# Budgeted Combinatorial Multi-Armed Bandits

Debojit Das
International Institute of Information
Technology (IIIT)
Hyderabad, India
debojit.das@research.iiit.ac.in

Shweta Jain
Indian Institute of Technology (IIT)
Ropar, India
shwetajain@iitrpr.ac.in

Sujit Gujar
International Institute of Information
Technology (IIIT)
Hyderabad, India
sujit.gujar@iiit.ac.in

## ABSTRACT

We consider a budgeted combinatorial multi-armed bandit setting where, in every round, the algorithm selects a *super-arm* consisting of one or more arms. The goal is to minimize the total expected regret after all rounds within a limited budget. Existing techniques in this literature either fix the budget per round or fix the number of arms pulled in each round. Our setting is more general where based on the remaining budget and remaining number of rounds, the algorithm can decide how many arms to be pulled in each round. First, we propose CBwK-Greedy-UCB algorithm, which uses a greedy technique, CBwK-Greedy, to allocate the arms to the rounds. Next, we propose a reduction of this problem to Bandits with Knapsacks (BwK) with a single pull. With this reduction, we propose CBwK-LP-UCB that uses PrimalDualBwK ingeniously. We rigorously prove regret bounds for CBwK-LP-UCB. We experimentally compare the two algorithms and observe that CBwK-Greedy-UCB performs incrementally better than CBwK-LP-UCB. We also show that for very high budgets, the regret goes to zero.

## KEYWORDS

Combinatorial MAB; Budgeted MAB; Crowdsourcing; Heterogeneous Arms

## 1 INTRODUCTION

In a classical stochastic Multi-Armed Bandit (MAB), there are a set of available arms (bandits), and each arm, when pulled, generates a reward from an unknown distribution. A typical goal is to design an algorithm to optimally pull one arm at each round to learn the mean of their reward distributions while maximizing their earned reward. The underlying concept behind multi-armed bandit problems is the trade-off between *exploration* and *exploitation*. The dilemma is as follows: exploring an arm may lead to playing suboptimal arms, whereas simply exploiting the arm with the highest reward may lead to wrongly identifying some arm as the optimal arm. Therefore, striking a balance is crucial. There is a vast amount of literature for a classical stochastic multi-armed bandit, and optimal algorithms exist to solve the problem up to a constant factor. Among the frequentist approaches, UCB1 algorithm by Auer et al. [2] achieves a regret matching with the lower bound regret for classical bandit

setting [18]. Whereas in Bayesian approaches, Thompson [25] proposed a Thomson Sampling algorithm which obtains the optimal regret bounds up to a constant factor [1, 14].

Though the classical MAB problem is important, practical applications generally require different variants. For instance, online shopping filters provide some contexts in the form of filters to improve the quality of the search results. Here, Contextual MAB (Li et al. [19]) are more suited than the classical MAB. In online advertising, the platform needs to shuffle through ads, and certain ads may not be available at certain times of the day. Researchers model this problem as a Sleeping MAB (Kleinberg et al. [17]).

This paper considers an essential variant, namely *Combinatorial Bandits with Knapsack constraint* (CBwK). In this variant, the algorithm also incurs a cost from pulling an arm apart from realizing a stochastic reward from an unknown distribution. CBwK variant combines the two well-known variants in MABs, namely, Combinatorial MAB (CMAB) (Chen et al. [7]) where a subset of arms need to be pulled at each round and Bandits with Knapsack (BwK) (Badanidiyuru et al. [4]) where a single-arm needs to be pulled at each round but within a budget constraint. An algorithm for CBwK aims to pull a subset of arms at each round to maximize the reward and ensure the budget constraint. There are many applications where combinatorial bandits with knapsack constraints are required. E.g., crowdsourcing where a subset of workers are required to be selected to do a given set of tasks, and there is a fixed budget to do so; internet routing where a path consisting of many nodes needs to be selected to send the packet within the time budget; and Internet of Things where a set of sensors need to be selected at every instance and taking data from a sensor is costly.

For example, consider a crowdsourcing platform where requesters can post their tasks that need to be completed by a team of workers. Consider that the requester has a series of tasks that need to be performed and has a fixed budget at disposal to hire the workers. The requester would like to hire the best workers to complete each task, but each worker is also associated with a cost. Further, the quality of workers for the task is unknown to the requester and has to be learnt over time. Note that the budget is a strict constraint, and no workers can be selected if the budget is exhausted, even if some tasks are left. In addition to the classical exploration-exploitation dilemma, subset selection with budget constraints poses an additional challenge to the requester. That is, how many workers the requester should select at each round. If it selects too many workers in the initial rounds, learning can be made faster. However, this would also result in incurring too much cost and hence exhaustion of the budget rapidly. On the other hand – if it selects only a few workers at each round, the tasks may be completed with inferior quality. Much budget may be left at the end, which ideally the

requester could have used in hiring more workers for the tasks. Works like [12, 23] have used MABs in crowdsourcing setting.

Even though there has been extensive research on both the variants CMAB ([6, 34]) and Budgeted MAB (BMAB) ([30, 33]), not much has been explored in Budgeted Combinatorial MABs (BCMAB). There is recent work on BCMAB by Sankararaman and Slivkins [21]. However, the algorithm divides the budget uniformly over all rounds, which leads to regret of $O(\sqrt{T} \log T)$. Regret of a given algorithm is the difference between the expected reward of an optimal algorithm that has access to the unknown parameters and the expected reward of the given algorithm.

In our setting, the challenge is that no super-arm is optimal across all rounds, whereas in approaches where budget is equally divided in all rounds, there is an optimal super-arm that the algorithm tries to identify. It shows the complexity of designing an algorithm and its regret analysis. We first propose a greedy algorithm for *offline setting* where we assume knowledge of the mean rewards. With this as offline solutions, we propose CBwK-Greedy-UCB an UCB-based approach for CBwK. One can easily observe that the regret for a fixed number of rounds drops to zero if we have enough budget to explore all the arms in every round. Though CBwK-Greedy-UCB performs well, the regret analysis is elusive due to possibly selecting different super-arms every round. To resolve the regret challenges, we devise another algorithm CBwK-LP-UCB by leveraging `PrimalDualBwK` from [4] using an ingenious reduction to their settings. We provide regret guarantees for CBwK-LP-UCB. We believe that CBwK-Greedy-UCB should have better performance, but we do not have formal proof. It is interesting to note that for fixed budget, the regret of CBwK-LP-UCB is $O(\log^2 T)$ as compared to $O(\sqrt{T} \log T)$ of Sankararaman and Slivkins [21]. In summary, the following are our contributions.

## 1.1 Contribution

To the best of our knowledge, we are the first to address CBwK that does not restrict the number of arms for any round while working with an overall budget instead of fixed budget per round. In particular,

(1) We propose two algorithms, namely, CBwK-Greedy-UCB which greedily allocates the arms in each round, and CBwK-LP-UCB inspired by existing PrimalDualBWK algorithm, which works for single pull in bandits with knapsack setting Badanidiyuru et al. [4].

(2) We rigorously prove that CBwK-LP-UCB achieves a regret of $O(\sqrt{\log(n^2 T)}(\sqrt{n \cdot OPT_{LP}} + OPT_{LP}\sqrt{\frac{n}{B'}}) + n \log(n^2 T) \log T)$ (Theorem 4) which is the best so far in the area of combinatorial bandits with knapsack where the number of arms is not fixed at each round. Here, $n$ is the number of arms, $T$ is the number of rounds, $B' = min(B, T)$ and $OPT_{LP}$ is the optimal value of the LP relaxation to our problem.

