Budgeted Combinatorial Multi-Armed Bandits

Thesis Proposal submitted in partial
fulfillment of the requirements of the degree of

Master of Science
in
Computer Science and Engineering
by Research

by

Debojit Das
20171129
debojit.das@research.iiit.ac.in

Advised by Dr. Sujit P Gujar

International Institute of Information Technology
Hyderabad - 500 032, INDIA
NOVEMBER, 2023

International Institute of Information Technology

Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Budgeted Combinatorial Multi-Armed Bandits" by Debojit Das, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date

Adviser: Prof. Sujit P Gujar

To my hard work and my parents.

Acknowledgements

I would like to express my deepest gratitude to Dr. Sujit Gujar, my advisor, for his exceptional guidance, unwavering support, and invaluable mentorship throughout my research. His expertise, dedication, and willingness to address my doubts have been instrumental in shaping this research. I am also grateful for our daily post-lunch tea sessions, which provided not only a refreshing break but also a platform for insightful discussions.

I would like to extend my heartfelt appreciation to Dr. Shweta Jain for her valuable contributions to this work and for her guidance during my paper-writing process. Her expertise and constructive feedback have significantly enhanced the quality of this thesis.

I would like to thank Kumar Abhishek for his early-stage mentorship, which laid a strong foundation for my research endeavors. I am also grateful to Sambhav Solanki for our discussions on Multi-Armed Bandits (MAB), which have broadened my understanding of the subject.

I would like to give a special shoutout to Anurag, Samhita, and Shantanu for making my time at MLL (Machine Learning Lab) incredibly enjoyable. Our countless fruitful discussions and shared experiences have not only enhanced my knowledge but also forged strong friendships. You guys have been amazing! I would also like to thank Manisha, Sanjay, Sankarshan, Shaily, and Varul for casual, but useful, discussions in the lab.

I owe a massive debt of gratitude to my roommate and best friend, Suryansh, and my other best friend, Aashna. Without your unwavering support and companionship, I honestly don't know how I would have survived college. To all my extremely close friends who have been with me through thick and thin - Aaron, Arpita, Aryan, Anushka, Devesh, Freya, Kunwar, Kripa, Priyank, Rohan, Shantanu, Shashwat, Shreyas, Souvik, Sriven, Ujwal - you all mean so much to me.

A special mention to Atirek, my badminton partner extraordinaire. Our intense games and juice sessions after playing have been a significant part of my college life. I couldn't have asked for a better partner!

Last but certainly not least, my heartfelt thanks go to my incredible parents. Your unwavering support, guidance, and unconditional love have been the driving force behind my success. I am forever grateful for everything you have done for me.

Without the collective support and contributions of these individuals, this thesis would not have been possible. I am truly fortunate to have been surrounded by such incredible mentors, friends, and family throughout this journey.

Abstract

Multi-armed bandits (MABs) have various applications in real life, however, most of these applications require certain variations to the classic MAB problem. In this thesis, we focus on one such variant - budgeted combinatorial MAB (BCMAB). A BCMAB allows for multiple arms to be pulled in each round with no restrictions on the number of arms selected per round or the budget consumed per round, as long as the arms pulled do not exceed the total budget. The reward structure is taken to be additive, i.e., the reward obtained on pulling a set of arms in a round is the sum of the rewards obtained by the individual arms. We study relevant problems and develop our solutions to BCMAB. We come up with the algorithms CBwK-Greedy-UCB and CBwK-LP-UCB. We mathematically prove regret bound for CBwK-LP-UCB. We experimentally analyze our algorithms and compare them with each other and with the previous most suited algorithm.

Research Papers Based on the Thesis Work

Conference Papers

1. **Debojit Das**, Shweta Jain and Sujit Gujar. Budgeted Combinatorial Multi-Armed Bandits. Appeared in Proceedings of the 21st International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2022 Das *et al.* [9]

Contents

Chapter Page

List of Figures

Figure Page

List of Tables

Chapter 1

Introduction

"The choice to make good choices is the best choice you can choose. Fail to make that choice and on most choices you will lose." – Ryan Lilly

1.1 Motivation

Imagine you're faced with a wide array of choices every day, like where to eat, what movie to watch, or which book to read. Making the best decision can feel overwhelming, especially when you have limited time and a specific budget to work with. But what if you not only had to pick one option, but also had to figure out how to combine different choices to create the ultimate experience? That's exactly what we'll explore in this thesis.

