \frac{1}{K}\sum_{i=1}^{K} p_i \end{aligned}$$

Since the combined reward is sum of all individual rewards, the expected reward is strictly increasing with respect to the expected rewards of the individual arms. Prior works in click optimization assume knowledge of clicks on individual advertisements and thus are semi-bandits (Li et al., 2020).

Recently (Rejwan and Mansour, 2020) proposed CSAR algorithm to find top K arms which solves this problem setup efficiently using Hadamard matrices. CSAR algorithm, similar to ours, divide the set of arms into groups of size $2K$. The algorithm, then estimates the individual arm rewards of the $2K$ arms in each group using Hadamard matrices of size $2K$. Each column maps to an arm in the group and for each row, an action can now be constructed by selected all arms corresponding to $+1$ elements in the row or the arms corresponding to -1 elements in the row.

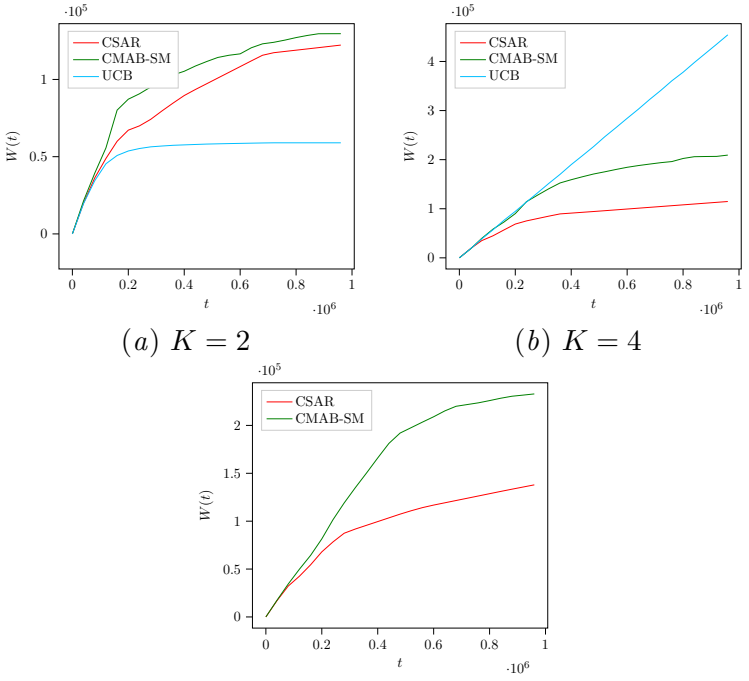
We compare our CMAB-SM with UCB algorithm and CSAR algorithm for $N = 45$ and various values of K for $T = 10^6$ steps. For CSAR algorithm we select $K \in \{2, 4, 8\}$ for easy construction of Hadamard matrices of size $2K$. For UCB algorithm we select $K \in \{2, 4\}$ for easy construction of the combinatorial action space of $\binom{N}{K}$. We plot average cumulative regret over 25 independent runs for fixed value of individual arm rewards sampled randomly uniformly from $[0, 1]$.

Figure 3 shows the comparison results between CMAB-SM, CSAR algorithm, and UCB algorithm. We note that for $K = 2$, UCB algorithm outperforms both CSAR algorithm and CMAB-SM algorithm. This is because for small values of K, UCB can explore all $\binom{N}{K}$ actions faster as UCB does not estimate individual arm rewards as CSAR algorithm, and UCB can eliminate sub optimal arms from direct comparison with the best arm unlike CMAB-SM algorithm. We note that in the cumulative regret of CSAR algorithm rises sharply for $K = 2$ in Figure 3a. We suspect that this is because of a large number of samples might be required to eliminate a sub-optimal arm. Since our implementation of the algorithm samples a group consecutively for $m = 1/2^r$ times in round r, we see a jump instead of a smooth rise. Such behaviour in not visible for $K = 4$ or $K = 8$ because we suspect that the elimination of sub-optimal arms might have been quicker compared for the case of $K = 2$ because of the large gap between arms.

Also we note that CMAB-SM does not perform as good as CSAR algorithm. We note that this difference is because CSAR estimates expected rewards of individual arms. CSAR algorithm can construct optimal action from the estimates and pull the optimal arm more frequently. CMAB-SM eliminates arms slowly compared to CSAR algorithm, as CSAR compares each arm from the K^{th} best arm which CMAB cannot perform. However, we note that even for $K = 4$, both CSAR and CMAB-SM algorithm outperform UCB algorithm by significant margin. Also, for $K = 8$, we note that UCB algorithm would not even finish exploring $\binom{45}{8} = 215553195$ arms in 10^6 time steps. Hence a comparison with UCB for $K = 8$ is futile.