(3) We experimentally show that CBwK-Greedy-UCB performs incrementally better than CBwK-LP-UCB, and both outperform existing works in the literature.

## 2 RELATED WORK

Lai and Robbins [18] initiated the work on stochastic MABs and showed that it is possible to achieve an asymptotic regret of $O(\log T)$, where $T$ is the number of rounds played. The UCB1 algorithm by Auer et al. [2] achieved regret of $O(\log T)$ uniformly over time. Since then, a lot of work has been done on stochastic MABs and their variants using the ideas from UCB1 algorithm.

Stochastic combinatorial multi-armed bandits (CMABs) were first introduced by Chen et al. [7]. The authors used an $(\alpha, \beta)$-approximation oracle that takes the distributions of arms and outputs a subset of arms with probability $\beta$ generates an $\alpha$ fraction of the optimal expected reward. The concept of $(\alpha, \beta)$-approximation oracle has been used in many CMAB related works such as Chen et al. [6]. This setting was extended to linear rewards by Wen et al. [32]. Recently, the regret of Thompson sampling-based algorithm was derived for CMAB by Wang and Chen [31]. All the above works did not work with any constraints. A few works in the literature have considered combinatorial bandits setting with constraints to be satisfied in each round. For example, Jain et al. [13] considers a setting where at each round the subset of arms need to be selected to satisfy a quality constraint. Other examples of settings that have considered combinatorial bandits include [5, 8, 11, 22].

Budgeted multi-armed bandits (BMABs) were introduced in Tran-Thanh et al. [26], and the concept of Knapsack in BMABs was later elaborated on in Tran-Thanh et al. [27]. Works like Nhat et al. [20], Tran-Thanh et al. [29], Zhang et al. [35], Tran-Thanh et al. [28], Jain et al. [13], Singh et al. [24] deal with variants of BMAB, but none of them consider the combinatorial setting. Badanidiyuru et al. [4] provide a setting of Bandits with Knapsacks as a generalization of several models. Our work is closely related to this, but they also consider a single pull setting as compared to our combinatorial setting.

Even though CMABs and BMABs have been explored quite intensively, not much literature exists for both of them together. Chen et al. [6], Zhou and Tomlin [36], Gao et al. [10] provide a setting that assumes a fixed size of super-arm. The closest work to ours is by Sankararaman and Slivkins [21]. They consider a similar setting of BCMAB with an overall budget and no restriction on the size of super-arm. However, they distribute the budget uniformly over all rounds and incurs a regret of $O(\sqrt{T} \log T)$, where $T$ is the number of rounds. Uniformly distributing the overall budget can result in arbitrary worst regret when the restriction is lifted. We do not impose this restriction of consuming uniform budget over all rounds, and our CBwK-LP-UCB algorithm incurs a regret of $O(\log^2 T)$, which is a substantial improvement.

## 3 PRELIMINARIES

Let $\mathcal{N} = \{1, 2, \dots, n\}$ denote the set of arms, where each arm $i$ has a stochastic reward with fixed but unknown distribution with unknown mean $\mu_i \in [0, 1]$. We represent the vector of mean rewards as $\mu = \{\mu_1, \mu_2, \dots, \mu_n\}$. Each arm $i$ when pulled further incurs a known cost, $c_i \in [0, 1]$. There is a total of $T$ rounds available along with a total budget of $B$. The algorithm is allowed to pull any number of arms in any round $t$. For example, in a crowdsourcing setting, these arms could be workers with $\mu_i$ being quality of worker $i$ and $T$ being the total number of tasks available. We consider the additive

setting where the reward obtained by pulling the subset of arms is additive. If the algorithm pulls a super-arm $S_t \subset \mathcal{N}$ in round $t$, its expected reward is $\sum_{i \in S_t} \mu_i$. The algorithms observes reward from each arm – i.e., it obtains *semi-bandit feedback*. Let $N_i(t)$ be the number of times arm $i$ has been pulled till round $t$.

The goal of an algorithm $ALG$ is to maximize the total expected reward obtained in overall rounds within the budget. For every round $t$, a super-arm $S_t$ comprising of one or more arms is selected. The super-arm $S_t$ receives an expected reward $REW_{ALG}(S_t)$. The problem can be formulated as an Integer Programming as follows:

$$
\begin{aligned}
\max_{\{S_t\}_{t=1}^T} REW_{ALG} \quad &= \sum_{t=1}^T \sum_{i \in S_t} \mu_i \\
\text{subject to} \quad &\sum_{i=1}^n N_i(T) c_i \leq B \\
&0 \leq N_i(T) \leq T
\end{aligned} \tag{1}
$$

Optimization Problem in Equation 1 is hard even when the mean rewards $\mu_i$s are known. For $T = 1$, the problem reduces to a Knapsack problem. In the unknown reward setting, Sankararaman and Slivkins [21] assumes fixed budget per round, which approximates Optimization Problem 1 as $T$ independent Knapsack problems. However, this may perform arbitrarily bad, as we prove below.

LEMMA 1. *Let* $T > 1$ *and ALG1 be an algorithm that fixes the budget per round as* $B' < B$. *Further, let* $ALG^*$ *be the optimal algorithm. Then, the ratio* $\frac{REW_{ALG^*}}{REW_{ALG1}}$ *can be arbitrarily bad.*

PROOF. Take $\mathcal{N} = \{1, 2\}$ and the costs of the arms be $c_1 = B' + \epsilon_1$ and $c_2 = \epsilon_2$, for some arbitrarily small positive $\epsilon_1$ and $\epsilon_2$. Since budget per round is $B'$, $ALG1$ can never select arm 1 even if arm 1 has a much higher mean reward than arm 2. If the optimal algorithm $ALG^*$ selects arm 1 at least once (even if it selects no other arm), the ratio of expected rewards $\frac{REW_{ALG^*}}{REW_{ALG1}}$ be at least $\frac{\mu_1}{T\mu_2}$. As the ratio of $\frac{\mu_1}{\mu_2}$ can be arbitrarily large (for $\mu_1$ close to 1 and $\mu_2$ close to 0), fixing the budget can perform arbitrarily bad. □

PROPOSITION 2. *Let* $T > 1$ *and ALG1 be an algorithm that fixes the number of arms to be pulled upfront, independent of the problem instance. Further, let* $ALG^*$ *be the optimal algorithm. Then, the ratio* $\frac{REW_{ALG^*}}{REW_{ALG1}}$ *can be arbitrarily bad.*

PROOF. (*Proofsketch*) Suppose, an algorithm decides to pull $K = 2$ arms every round. Take $\mathcal{N} = \{1, 2\}$ and the costs of the arms be $c_1 = \epsilon$ and $c_2 = 1$ and mean values be such that $\mu_1 > \mu_2$. Let $B = T\epsilon$. Then $REW_{ALG1} = (\mu_1 + \mu_2) \frac{B}{(c_1 + c_2)}$ where as pulling arm 1 for every round gives reward of $REW_{ALG^*} = \frac{B}{c_1}\mu_1$. Thus, $\frac{REW_{ALG^*}}{REW_{ALG1}} = \frac{\mu_1(c_1 + c_2)}{c_1(\mu_1 + \mu_2)} > \frac{1}{2}(1 + \frac{1}{\epsilon})$. As $\epsilon$ can be arbitrarily small, this ratio can be arbitrarily bad. □

Lemma 1 and Proposition 2 illustrate that fixing budget or number of pulls at any round can lead to arbitrary bad reward with respect to optimal, and hence show the complexity of solving Optimization Problem 1. Since the problem is hard even for known reward setting, we first discuss the algorithms that can be used for known rewards setting.