We'll dive into the fascinating world of decision-making, uncovering the secrets to *optimizing our choices* while *dealing with budget constraints* and the complexities of *combining different options*. Think of it as solving a puzzle where we have to make smart decisions to get the most out of our resources. We'll unravel strategies and techniques that can help us navigate this puzzle and make the most of the opportunities at hand. Get ready to embark on an exciting journey where we'll unlock the art of decision-making, discovering how to make the best choices in a world full of endless possibilities.

But before we dive headfirst into the meat of our thesis, let's take a moment to explore a few more scenarios that highlight the challenge of decision-making when time and money are in limited supply.

1.1.1 Slot Machines

All right, picture this: After months of non-stop stress, you finally score a well-deserved week off from work. And guess what? Your awesome company decides to send you to Las Vegas as a thank-you for all your hard work. Now, Vegas is famous for its slot machines, and you've heard stories of people hitting the jackpot. You know that some machines give you better odds than others, and if you play your cards right, you might just be able to say goodbye to your boring desk job forever.

But hold up, here's the catch: All those slot machines look the same, making it impossible to tell which ones are worth your time. And on top of that, you've only got a week for your vacation and can only rely on the money you've saved up from your regular paycheck. How can you *maximize your chances of hitting it big, making the most of your tight budget and limited time in Vegas?*

Figure 1.1: Slot Machines

Image credit: lasvegasdirect.com

1.1.2 Routing

You've recently moved to a new city and lucked out by finding the perfect flat in an affordable and amazing neighborhood. The only downside? Your office is a trek away, and there are multiple routes you can take from your flat to get there. Naturally, you want to minimize your commute time as much as possible. But here's the kicker – the traffic on these roads is as unpredictable as the weather, with variables like time of day and day of the week affecting congestion levels. So, even if you stick to the same route, you can expect different travel times.

Now, the question is: *how will you select the most optimal route every single day to minimize time wasted over the course of a year?* You might think that considering factors like time and day of the week will likely lead to better results, but it might take longer to figure out the ideal route. Of course, these days, we have the luxury of online maps that solve this problem. But have you ever wondered how they approached this challenge?