H.2. Maximum of Bernoulli rewards

We consider a case where agent is a recommendation system that shows a list of restaurants or hotels, and user provides feedback whether or not the list is useful. A user finds the list useful when she is able to get a recommendation suiting her requirements. We take the reward of individual arm to be discrete with value 1 if the item was useful, and 0 for



(a) $K = 2$ (b) $K = 4$

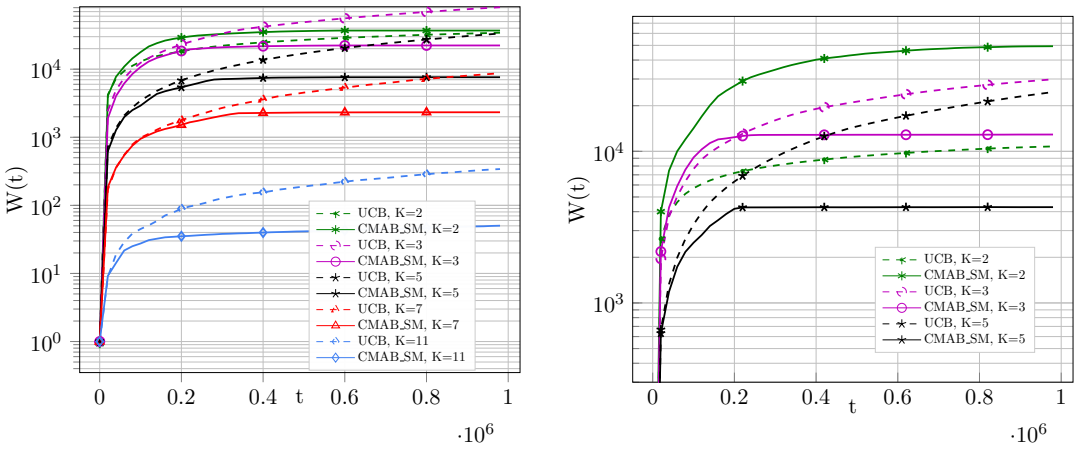
(c) $K = 8$

Figure 3: Comparison results for CMAB-SM and CSAR Algorithm for combinatorial linear bandit reward as mean of individual arm rewards.

the case where the item in list is not useful. We assume that the rewards follow Bernoulli distribution. Since the individual rewards are not observed, this is a bandit setting. Further, note that the maximum function is not a linear function. We will now show the strictly increasing property of the function. The expected reward of selecting K arms is given as

$$\begin{aligned}
\mathbb{E}[r] &= \mathbb{E}\left[\max(X_1, X_2, \cdots, X_K)\right] && (76)\\
&= 1\left(1 - P\left(\bigcap_{i=1}^{K}\{X_i = 0\}\right)\right)\\
&\quad +0\left(P\left(\bigcap_{i=1}^{K}\{X_i = 0\}\right)\right) && (77)\\
&= 1\left(1 - \prod_{i=1}^{K}(1-p_i)\right) + 0\left(\prod_{i=1}^{K}(1-p_i)\right) && (78)\\
&= 1 - \prod_{i=1}^{K}(1-p_i), && (79)
\end{aligned}$$

where (76) is the expected value of the function of individual rewards, (77) follows from the fact that individual rewards are Bernoulli distributed and their maximum is zero only when



(a) $N = 24$ (b) $N = 12$

Figure 4: Reward of actions is the maximum of rewards of individual arms as described in section H.2

all the individual rewards are zero, (78) holds since the rewards are independent of each other.

(Gopalan et al., 2014) present a unique way to solve this problem in Section 4.2 of of their paper. They consider the success probability p_i of arm i comes from the set $\{1-\beta^R, 1-\beta^{R-1}, \cdots, 1-\beta\}$ for all $i \in \{1, 2, \cdots, N\}$ and $\beta \in (0,1)$ and $R > 0$ are fixed parameters. Additionally, their bounds are of order $O\left(\binom{N-1}{K}\right)$. Our setting is a generalization of the setting considered by (Gopalan et al., 2014). We let the p_i lie in the set $[0,1]$ and we obtain a bound which is polynomial in N, K.

The reward is non-linear and the expected value of reward is strictly increasing in expected rewards of individual arms. Cascade model of click optimization by (Kveton et al., 2015a) uses a similar problem formulation, however they still consider information on clicks on individual items. Figure 4 shows the evaluation results in this case. We note that for $K \geq 3$, the proposed algorithm significantly outperforms UCB. Even for $N = 24$ and $K = 2$, where $\binom{N}{K} = 220$, $W(T)$ for CMAB-SM is close to UCB.

We note that the cumulative regret at any time t decreases as K increases. This follows from the fact that as with increasing K, a user will have more choices at any given time and it is more likely that the arm with the highest reward is in the K chosen arms.