*Note:* Optimization Problem 1 can be modelled as a Dynamic Programming (DP) problem by multiplying the costs $c_i$ by some scaling factor $s$ to integral values (accordingly the budget $B$ now becomes $sB$). Even though we will get an optimal solution from this, it will run in $O(nBTs^2)$, which is fairly slow. Furthermore, in the unknown stochastic setting, this DP will have to be called after every round, making it infeasible for practical use.

Inspired by the greedy solution of knapsack problem, we propose a greedy approach in the next subsection which select the arms with respect to bangperbuck ratio.

## 3.1 Deterministic Setting with Known $\mu$ - Greedy Approach (CBwK-Greedy)

Since modelling Optimization Problem 1 as a DP can be very slow, we make attempts towards a greedy approximation. When the rewards are given, we can reduce this problem to knapsack problem where each arm have $T$ copies and the goal is to select the a subset of these $nT$ arms available so as to maximize the reward subject to budget constraint. Note that this approach will not work in an online setting because, the decision has to be made at every time instance and an arm can be pulled atmost once at that time instance. We will see later how we can adapt the offline greedy algorithm to the online setting.

Let us first define the bangperbuck ratio of an arm. The *bangperbuck* ratio of an arm $i$ is given by the ratio of its value and cost, i.e., $\frac{\mu_i}{c_i}$. The core idea behind the greedy solution is to select arms with higher bangperbuck ratio maximum number of times possible before the arms with lower bangperbuck. The approach is as follows. Select the arm with the highest bangperbuck ratio for as many tasks as possible, without violating budget constraints. Then select the arm with the next highest bangperbuck ratio for as many tasks as possible, without violating budget constraints. Continue this until we are done with the arm with the lowest bangperbuck ratio. We present it formally in Algorithm 1.

---

**Algorithm 1:** CBwK-Greedy

1: **Input**: Number of arms $n$, budget $B$, number of rounds $T$ and the mean values of the arms $\mu = \{\mu_1, \ldots \mu_n\}$.
   **Output**: Allocation $\{N_1(T), N_2(T), \ldots, N_n(T)\}$
2: Initialize budget remaining $B_r = B, S_1 = S_2 = \ldots, S_T = \phi$
3: Sort the arms in non-increasing order of their bangperbuck ratios. Number them as $x_1, x_2, \ldots, x_n$. Hence $\frac{\mu_{x_i}}{c_{x_i}} \geq \frac{\mu_{x_{i'}}}{c_{x_{i'}}} \forall$ pairs $i < i'$.
4: **for** $i = 1, 2, \ldots, n$ **do**
5: $\quad N_{x_i}(T) = \min\{T, \lfloor \frac{B_r}{c_{x_i}} \rfloor\}$
6: $\quad B_r = B_r - c_{x_i} N_{x_i}(T)$
7: **end for**

---

We have the following remarks with respect to CBwK-Greedy algorithm:

*Remark 1:* Since CBwK-Greedy is similar to that of greedy algorithm of knapsack algorithm, it can be easily modified to achieve $\frac{REW_{ALG^*}}{REW_{\text{CBwK-Greedy}}} = 2$

*Remark 2:* CBwK-Greedy runs in O($n \log n$), dominated by the sorting step. This is much faster than the DP solution.

## 3.2 Deterministic Setting with Known $\mu$ - Modelling it as LP (CBwK-LP)

We can also write the optimization problem in Equation 1 as a linear programming problem similar to [4]. Following their works, we propose the following reduction to their single pull setting:

The new setting has $T$ rounds with every round consisting of $n$ plays: on the $i^{th}$ play, the algorithm can choose whether to pull arm $i$, or to not pull any arm. The setting is thus transformed into a single pull setting with $nT$ rounds, where it is not required to pull an arm in every round. To incorporate this reduction, we consider $n$ additional resources. We model each arm as a budget constrained resource, such that each arm can be pulled at most $T$ times, and deterministically consumes one unit per pull. Therefore, now instead of just having a cost $c_i$ associated with each arm $i$, there is a cost vector $C_i$ associated with each arm $i$. The length of $C_i$ vector will be $n + 1$, with first $n$ components corresponding to $n$ additional resource and last one denoting the cost $c_i$. Note that for each arm $i$, the $i^{th}$ component will be one, the last component will be $c_i$ and rest all will be zero.

Now, let $B' = \min(B, T)$. We scale the costs $c_i$ for each arm $i$ as well as the costs for the additional resources to make all budgets uniformly equal to $B'$. Thus, our cost matrix $M$, which is of the size $(n + 1) \times n$, can be written as:

$$M_{ji} = \begin{cases} c_i \cdot B'/B & \text{if } j = N + 1 \\ 1 \cdot B'/T, & \text{if } i = j \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

Here, $M_{ji}$ indicates cost of resource $j$ if we pull arm $i$. Now, we write this as a relaxed LP as shown in Equation 3. We also write its dual in Equation 4.

$$\max \sum_{i \in \mathcal{N}} \zeta_i \mu_i, \quad \zeta_i \in \mathbb{R},$$
$$\text{s.t.} \sum_{i \in \mathcal{N}} \zeta_i M_{ji} \leq B' \quad \forall j \in \{1, 2, \ldots, n + 1\} \quad (3)$$
$$\zeta_i \geq 0, \quad \forall i \in \mathcal{N}$$

The variables $\zeta_i$ represent the fractional relaxation for the number of rounds in which a given arm $i$ is selected. This is a bounded LP, because $\sum_{i \in \mathcal{N}} \zeta_i \mu_i \leq \sum_{i \in \mathcal{N}} \zeta_i \leq NT$. Let, the optimal value of this LP is denoted by $OPT_{LP}$. We now present the dual formulation of the problem.

$$\min B' \sum_j \eta_j, \quad \eta_j \in \mathbb{R}$$
$$\text{s.t.} \sum_j \eta_j M_{ji} \geq \mu_i, \quad \forall i \in \mathcal{N} \quad (4)$$
$$\eta_j \geq 0 \quad \forall j \in \{1, 2, \ldots, n + 1\}$$

The dual variables $\eta_j$ can be interpreted as a unit cost for the corresponding resource $j$. We refer to the algorithm to solve above dual as CBwK-LP and it is easy to see $REW_{\text{CBwK-LP}} = OPT_{LP}$ (due to LP duality).

*Remark:* In a deterministic setting, it is fairly obvious why $OPT_{LP} \geq REW_{ALG^*}$. However, it is not as trivial when reward from arm pulls are stochastic. We prove this in the following Lemma.