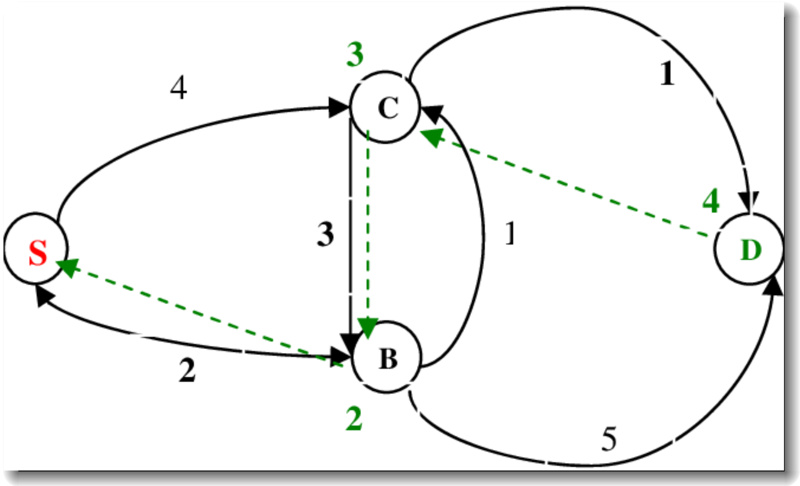


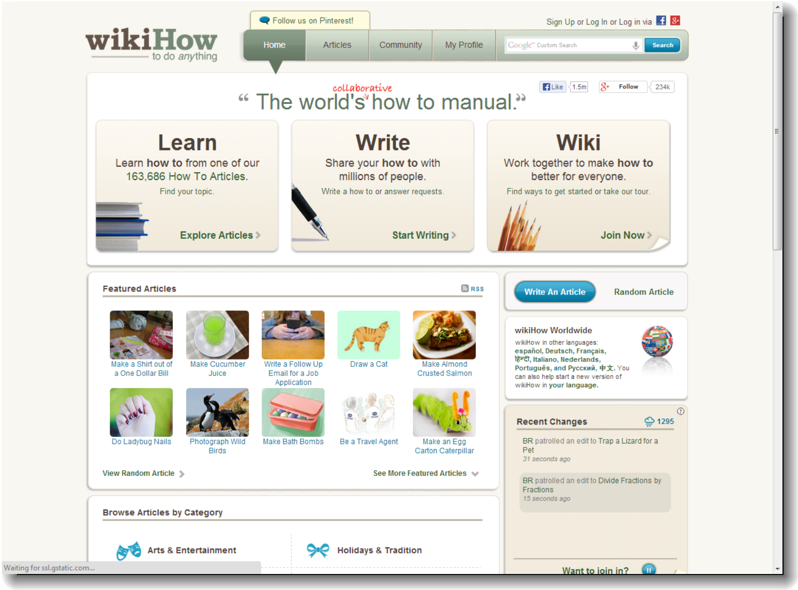
Figure 1.2: Routing

Image credit: researchgate.net

1.1.3 Advertising

This time, let's dive into the world of advertising! So, you've recently launched your website, and now you need some cash to keep it up and running smoothly. Your brilliant idea? Create slots for advertisements and charge a commission for every click those ads receive. That sounds like a win-win situation, right? But now comes the tricky part – many clients approach you with their ads, and you need to figure out which ones to display.

The question is: *how do you determine which ads make the cut?* Is it simply first-come, first-served? Well, not exactly. You might consider various factors like relevance, user preferences, and even location and time. After all, competing brands wouldn't be thrilled to have their ads displayed side by side. Users are likelier to click on an ad if they've interacted with similar ones. Hence there's a bit of a strategy involved.



“The world's collaborative how to manual.”

Learn how to from one of our 163,686 How To Articles.

Find your topic.

Share your how to with millions of people.

Write a how to or answer requests.

Work together to make how to better for everyone.

Find ways to get started or take our tour.

wikiHow in other languages: español, Deutsch, Français, हिन्दी, Italiano, Nederlands, Português, and Русский, 中文. You can also help start a new version of wikiHow in your language.

BR patrolled an edit to Trap a Lizard for a Pet
31 seconds ago

BR patrolled an edit to Divide Fractions by Fractions
15 seconds ago

Figure 1.3: Advertisements on a website

Image credit: blog.ispionage.com

1.1.4 Drug Testing

Now let's shift our focus to a critical area of concern – drug testing. Imagine you work in the healthcare ministry, and researchers have claimed to develop potential vaccinations for Covid-19. Exciting, right? But here's the catch: all these vaccines are still in their early stages, and their true potential and possible reactions are unknown. As a responsible healthcare professional, you understand the importance of making the best vaccine available to the public, but you also recognize the need for caution. Rushing into widespread administration without thorough testing could have unexpected consequences on the human body. After all, how each individual will respond to a vaccine remains fixed but unknown.

So, the question arises: *how do you propose proceeding with the vaccination drive while minimizing risks and ensuring the earliest possible availability of the vaccines?* Is there a way to strike a balance? One possible consideration is to gather context about the individuals receiving the vaccines, such as their age, weight, and gender. Could this additional information impact the results? It's worth exploring whether these factors might play a role in how well the vaccine responds to each individual.

Figure 1.4: Choosing the Right Vaccine

Image credit: coe.int

1.1.5 Sensors

Imagine you've created an app that enables users to sense and collect data. People can install this app on their devices, allowing it to gather the required data, which is then sent to a central server for processing. The accuracy of the sensed data heavily relies on the device's location and type. However, certain irregular factors like the device's charge level and heating issues can introduce some randomness or unpredictability into the observed data.

Now, here's an important aspect to consider: many of these sensors will overlap in terms of their locations. If using a sensor incurs a cost, it would be wise to select the sensors strategically to minimize redundancy in the collected data. The question then arises: *how do you decide which sensor to choose at any given time?*

This decision-making process requires careful consideration. Factors such as the desired accuracy level, the relevance of specific sensors to the data being collected, and the cost associated with using each sensor need to be weighed. By selecting sensors in a way that reduces redundancy and optimizes cost-effectiveness, you can maximize the efficiency and value of the collected data.

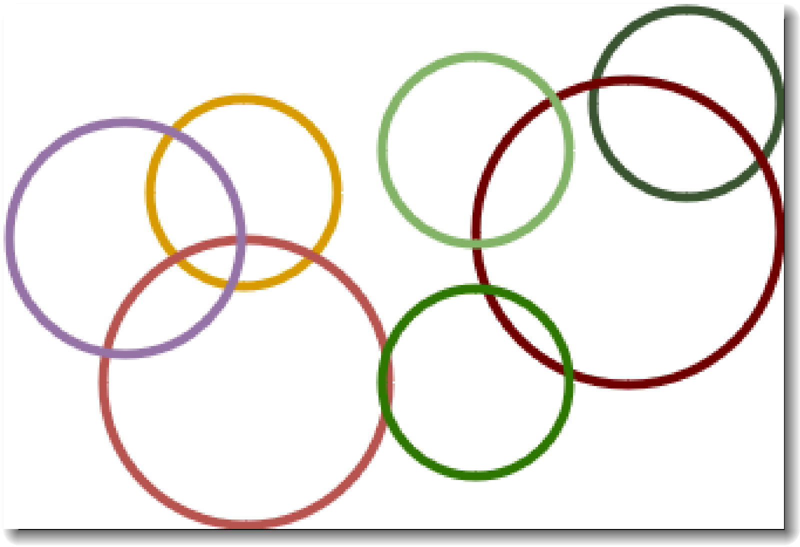


Figure 1.5: Choosing a Set of Sensors

1.1.6 Crowdsourcing

So, imagine you're not just any ordinary worker but the Founder and CEO of your very own company. One day, a major client comes knocking with a project proposal that involves a complex task. But here's the twist – you need to do the task multiple times, and after each completion, the client assigns the next one and dishes out rewards based on the quality of the work.

Naturally, the client has set a reasonable budget for the project. Now, you quickly realize that being efficient in selecting your workers is absolutely crucial. You don't want to overuse underperforming team members and risk subpar results. At the same time, you can't afford to under-spend and waste that precious budget. After all, you only have a fixed number of tasks in the contract. Talk about a balancing act, right?

The challenge is that you do not have a crystal ball to predict how well each worker will perform on this particular task. All you know is that more workers would probably yield similar or even better results. So, *how are you going to strategize and assemble the dream team for each job?*

But wait, there's more! *Not all workers are the same. They have different skill sets and areas of expertise, and certain tasks may require specific know-how. Plus, there's a possibility that the initial team you put together is already competent enough, and adding more workers, no matter how skilled they are individually, won't significantly improve the outcome.* How will your plans adapt to these changes?

Figure 1.6: Team Selection

Image credit: vectorstock.com

1.2 The Multi-Armed Bandit Problem

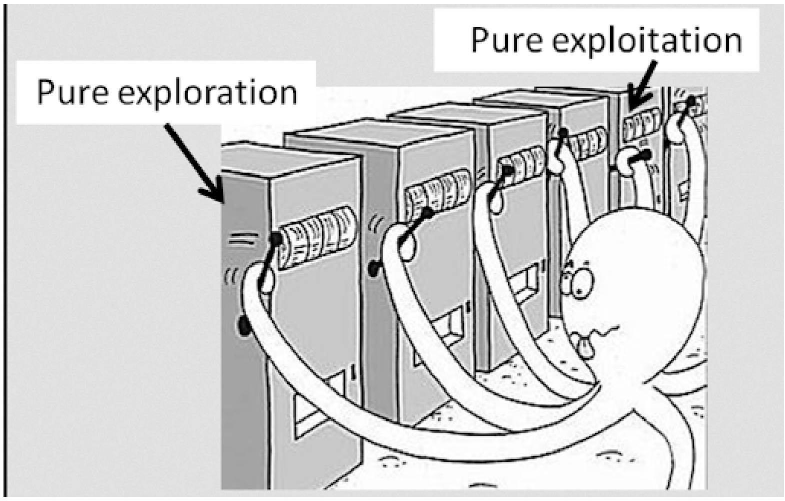


Figure 1.7: Slot Machines as an MAB

Image credit: primarydigit.com

All the six scenarios in Section 1.1 share a common characteristic: they involve multiple alternatives or options to choose from and we must also consider certain time and budget constraints. However, since the rewards associated with each choice are stochastic, we cannot determine their true quality until we try them out multiple times. Our perception of an alternative's goodness becomes more solid as we gather more information through repeated trials. However, trying out every choice may end up not opting for the best option far too often. This leads to the *Exploration-Exploitation Dilemma*. Our objective should be to select the best possible alternative as often as possible.

The Exploration-Exploitation Dilemma is the basis for the *Multi-Armed Bandit (MAB) problem.* A MAB is perhaps best explained by the scenario in Section 1.1.1. Here, *a row of slot machines would comprise the multi-armed bandit, with each machine corresponding to an arm of the bandit.*

In a MAB, we have *several arms*, each having a *fixed unknown reward value*, and our goal as the mechanism designer is to pull the arms in such a way that the *total reward is maximized over a large number of pulls.* Looking back at the scenarios in Section 1.1, all of them can be modeled as MAB problems. The parallels are given in Table 1.1.