LEMMA 3. *$OPT_{LP}$ is an upper bound on the value of the optimal reward: $OPT_{LP} \geq REW_{ALG^*}$, where $ALG^*$ is an optimal algorithm.*

PROOF. Let $\eta^* = (\eta_1^*, \ldots, \eta_d^*)$ denote an optimal solution to Equation 4. Interpret each $\eta_j^*$ as a unit cost for the corresponding resource $j$. By strong LP duality, we have $B' \sum_j \eta_j^* = OPT_{LP}$. Dual feasibility implies that for each arm $i$, the expected cost of resources consumed when $i$ is pulled exceeds the expected reward produced. Thus, if we let $Z_t$ denote the sum of rewards gained in rounds $1, \ldots, t$ of the optimal dynamic policy, plus the cost of the remaining resource endowment after round $t$, then the stochastic process $Z_0, Z_1, \ldots, Z'_T$ is a supermartingale. Note that $Z_0 = B' \sum_j \eta_j^* = OPT_{LP}$, and $Z_{T'-1}$ equals the algorithm's total payoff, plus the cost of the remaining (non-negative) resource supply at the start of round $T'$. By Doob's optional stopping theorem, $Z_0 \geq E[Z_{T'-1}]$ and the lemma is proved. □

## 3.3 Regret

Let the optimal algorithm for Optimization Problem 1 be $ALG^*$. Its reward is given by $REW_{ALG^*}$. We define regret incurred by an algorithm $ALG$ as

$$REG_{ALG}(\mathcal{N}, T, B) = REW_{ALG^*}(\mathcal{N}, T, B) - REW_{ALG}(\mathcal{N}, T, B) \quad (5)$$

## 4 PROPOSED APPROACHES

### 4.1 CBwK-Greedy-UCB

Our first algorithm CBwK-Greedy-UCB in unknown reward setting is presented in Algorithm 2. The algorithm basically extends the greedy algorithm presented in section 3.1 so as to select a subset of arms at each round $t$.

---

**Algorithm 2:** CBwK-Greedy-UCB

1: **Input**: Number of arms $n$, budget $B$, number of rounds $T$
   **Output**: Subset of arms $S_t$ to be pulled for each round $t$.
2: Initialize remaining budget $B_r = B$
3: **for** $i$ in $\mathcal{N}$ **do**
4:   **if** $c_i \leq B_r$ **then**
5:     Pull arm $i$ and update UCB value of arm $i$, $N_i(t) = 1$, and $B_r = B_r - c_i$
6:   **end if**
7: **end for**
8: **for** rounds $t = 2, 3, \ldots, T$ **do**
9:   Call CBwK-Greedy($n, B_r, T - t, UCB_1, UCB_2, \ldots, UCB_n$) to get the allocation $N_1(T - t), N_2(T - t), \ldots, N_n(T - t)$.
10:   **for** arm $i$ in $\mathcal{N}$ **do**
11:     **if** $N_i(T - t) \geq 1$ and $c_i \leq B_r$ **then**
12:       $S_t = S_t \cup \{i\}, B_r = B_r - c_i$
13:     **end if**
14:   **end for**
15:   Pull all the arms in $S_t$ and update UCB values of the arms $i$ and $N_i(t) = N_i(t) + 1 \; \forall i \in S_t$.
16: **end for**

---

Steps 3-7 correspond to the first round. In the first round, we select every arm in the super-arm as long as they can be pulled without exceeding the budget. Steps 8-15 correspond to the remaining $T - 1$ round. At the start of each round, we use the greedy approach discussed in section 3.1 on the remaining budget and the remaining number of tasks to get the number of times each arm should be selected, if the UCB estimates of each arm were their real mean values. Any arm that the greedy approach does not pull in the remaining tasks is not pulled by CBwK-Greedy-UCB for the next round. All the remaining arms are pulled as long as they can be pulled without exceeding the budget. The idea here is that if the greedy algorithm does not select an arm even once, it either has very low bangperbuck ratio (as per the UCB estimate until that round), or has a higher cost than the remaining budget. In either case, it makes sense to not select the arm for that round.

*Note:* We do not prove a regret bound for CBwK-Greedy-UCB as the analysis gets quite tricky since it is not easy to estimate the size of super-arm $S_t$ selected in any round $t$.

## 4.2 CBwK-LP-UCB

The intuitive idea for the CBwK-LP-UCB algorithm is to greedily select arms with the highest estimated bangperbuck ratio. Here, the bangperbuck ratio for a given arm $i$ is defined as $\mu_i/(\eta^* C_i)$, where the denominator represents the expected cost of pulling this arm. The algorithm CBwK-LP-UCB is formally stated in Algorithm 3.

---

**Algorithm 3:** CBwK-LP-UCB

---

1: **Input**: Number of arms $n$, budget $B$, number of rounds $T$
   **Output**: Allocation $\{S_1, S_2, \ldots, S_T\}$
2: $S_t = \phi \ \forall t$
3: **Initialization**
4: In the first round, pull arm $i$ on the $i^{th}$ play i.e. $S_1 = S_1 \cup \{i\}$.
5: $v_1 = \mathbf{1} \in [0, 1]^{n+1}$
6: Set $\epsilon = \sqrt{\frac{\ln{(n+1)}}{\min{(B,T)}}}$
7: **for** rounds $t = 2, 3, \ldots, T$ **do**
8:    **for** each arm $i \in \mathcal{N}$ **do**
9:       Compute UCB estimate for the expected reward, $u_{t,i} \in [0, 1]$
10:       Expected cost for one pull of arm $i$ is estimated by $EstCost_i = C_i \cdot v_t$
11:    **end for**
12:    **for** play $i = 1, 2, \ldots, n$ **do**
13:       $S_t = S_t \cup \{i\}$ if there is enough budget remaining to pull arm $i$ after all the estimated better arms (arms with higher estimated bangperbuck) have been pulled for the remaining rounds
14:       If pulled, update estimated unit cost for each resource $j$: $v_{t+1}(j) = v_t(j)(1 + \epsilon)^{M_{ji}}$
15:    **end for**
16: **end for**

---

Steps 1 and 2 corresponds to the first round where we pull the $i^{th}$ arm in the $i^{th}$ play. Steps 3 and 4 set the parameters. Steps 5-14 correspond to the remaining $(T - 1)$ rounds. In steps 6-9, for each arm, we update its UCB estimate and expected cost of pulling that

arm. In steps 10-14 we decide whether to pull an arm or not by prioritizing arms with higher bangperbuck ratios. If it gets pulled, we update its estimated cost accordingly. $v_t$ gives an estimate of $\eta^*$ after set of rounds $t$.

CBwK-LP-UCB is inspired from PrimalDualBwK algorithm for single pull setting. Here, we list down the key differences among the two algorithms.

*Key Differences between CBwK-LP-UCB and PrimalDualBwK.*

- The fundamental difference between CBwK-LP-UCB and Primal-DualBwK is that every round in CBwK-LP-UCB has $n$ plays whereas a round in PrimalDualBwK has only one play. CBwK-LP-UCB can also choose to smartly not pull any arm in a play even if it is possible to, but PrimalDualBwK always pulls an arm every round as long as budget permits.
- The main challenge in adapting PrimalDualBwK for combinatorial setting, and why it is not a trivial extension with number of rounds as $nT$, is as follows: Once an arm $i$ is selected by PrimalDualBwK in round $t$, it will once again be available for selection in round $t + 1$. However, since several consecutive plays fall in the same round for CBwK-LP-UCB, if arm $i$ is selected in round $t$ for play $p$, it will not be available for any other play $p'$ in round $t$. We tackle this by smartly choosing to not pull any arm in certain plays. An arm is considered for pull exactly once per round, and will only be selected if there is enough budget remaining to pull it after all the estimated better arms (arms with higher estimated bangperbuck) have been pulled for the remaining rounds.
- Another important distinction is that, in CBwK-LP-UCB, the UCB values of all the arms are updated together at the start of every round, and they do not change with every play.