In real-world scenarios, *additional constraints or contextual factors often come into play, influencing decision-making processes.* For example, in Section 1.1.4, we may consider factors like age and weight when selecting the appropriate drug for a person. In Section 1.1.5, we aim to choose sensors that minimize data redundancy and resource waste. In Section 1.1.6, the availability of a limited budget allows us to select multiple workers simultaneously. Incorporating such contextual information can lead to interesting variations of the vanilla multi-armed bandit (MAB) problem discussed earlier.

Over the years, extensive research has been conducted on the classic MAB problem and its popular variants. However, one relatively unexplored variation is the *Budgeted Combinatorial Multi-Armed Bandit (BCMAB)*, which we will focus on in this thesis. *BCMAB introduces a budget constraint and allows for the selection of multiple arms in each round, creating a combinatorial setting.* By addressing this variation, we aim to explore new challenges and opportunities in the field of MAB.

In the following section, we will delve deeper into the BCMAB problem, discussing its formulation, existing approaches, and the specific contributions of this thesis.

Scenario	**Parallel for arms**	**Number of pulls**	**Reward**
Slot Machines (Section 1.1.1)	Each machine is an arm	Number of pulls you can make in one week assuming a fixed rate	Maximise reward received from the slot machines
Routing (Section 1.1.2)	Each route is an arm	Number of times you will go to your office	Minimise time spent on traveling (maximize negative of time)
Advertising (Section 1.1.3)	Each ad is an arm	Number of time slots you have for the ads over a month (or any fixed duration)	Maximise the commission by maximizing the number of ads being clicked
Drug Testing (Section 1.1.4)	Each vaccine is an arm	Number of people who will take the vaccines	Maximise the safety of the public by giving them the best vaccine
Sensors (Section 1.1.5)	Each sensor is an arm	Number of time slots over a month (or any fixed duration)	Maximise the data collected
Crowdsourcing (Section 1.1.6)	Each worker is an arm	Number of jobs	Maximise the reward you get on completing a job

Table 1.1: Parallels

1.3 Problem Addressed

As discussed in the previous section, many real-world scenarios can be modeled as variations of the Multi-Armed Bandit (MAB) problem. This thesis focuses on an important, intriguing, yet quite unexplored variant known as the Budgeted Combinatorial Multi-Armed Bandit (BCMAB). This particular variant introduces a budget constraint and allows for the selection of multiple arms in each round, creating a combinatorial setting. The objective in BCMAB is to minimize the total regret incurred on pulling a subset of arms from the set of all available arms over a fixed and known number of rounds.

Regret of an algorithm is the difference in reward obtained on following the given algorithm and the reward obtained on following the optimal algorithm. Regret can be thought of as the loss the algorithm incurs by deviating from the optimal algorithm.

The BCMAB setting bears a strong resemblance to the scenarios presented in Sections 1.1.3 (Advertising), 1.1.5 (Sensors), 1.1.6 (Crowdsourcing),. In fact, our primary motivation for addressing this problem stems from the need to solve the crowdsourcing scenario introduced in Section 1.1.6. However, BCMAB tackles a more general problem encompassing a wider range of applications and decision-making scenarios.

In the crowdsourcing scenario, for instance, we have a limited budget to allocate among a pool of workers, and we aim to maximize the overall reward obtained by assigning multiple workers to different tasks. Similarly, in the advertising scenario, we have a limited number of time slots for displaying ads, and our objective is to select a combination of ads that maximizes the total commission earned through ad clicks. In the sensors scenario, we want to choose a subset of sensors within a given time frame to collect data efficiently and avoid redundancy.

The BCMAB problem provides a unified framework for addressing these challenges and more. By incorporating budget constraints and the combinatorial nature of arm selection, it offers a flexible and powerful approach to optimizing resource allocation and decision-making in various real-world scenarios.

In the upcoming sections, we will delve deeper into the BCMAB problem, exploring its formulation, existing approaches, and potential solutions. Through our research, we aim to contribute novel insights and techniques to address this challenging problem and advance the field of Multi-Armed Bandits.

The key features of the BCMAB setting are as follows:

- The BCMAB has a set of set of arms with unknown, but fixed values
- The value of an arm can only be observed on pulling the arm
- The BCMAB is played for a given number of rounds and has a given initial budget
- Pulling an arm incurs a known cost
- There are no restrictions on the number of arms being pulled in a given round, or the budget being consumed in a given round
- No arm can be pulled more than once in any given round
- If pulling an arm exceeds the remaining budget, that arm can not be pulled
- The objective is to maximize the reward across all rounds
- BCMAB has an additive reward structure for a round, i.e., the reward obtained on pulling a set of arms in a round is the sum of the rewards obtained from each of those arms

The key difference between BCMAB and its submodular extension BCMAB-S is that BCMAB-S has a monotonically non-decreasing submodular reward structure for a round, i.e., the reward obtained on pulling a set of arms in a round is at max (and generally less than) sum of the rewards obtained from each of those arms. Arms added to an existing set of arms produce diminishing gains in rewards.

1.4 Contributions

Our contributions are listed as follows:

1. To the best of our knowledge, we are the first to address a combinatorial MAB setting that does not restrict the number of arms for any round while working with an overall budget instead of a fixed budget per round.

2. We proposed a greedy algorithm CBwK-Greedy (Algorithm 8) which solves the BCMAB problem if the values of each arm were known.

3. Based on CBwK-Greedy, we proposed the algorithm CBwK-Greedy-UCB (Algorithm 9), which solves the BCMAB problem.

4. Inspired by the existing PrimalDualBWK Badanidiyuru *et al.* [4] algorithm, which works for single pull in budgeted MAB, we proposed the algorithm CBwK-LP-UCB (Algorithm 10) for our BCMAB setting.

5. We rigorously prove that CBwK-LP-UCB achieves a regret of $O(\log^2 T)$ (Theorem 4.1) which is the best so far in the area of BCMAB. Here, T is the number of rounds.

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

1.5 Organisation of the Thesis

Chapter 1 serves as an introductory chapter, laying the foundation for the thesis by presenting motivating scenarios related to the multi-armed bandit problem. It not only familiarizes readers with the problem at hand but also outlines the specific problem that this thesis aims to address. Moreover, it highlights the unique contributions and novel approaches proposed in this work.

In Chapter 2, the focus shifts to the fundamental concepts necessary to comprehend the addressed problem and appreciate the contributions of this thesis. This chapter delves into the intricacies of the basic multi-armed bandit problem, exploring its various formulations and popular variants. Additionally, it provides an extensive survey of existing literature on these variants, further reinforcing the importance and relevance of the proposed research.

Chapter 3 takes a deep dive into the core of the thesis, tackling the specific problem of Budgeted Combinatorial Multi-Armed Bandit (BCMAB) and its submodular extension, BCMAB-S. It presents a comprehensive framework for solving these problems, discussing the design considerations, methodologies, and algorithms developed to address them effectively. The chapter also provides detailed explanations of the proposed algorithms, elucidating their underlying mechanisms and decision-making strategies.

To establish the theoretical foundations of the proposed algorithms, Chapter 4 presents rigorous mathematical proofs. It establishes regret bounds and optimality properties, providing theoretical guarantees for the performance of the proposed algorithms. The chapter employs mathematical reasoning and analysis to demonstrate the superior performance of the developed solutions in comparison to existing approaches.

Chapter 5 moves from theory to practice, showcasing experimental results obtained from simulations and real-world scenarios. These experiments provide empirical evidence to support the claims made regarding the performance and effectiveness of the proposed algorithms. The chapter includes comprehensive analyses, statistical evaluations, and insightful interpretations of the experimental findings, further solidifying the credibility and robustness of the proposed solutions.

Finally, Chapter 6 brings the thesis to a close by summarizing the key findings, contributions, and implications of the research. It reflects on the journey undertaken throughout the thesis, highlighting the advancements made in the field of multi-armed bandits and the significance of the proposed algorithms for solving the addressed problems.

Chapter 2

Background

In this chapter, we start by introducing the concept of Multi-Armed Bandits (MABs). We explain what MABs are and discuss some basic ideas about them. We also talk about different ways MABs can be set up and different types of MABs. Next, we give a brief overview of the main problem our thesis will focus on. We also look at a few studies that deal with more complicated types of MABs. In the end, we discuss why the current methods are not enough to solve our problem and why we need to come up with new methods.

2.1 Multi-Armed Bandits

We start this section by introducing the basic notations used in MAB in Section 2.1.1. Then, we discuss the basic MAB setting in Section 2.1.2. Following that we define regret for this setting in Section 2.1.3. In Section 2.1.4, we discuss the UCB1 algorithm which is commonly used to solve MAB problems.

2.1.1 Notation

An MAB consists of the set of arms $\mathcal{N} = \{1, 2, \ldots, n\}$. Each arm $i \in \mathcal{N}$ has a fixed, but unknown, reward distribution associated with it, with an unknown mean value of $\mu_i \in [0, 1]$. Over the course of T rounds, we pull an arm in every round.