Note that, our main contribution is the reduction of our combinatorial setting into the single pull setting provided by Badanidiyuru et al. [4] such that the generated single pull solution works in original setting flawlessly. CBwK-LP-UCB is a modified version of their PrimalDualBwK algorithm. With such ingenious mapping, the regret proof becomes similar to that of [4]. We use their techniques to suits our combinatorial setting and present the regret analysis for CBwK-LP-UCBand prove a regret bound of the form:

$$OPT_{LP} - REW_{\text{CBwK-LP-UCB}} \leq f(OPT_{LP}) \tag{6}$$

where $f(\cdot)$ is a linear function that depends only on parameters $(B', n, T')$. Regret bound (6) implies the claimed regret bounds relative to $REW_{ALG^*}$ because

$$REW_{\text{CBwK-LP-UCB}} \geq OPT_{LP} - f(OPT_{LP}) \geq REW_{ALG^*} - f(REW_{ALG^*}) \tag{7}$$

where the second inequality follows trivially because $g(x) = max(x - f(x), 0)$ is a non-decreasing function of $x$ for $x \geq 0$ for a linear $f(\cdot)$, and $OPT_{LP} \geq REW_{ALG^*}$ from Lemma 3. From Equation 7 and Equation 5,

$$REG_{\text{CBwK-LP-UCB}} = REW_{ALG^*} - REW_{\text{CBwK-LP-UCB}} \leq f(REW_{ALG^*}) \tag{8}$$

## 4.3 Differences with SemiBwK-RRS

Even though Sankararaman and Slivkins [21] also consider a budgeted combinatorial setting with semi-bandit feedback, the key difference lies in the fact that they assume a fixed budget per round.

They solve an LP in each round which considers a pre-determined fixed budget for every round. We have already shown in Lemma 1 that this can perform arbitrarily bad. We use budget in each round adaptively, thus leading to better regret bounds.

In the next section, we bound our regret and prove Equation 8.

## 5 REGRET ANALYSIS OF CBWK-LP-UCB

We start this section by introducing some notations that will be used in our regret analysis. Then we formally present our result in Theorem 4. For the sake of clarity in notations, let $d = n + 1$ be the dimension of the cost vector (represented as $C_i$ for arm $i$). Take distribution $y_t$ as a vector of normalized costs of resources, i.e., $y_t(j) = \frac{v_t(j)}{\sum_{j=1}^d v_t(j)}$. Let us take $W$ as the total expected normalized cost consumed by the algorithm after the first round (hence, $W = \sum_{t=2}^T \sum_{i \in S_t} y_t^T C_i$). $S_t$ : the set of arms selected in their respective pulls in round $t$. With this, we claim the regret guarantee for CBwK-LP-UCB:

**Theorem 4.** *The regret of algorithm CBwK-LP-UCB with parameter $\epsilon = \sqrt{\frac{\ln d}{B'}}$, for $d = n + 1$, satisfies*

$OPT_{LP} - REW_{CBwK-LP-UCB} \leq$

$O(\sqrt{\log(n^2 T)})(\sqrt{n \cdot OPT_{LP}} + OPT_{LP}\sqrt{\frac{n}{B'}}) + O(n)\log(n^2 T)\log(T)$

**Proof.** (Overview) Using useful result adapted from Kleinberg [15], in Section 5.1, we bound the term $OPT_{LP} - f(OPT_{LP})$ for the deterministic (but unknown) setting. In Section 5.2, we extend this for the unknown stochastic setting, as described in Equation 8, completing the proof.

□

Step 12 in Algorithm 3 uses the multiplicative weights update technique by Freund and Schapire [9]. It is an online technique for maintaining a $d$-dimensional probability vector $y$ while observing a sequence of $d$-dimensional payoff vectors $\pi_1, \ldots, \pi_\tau$. We use the following related result adapted from Kleinberg [15].

**Proposition 5.** *Fix any parameter $\epsilon \in (0, 1)$ and any stopping time $\tau$. For any sequence of payoff vectors $\pi_1, \ldots, \pi_\tau \in [0, 1]^d$, we have*

$$\forall y \in \Delta[d] \qquad \sum_{t=1}^\tau y_t^T \pi_t \geq (1 - \epsilon) \sum_{t=1}^\tau y^T \pi_t - \frac{\ln d}{\epsilon}$$

### 5.1 Deterministic Rewards

In this subsection, we consider the setting where pulling an arm $i$ deterministically generates reward $\mu_i$. We bound the regret incurred if the arms are pulled as per the algorithm CBwK-LP-UCB.

The payoff vector in any round $t > 1$, is given by $\pi_t = \sum_{i \in S_t} C_i$. Take the total cost consumed by Algorithm 3 as $W = \sum_{t=2}^T \sum_{i \in S_t} y_t^T \cdot C_i$. We want to maximise this $W$.

To see why $W$ is worth maximizing, let us relate it to the total reward collected by the algorithm in rounds $t > 1$ denoted by $REW_{CBwK-LP-UCB} = \sum_{t=2}^T rew_t$, where $rew_t$ is the reward collected in the round $t$. We will prove in Lemma 7 that $REW_{CBwK-LP-UCB} \geq W \cdot \frac{OPT_{LP}}{B'}$. For this reason, maximizing $W$ also helps maximize $REW$.

Let $\zeta^*$ denote an optimal solution of the primal linear program (LP-primal). Then $OPT_{LP} = \mu^T \zeta^*$ denote the optimal value of that LP.

**Claim 6.** *We claim that there exists a $z_t$, such that*

$$z_t \in \arg \max_{z \in \Delta[\mathcal{N}]} \frac{\mu^T z}{y_t^T M z} \qquad (9)$$

**Proof.** $z_t$ is a distribution that maximizes the bangperbuck ratio among all distributions $z$ over arms. Indeed, the argmax in Equation (9) is well-defined as that of a continuous function on a compact set. Say it is attained by some distribution $z$ over arms, and let $\rho \in R$ be the corresponding max. By maximality of $\rho$, the linear inequality $\rho y_t^T M z \geq \mu^T z$ also holds at some extremal point of the probability simplex $\Delta[\mathcal{N}]$, i.e. at some point-mass distribution. For any such point-mass distribution, the corresponding arm maximizes the bang-per-buck ratio in the algorithm.

□

**Lemma 7.** $REW_{CBwK-LP-UCB} \geq W \cdot \frac{OPT_{LP}}{B'}$

**Proof.**

$$y_t^T \pi_t = y_t^T M z_t \leq \frac{rew_t(y_t^T M \zeta^*)}{OPT_{LP}}$$

$$W \leq \frac{1}{OPT_{LP}} \sum_{t=2}^T rew_t(y_t^T M \zeta^*)$$

$$= \frac{1}{OPT_{LP}} \sum_{t=2}^T (rew_t y_t^T) M \zeta^*$$

Now, let $\bar{y} = \frac{1}{REW_{CBwK-LP-UCB}} \sum_{t=1}^T rew_t \cdot y_t \in [0, 1]^d$ be the rewards weighted average of distributions $y_2, \ldots, y_T$, it follows that

$$W \leq \left( \frac{REW_{CBwK-LP-UCB}}{OPT_{LP}} \right) \bar{y}^T M \zeta^* \leq \left( \frac{REW_{CBwK-LP-UCB}}{OPT_{LP}} \right) B'$$

The last inequality follows because all components of $M\zeta^*$ are at most $B'$ by the primal feasibility of $\zeta^*$.

Hence $REW_{CBwK-LP-UCB} \geq W \cdot \frac{OPT_{LP}}{B'}'$

□

Combining Lemma 7 and the regret bound from Proposition 5, we obtain $\forall y \in \Delta[d]$

$$REW_{CBwK-LP-UCB} \geq W \frac{OPT_{LP}}{B'} \geq \left[ (1 - \epsilon) \sum_{t=2}^T y^T M z_t - \frac{\ln d}{\epsilon} \right] \frac{OPT_{LP}}{B'}$$

The algorithm stops after round $T$. By this round, either $B$ (from the original setting in Optimization Problem 1) has been fully consumed, or some arm has been selected $T$ times. Hence the consumption of some resource $j$ is at least $B'$. In a formula: $\sum_{t=1}^T y_t^T M z_t \geq B'$. Since the total cost for any resource is at max $n$ for $t = 1$, $\sum_{t=2}^T y_t^T M z_t \geq B' - n$. Hence,

$$REW_{CBwK-LP-UCB} \quad \geq [(1 - \epsilon)(B' - n) - \frac{\ln d}{\epsilon}] \cdot \frac{OPT_{LP}}{B'}$$

$$\geq OPT_{LP} - [\epsilon B + n + \frac{\ln d}{\epsilon}] \cdot \frac{OPT_{LP}}{B'}$$

$$= OPT_{LP} - O(\sqrt{B \ln d} + n) \cdot \frac{OPT_{LP}}{B}$$

$$\text{where } \epsilon = \sqrt{\frac{\ln d}{B'}} \text{ and } d = n + 1$$

## 5.2 Stochastic Rewards

Here, we use the techniques from the previous subsection and provide regret bound for the stochastic setting (Equation 8).

The algorithm computes UCBs on expected rewards $u_{t,i} \in [0, 1]$, for each arm $i$ after every round $t$. The vector $u_t \in [0, 1]^n$ represents these UCBs, where $i^{th}$ component equals $u_{t,i}$. Let $M$ be the resource-consumption matrix. That is, $M \in [0, 1]^{d \times n}$ denotes the matrix whose $(j, i)^{th}$ entry $M_{ji}$ is the actual consumption of resource $j$ if arm $i$ were chosen.

As in the Section 5.1, we claim there exists a $z_t$ such that

$$z_t \in \arg \max_{z \in \Delta N} \frac{u_t^T z}{y_t^T M z} \qquad (10)$$

As before, $z_t$ is a distribution that maximizes the bangperbuck ratio among all distributions $z$ over arms.

We define a confidence radius $rad(x, N) = \sqrt{\frac{x C_{rad}}{N}} + \frac{C_{rad}}{N}$, where $C_{rad} = \theta(\log ndT)$. We adapt the following result from Kleinberg et al. [16] and Babaioff et al. [3].

PROPOSITION 8. *Consider some distribution with values in [0, 1] and expectation $x$. Let $\hat{x}$ be the average of $N$ independent samples from this distribution. Then $\forall C_{rad} > 0$*

$$Pr[|x - \bar{x}| \le rad(\bar{x}, N) \le 3rad(x, N)] \ge 1 - \exp -\omega(C_{rad})$$

Using Proposition 8 and our choice of $C_{rad}$, it holds with probability at least $1 - T^{-1}$ that the confidence interval for every latent parameter, in every round of execution, contains the true value of that latent parameter. We call this high-probability event a *clean execution* of CBwK-LP-UCB. Our regret guarantee will hold deterministically assuming that a clean execution takes place. The regret can be at most $T$ when a clean execution does not take place, and since this event has probability at most $T^{-1}$ it contributes only $O(1)$ to the regret. Now, we assume a clean execution of CBwK-LP-UCB.

CLAIM 9. *In a clean execution of Algorithm PrimalDualBwK with parameter $\epsilon = \sqrt{\frac{\ln d}{B}}$, the algorithm's total reward satisfies the bound*

$$OPT_{LP} - REW_{CBwK-LP-UCB} \le 2OPT_{LP}(\sqrt{\frac{\ln d}{B'}} + \frac{n}{B}) + n + |\sum_{t=2}^{T} \delta_t^T z_t|$$

*where $d = n + 1$ and $\delta_t^T = u_t - \mu$ for each round $t$.*

Let $\zeta^*$ denote an optimal solution of the primal linear program given in Equation 3, and let $OPT_{LP} = \mu^T \zeta^*$ denote the optimal value of that LP. Let $REW_{UCB} = \sum_{t=2}^{T} u_t^T z_t$ denote the total payoff the algorithm would have obtained, after its initialization phase, if the actual payoff at time $t$ were replaced with the upper confidence bound.
As before, $\sum_{t=1}^{T} y_t^T M z_t \ge B'$. Once again, since the total cost for any resource is at most $n$ for $t = 1$, $\sum_{t=2}^{T} y_t^T M z_t \ge B' - n$
Let $\bar{y} = \frac{1}{REW_{UCB}} \sum_{t=2}^{T} (u_t^T z_t) y_t$. Assuming a clean execution,

$$B' \ge \bar{y}^T M \zeta^* = \frac{1}{REW_{UCB}} \sum_{t=2}^{T} (u_t^T z_t)(y_t M \zeta^*)$$

$$\ge \frac{1}{REW_{UCB}} \sum_{t=2}^{T} (u_t^T \zeta^*)(y_t M z_t)$$

$$\ge \frac{1}{REW_{UCB}} \sum_{t=2}^{T} (r^T \zeta^*)(y_t M z_t)$$

$$\ge \frac{OPT_{LP}}{REW_{UCB}} \left[ (1 - \epsilon) y^T \left( \sum_{t=2}^{T} M z_t \right) - \frac{\ln d}{\epsilon} \right]$$

$$\implies REW_{UCB} \ge \frac{OPT_{LP}}{B'} [B' - \epsilon B' - n - \frac{\ln d}{B}]$$

$$\implies REW_{UCB} \ge OPT_{LP}[1 - \epsilon - \frac{n}{B} - \frac{\ln d}{\epsilon B}]$$

The algorithm's actual payoff, $REW_{CBwK-LP-UCB} = \sum_{t=1}^{T} \mu^T z_t$, satisfies the inequality

$$REW_{CBwK-LP-UCB} \ge REW_{UCB} - \sum_{t=2}^{T} (u_t - \mu)^T z_t$$

$$= REW_{UCB} - \sum_{t=2}^{T} \delta_t^T z_t$$

We use the following proposition from Badanidiyuru et al..

PROPOSITION 10. *(Badanidiyuru et al. [4]) Consider two sequences of vectors $a_1, \ldots, a_\tau$ and $b_1, \ldots, b_\tau$, in $[0, 1]^n$, and a vector $a_0 \in [0, 1]^n$. For each arm $i$ and each round $t > 1$, let $\bar{a}_{t,i} \in [0, 1]$ be the average observed outcome up to round $t$, i.e., the average outcome $a_{s,i}$ over all rounds $s \le t$ in which arm $i$ has been chosen by the algorithm; let $N_{t,i}$ be the number of such rounds. Assume that for each arm $i$ and all rounds $t$ with $1 < t < T$, we have*

$$|b_{t,i} - a_{0,i}| \le 2rad(\bar{a}_{t,i}, N_{t,i}) \le 6rad(a_{0,i}, N_{t,i}),$$

$$|\bar{a}_{t,i} - a_{0,i}| \le rad(\bar{a}_{t,i}, N_{t,i}).$$

*Let $A = \sum_{t=1}^{T} a_{t,i}$ be the total outcome collected by the algorithm. Then,*

$$|\sum_{t=2}^{T} (b_t - a_t)^T z_t| \le O(\sqrt{C_{rad} n A} + C_{rad} n \log T).$$

Taking $a_t = \mu$, $b_t = u_t$ and vector $a_0 = \mu$ in Proposition 10, we get

$$|\sum_{t=2}^{T} \delta_t z_t| \le O(\sqrt{C_{rad} n REW_{CBwK-LP-UCB}} + C_{rad} n \log T). \qquad (11)$$

We now combine the results to provide the proof for Theorem 4.