2.1.2 Setting

In the *vanilla MAB*, arms are *stochastic* in nature, and pulling arm i generates a reward X_i, which is a *Bernoulli random variable* with parameter μ_i, having support $D_i \in \{0, 1\}$. The goal is to generate high rewards over T rounds and pull the arm with the highest mean value. Due to the stochasticity of arms, an arm with the highest expected mean in the earlier rounds may not be the best, and it is sometimes a good idea to pull other good arms for better exploration. However, pulling the other arms too often will yield a lower expected reward. This leads to the *Exploration-Exploitation Dilemma*.

Let us explain the Exploration Exploitation Dilemma using a simple example. Take $T = 1000$, $\mathcal{N} = [1, 2], \mu_1 = 0.55$ and $\mu_2 = 0.45$. Let us say that we pulled arm 1 five times and the observed rewards were $1, 0, 0, 1, 0$, and we pulled arm 2 five times and the observed rewards were $1, 0, 1, 1, 0$. Now, the estimated mean of arm 1 is 0.4, and the estimated mean of arm 1 is 0.6. At this point, if we decide to pull arm 2 for the remaining 990 rounds, it will result in a much lower reward than pulling arm 1. However, if the observed rewards were the other way around, and the estimate for arm 1 was 0.6 and for arm 2 was 0.4, pulling arm 1 for the remaining rounds would have been optimal. We need to *exploit the information only after we have explored enough*.

2.1.3 Regret

Consider a simple algorithm $\mathcal{A}_1$ where we pull the arms in a *round-robin* manner, i.e.; arms are numbered from 1 to n and in round t, we pull the arm numbered $(1 + t \mod i)$. Consider another algorithm $\mathcal{A}_2$ where we do a round-robin for the first $T/3$ rounds and get an estimate of the mean reward of each arm and play the best-estimated arm for the remaining rounds. How do we compare the performances of $\mathcal{A}_1$ and $\mathcal{A}_2$ to each other? How do we compare their performances with any arbitrary algorithm?

Definition 2.1 (Regret). *The regret of an algorithm $\mathcal{A}$ is a measure of how well it performs compared to an optimal algorithm $\mathcal{A}^*$. Formally speaking, the regret of algorithm $\mathcal{A}$, denoted by $Reg(\mathcal{A})$ is given by*

$$Reg(\mathcal{A}) = \sum_{t=1}^{T} (\mu_{i_t^*} - X_{i_t}) \tag{2.1}$$

Here, i_t is the arm pulled by algorithm $\mathcal{A}$ in round t, and i_t^* is the arm pulled by the optimal algorithm $\mathcal{A}^*$ in the same round. X_{i_t} denotes the reward obtained on pulling arm i_t in round t by algorithm $\mathcal{A}$.

Since the arms are stochastic, we talk about regret in expectation.

Definition 2.2 (Expected Regret). *The expected regret of algorithm $\mathcal{A}$ is given by*

$$\mathbb{E}[Reg(\mathcal{A})] = \sum_{t=1}^{T}(\mu_{i_t^*} - \mu_{i_t}) \tag{2.2}$$

Naturally, the lower the expected regret of an algorithm, the better it is expected to perform. Comparing an algorithm $\mathcal{A}_1$ with the optimal algorithm $\mathcal{A}^*$ gives it an absolute value in the form of expected regret. Hence, comparing two or more algorithms becomes easier simply by looking at their expected regret.

2.1.4 UCB1

One of the most popular algorithms that solve the MAB problem is the UCB1 algorithm (Auer *et al.* [2]). *The basic idea of the algorithm is to pull the arm with the highest upper confidence bound (UCB) estimate.* Pulling an arm gives us a tighter UCB estimate for the arm. However, if we have pulled an arm for very few rounds, the confidence in its value decreases, and eventually, the UCB value for that arm becomes high enough for it to get pulled. This technique solves the exploration-exploitation dilemma quite well.