PROOF. (Theorem 4) For $n \ge \frac{B'}{\log dT}$, the bound in Claim 9 is trivially true. Therefore we can assume without loss of generality that $n < \frac{B'}{\log dT}$. We observe that

$$OPT_{LP} \left( \sqrt{\frac{\ln d}{B'}} + \frac{n}{B'} \right) = O \left( \sqrt{n \log (ndT)} \frac{OPT_{LP}}{\sqrt{B}} \right)$$

The term $n$ on the right side of the bound in Claim 9 is bounded above by $n \log (dnT)$.

The theorem follows by plugging in $C_{rad} = \theta(\log ndT)$, and $d = n + 1 = \theta(n)$, in Equation 11, along with Claim 9. □

## 6 SIMULATION-BASED EXPERIMENTS

In this section, we compare our proposed algorithms CBwK-Greedy-UCB and CBwK-LP-UCB with the existing SemiBwK-RRS algorithm for CBMAB problem. We begin with explaining the experimental setting and then analyse the results obtained.

(a) wrt $B$, for fixed $T$
(b) wrt $T$, for fixed $B$
(c) wrt $T$, for fixed $\frac{B}{T}$
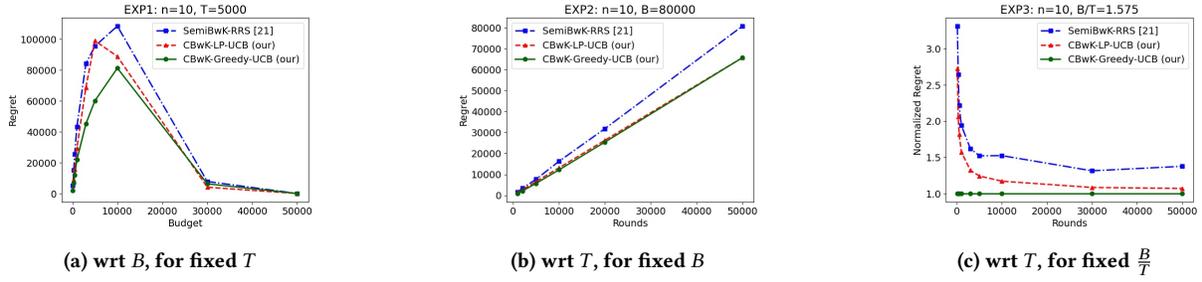
Figure 1: Regret Comparison under different settings when the rewards and costs of the arm are drawn independently from identical distribution

## 6.1 Experimental Set-up

For simulation of arms, we generate mean values and costs as follows: $\forall i \in \mathcal{N}, \mu_i \sim U[0,1], c_i \sim U[0,1]$. We take $\alpha = 5$ for CBwK-Greedy-UCBas well as SemiBwK-RRS. We take $\epsilon = 0$ in SemiBwK-RRS, to maintain consistency with what Sankararaman and Slivkins [21] mentioned in their experiments. We report average over 100 randomly generated instances for each of the following experiments.

EXP1: Varying $B = 100 \rightarrow 50000$ for $n = 10$ and $T = 5000$ to study effect of the budget on regret.

EXP2: We vary $T = 1000 \rightarrow 50000$ for fixed budget $B = 80000$ and $n = 10$ to study effect of increased rounds on regret.

EXP3: We vary $T = 1000 \rightarrow 50000$ for fixed budget/round ratio $B/T = 1.575, n = 10$ to study effect of the increased task (with proportional budget increase) on regret.

EXP4: For non i.i.d. arms (Figure 2), we vary $T = 100 \rightarrow 2000$ for fixed budget/round ratio $B/T = 1.575$ and $n = 4$ to study effect of the increased task (with proportional budget increase) on regret when the arms rewards are selected as follows: $\mu_1, c_1 \sim U[0.9, 1], \mu_2, c_2 \sim U[0.6, 0.8], \mu_3, c_3 \sim U[0.2, 0.4]$, and $\mu_4, c_4 \sim U[0, 0.1]$. In practical scenarios these arms can be seen as high, medium, low, very low rewarding arms.
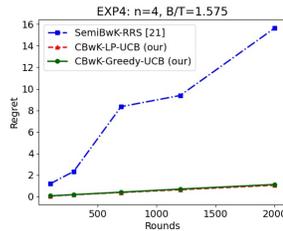


Figure 2: Regret w.r.t. $T$ for fixed $B/T$, non-i.i.d. arms

## 6.2 Empirical Analysis

As can be seen from the figures that CBwK-Greedy-UCB achieves the lowest regret in all four experiments. CBwK-LP-UCB performs almost as good as CBwK-Greedy-UCB, whereas SemiBwK-RRS performs a lot worse. The difference in the algorithms is less evident in EXP1 and EXP2 as regret dominantly depends on $OPT_{LP}$, for all three algorithms and and $OPT_{LP}$ increases with increase

in $B$ as well as $T$. Thus, to study relative performance of the algorithms as $T$ increase, we plot regrets relative to CBwK-Greedy-UCB (CBwK-Greedy-UCB normalized to 1) for EXP3. It clearly indicates superiority of our algorithms by not fixing the budget per round. We see that the regrets of CBwK-Greedy-UCB and CBwK-LP-UCB are 35% better than that of SemiBwK-RRS in EXP1, EXP2 and EXP3, and multiple times better in EXP4. The standard deviation for CBwK-Greedy-UCB was also not very high. For EXP1, EXP2 and EXP3, the Coefficient of Variance for CBwK-Greedy-UCB was noted to be below 28%, whereas it reached around 42% in the worst case for SemiBwK-RRS.

*Regret for High Budget.* For budget values $B' >= T \sum_{i=1}^{n} c_i$, we get a regret of 0. It is because the optimal solution consists of selecting every arm for every round. In both CBwK-Greedy-UCB as well as CBwK-LP-UCB, we choose to not select an arm only when there is not enough budget to select it after the arms with higher bangperbuck have been selected. If the budget is enough to select all arms for all rounds, that scenario will never occur. Hence, the super-arms selected by CBwK-Greedy-UCB, CBwK-LP-UCB and optimal solution will be exactly the same in every round.

## 7 CONCLUSION

We considered a Budgeted Combinatorial Multi-Armed Bandit setting with semi-bandit feedback. The existing literature has a fixed number of arm pulls (generally single pull) or a fixed pre-determined budget per round. We focused on the more general setting without any such restrictions. We first proposed CBwK-Greedy, which uses a greedy technique to solve this in the offline setting. We showed why the problem is difficult to solve. We provided a reduction to the knapsack problem by creating $T$ copies of each arm and showed that our greedy algorithm is 2-approx. We designed CBwK-LP-UCB using ingenious reduction to PrimalDualBwK by Badanidiyuru et al. [4]. We provided regret bound for CBwK-LP-UCB in the unknown stochastic setting. We compared our work with SemiBwK-RRS, the closest work to our setting, and experimentally demonstrated that CBwK-Greedy-UCB and CBwK-LP-UCB outperform SemiBwK-RRS.

We believe an exciting direction to explore would be a non-additive setting such as a submodular combination of rewards. One can extend our work to variants such as sleeping bandits and contextual bandits. We believe our paper is the first step in this direction as the setting we have considered is relatively unexplored.

# REFERENCES

[1] S. Agrawal and N. Goyal. Analysis of thompson sampling for the multi-armed bandit problem. In *Conference on learning theory*, pages 39–1. JMLR Workshop and Conference Proceedings, 2012.

[2] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47(2–3):235–256, May 2002. ISSN 0885-6125.

[3] M. Babaioff, S. Dughmi, R. Kleinberg, and A. Slivkins. Dynamic pricing with limited supply. *ACM Trans. Econ. Comput.*, 3(1), Mar. 2015. ISSN 2167-8375.

[4] A. Badanidiyuru, R. Kleinberg, and A. Slivkins. Bandits with knapsacks. *J. ACM*, 65(3), Mar. 2018. ISSN 0004-5411.

[5] S. Bhat, S. Jain, S. Gujar, and Y. Narahari. An optimal bidimensional multi-armed bandit auction for multi-unit procurement, 2015.

[6] L. Chen, J. Xu, and Z. Lu. Contextual combinatorial multi-armed bandits with volatile arms and submodular reward. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[7] W. Chen, Y. Wang, and Y. Yuan. Combinatorial multi-armed bandit: General framework and applications. In S. Dasgupta and D. 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.

[8] A. Deva, K. Abhishek, and S. Gujar. A multi-arm bandit approach to subset selection under constraints, 2021.

[9] Y. Freund and R. E. Schapire. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, 1997.

[10] G. Gao, J. Wu, M. Xiao, and G. Chen. Combinatorial multi-armed bandit based unknown worker recruitment in heterogeneous crowdsensing. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, pages 179–188, 2020.

[11] S. Jain and S. Gujar. A multiarmed bandit based incentive mechanism for a subset selection of customers for demand response in smart grids. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(02):2046–2053, Apr. 2020.

[12] S. Jain, G. Ghalme, S. Bhat, S. Gujar, and Y. Narahari. A deterministic mab mechanism for crowdsourcing with logarithmic regret and immediate payments. AAMAS '16, page 86–94, Richland, SC, 2016. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450342391.

[13] S. Jain, S. Gujar, S. Bhat, O. Zoeter, and Y. Narahari. A quality assuring, cost optimal multi-armed bandit mechanism for expertsourcing. *Artificial Intelligence*, 254:44–63, 2018. ISSN 0004-3702.

[14] E. Kaufmann, N. Korda, and R. Munos. Thompson sampling: An asymptotically optimal finite-time analysis. In *International conference on algorithmic learning theory*, pages 199–213. Springer, 2012.

[15] R. Kleinberg. Notes from week 2: Prediction algorithms and zero-sum games, 2007.

[16] R. Kleinberg, A. Slivkins, and E. Upfal. Multi-armed bandits in metric spaces. *CoRR*, abs/0809.4882, 2008. URL http://arxiv.org/abs/0809.4882.

[17] R. Kleinberg, A. Niculescu-Mizil, and Y. Sharma. Regret bounds for sleeping experts and bandits. *Mach. Learn.*, 80(2–3):245–272, Sept. 2010. ISSN 0885-6125.

[18] T. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Adv. Appl. Math.*, 6(1):4–22, Mar. 1985. ISSN 0196-8858.

[19] L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. *CoRR*, abs/1003.0146, 2010. URL http://arxiv.org/abs/1003.0146.

[20] V. Q. T. Nhat, S. Stein, L. Tran-Thanh, and N. Jennings. What prize is right? how to learn the optimal structure for crowdsourcing contests. In A. Nayak and A. Sharma, editors, *PRICAI 2019: Trends in Artificial Intelligence*, volume 1160, pages 85–97. Springer, August 2019.

[21] K. A. Sankararaman and A. Slivkins. Semi-bandits with knapsacks. *CoRR*, abs/1705.08110, 2017. URL http://arxiv.org/abs/1705.08110.

[22] A. D. Sarma, S. Gujar, and Y. Narahari. Multi-armed bandit mechanisms for multi-slot sponsored search auctions, 2010.

[23] Z. R. Shi, Z. S. Wu, R. Ghani, and F. Fang. Bandit data-driven optimization: AI for social good and beyond. *CoRR*, abs/2008.11707, 2020. URL https://arxiv.org/abs/2008.11707.

[24] A. Singh, P. M. Reddy, S. Jain, and S. Gujar. Designing bounded min-knapsack bandits algorithm for sustainable demand response. In *Pacific Rim International Conference on Artificial Intelligence*, pages 3–17. Springer, 2021.

[25] W. R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

[26] L. Tran-Thanh, A. C. Chapman, E. M. de Cote, A. Rogers, and N. R. Jennings. Epsilon-first policies for budget-limited multi-armed bandits. In *AAAI*, 2010.

[27] L. Tran-Thanh, A. C. Chapman, A. Rogers, and N. R. Jennings. Knapsack based optimal policies for budget-limited multi-armed bandits. *CoRR*, abs/1204.1909, 2012. URL http://arxiv.org/abs/1204.1909.

[28] L. Tran-Thanh, S. Stein, A. Rogers, and N. Jennings. Efficient crowdsourcing of unknown experts using multi-armed bandits. volume 214, 08 2012.

[29] L. Tran-Thanh, M. Venanzi, A. Rogers, and N. R. Jennings. Efficient budget allocation with accuracy guarantees for crowdsourcing classification tasks. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems*, AAMAS '13, page 901–908, Richland, SC, 2013. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450319935.

[30] F. Trovò, S. Paladino, M. Restelli, and N. Gatti. Budgeted multi–armed bandit in continuous action space. In *Proceedings of the Twenty-Second European Conference on Artificial Intelligence*, ECAI'16, page 560–568, NLD, 2016. IOS Press. ISBN 9781614996712.

[31] S. Wang and W. Chen. Thompson sampling for combinatorial semi-bandits. In *International Conference on Machine Learning*, pages 5114–5122. PMLR, 2018.

[32] Z. Wen, B. Kveton, and A. Ashkan. Efficient learning in large-scale combinatorial semi-bandits. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1113–1122, Lille, France, 07–09 Jul 2015. PMLR.

[33] Y. Xia, T. Qin, W. Ding, H. Li, X. Zhang, N. Yu, and T.-Y. Liu. Finite budget analysis of multi-armed bandit problems. *Neurocomputing*, 258, 03 2017.

[34] H. Xu, Y. Liu, W. C. Lau, and R. Li. Combinatorial multi-armed bandits with concave rewards and fairness constraints. In C. Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 2554–2560. International Joint Conferences on Artificial Intelligence Organization, 7 2020. Main track.

[35] H. Zhang, Y. Ma, and M. Sugiyama. Bandit-based task assignment for heterogeneous crowdsourcing. *CoRR*, abs/1507.05800, 2015. URL http://arxiv.org/abs/1507.05800.

[36] D. P. Zhou and C. J. Tomlin. Budget-constrained multi-armed bandits with multiple plays, 2017.