Let $N_{i_t}(t)$ be the number of times arm i_t was pulled in the first t rounds. The algorithm is as follows:

Algorithm 1: UCB1

```
Parameter $\alpha > 0$
for $t$ in 1 to $n$ do
    Select arm $i_t = t$.
    Observe reward $X_{i_t}$.
    $\hat{\mu}_{i_t} = X_{i_t}$
end for
for $t$ in $n + 1$ to $T$ do
    Select arm $i_t \in \underset{i \in \mathcal{N}}{\operatorname{argmax}} \left( \hat{\mu}_{i_t} + \sqrt{\frac{\alpha \ln T}{N_{i_t}(t)}} \right)$
    ▷ $\hat{\mu}_{i_t}$ is the exploitation term and $\sqrt{\frac{\alpha \ln T}{N_{i_t}(t)}}$ is the exploration term.
    Observe reward $X_{i_t}$.
    Update UCB estimate for arm $i_t$, given by $\hat{\mu}_{i_t}$.
end for
```

Theorem 2.1. *UCB1 achieves a regret of $O(\log T)$, where T is the number of rounds played.*

A lot of work has been done to come up with algorithms that perform well in such a scenario. Lai & Robbins [21] initiated the work on stochastic MABs and showed that it is possible to achieve an asymptotic regret (defined in Section 2.1.3) of $O(\log T)$. 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 the UCB1 algorithm.

2.2 Variations to the MAB problem

The setting described in Section 2.1.2 is often too simple for most practical applications. This section will consider a few variations of this vanilla MAB. Section 2.2.1 talks about the possible reward settings. In Section 2.2.2, we talk about Combinatorial MABs; in Section 2.2.3, we talk about Budgeted MABs; and in Section 2.2.4, we talk about contextual MABs. Section 2.2.5 talks about the different types of feedback we can get after every round.

2.2.1 Reward Settings

Until now, we have been assuming a stochastic reward setting for the MAB problem. Let us talk about the different types of reward settings in more detail.

2.2.1.1 Stochastic

In the stochastic reward setting, *each arm has a fixed unknown reward distribution, and when pulled, it generates a reward sampled from the said distribution.*

Consider this example of drug testing. Let us say that scientists have come up with n vaccines for a particular variant of coronavirus. Each vaccine can be considered an arm of the bandit, where each vaccine has fixed unknown effectiveness in fighting the virus. Since the vaccine may perform differently on different individuals, a stochastic reward is observed every time a vaccine is given.

2.2.1.2 Deterministic

In the deterministic reward setting, *each arm has a fixed unknown reward, and when pulled, it generates said reward.* This reward may change every round (that also applies to the stochastic reward setting); however, it remains deterministic in that any arm will always yield the same reward no matter how many times it is pulled as long as the reward does not change.

Consider this example of charging your mobile phone from 0 to 100. If you have n different chargers, they will have a fixed, unknown, pre-determined time to charge a said phone from 0 to 100 under certain conditions (say, the phone is switched off to minimize leakage). Each charger can hence be thought of as an arm of the bandit. The time taken (which can be used to measure the reward) may, however, change as the charger gets worse in quality over the years, but on any given day, a charger will always take the same amount of time to charge the phone no matter how many times you repeat it.

2.2.1.3 Adversarial

In the adversarial reward setting, *an adversary decides on the rewards for each arm after seeing our policy for a round. The adversary wants to give us as low of a reward as possible.*

Consider the game of rock-paper-scissors. We have three arms in the MAB, each corresponding to an option. The adversary gets to decide what to play after they observe what we have played. We often use randomization to reduce the adversary's power in such cases. The adversary may observe our policy to play rock, paper, or scissors randomly, but it will not be able to give us the lowest reward every round.

Henceforth, we will assume a stochastic reward setting unless specified otherwise.

2.2.2 Combinatorial MAB (CMAB)

In CMAB (Chen *et al.* [8]), in each round t, we *select a subset of arms (s_t)* instead of always selecting just one arm (i_t). Before we discuss it any further, we need to introduce two important terms.

Definition 2.3 (Superarm). *The subset of arms selected in a round is called a superarm.*

We often treat a superarm s as a single entity, in the sense that we often talk about the reward of a superarm instead of the individual rewards of every arm in the superarm.

Definition 2.4 (Oracle). *An oracle takes the reward vector $\mu_1, \mu_2, \ldots, \mu_n$ and returns an optimal superarm.*

2.2.2.1 Setting

A CMAB consists of n arms. Each arm i has a fixed but unknown reward distribution associated with it, with an unknown mean value of μ_i. Over the course of T rounds, we select a superarm in every round. There is a (possibly) unknown reward function f, which generates a reward given a superarm. In the vanilla MAB, we tend to converge on the optimal arm i^*. Here, we tend to converge on the optimal superarm, denoted by s^*, and pulling any other superarm adds to our expected regret.

For CMAB, we define the expected regret of algorithm $\mathcal{A}$ as follows:

$$\mathbb{E}[Reg(\mathcal{A})] = \sum_{t=1}^{T} (\mu_{s^*} - \mu_{s_t}) \tag{2.3}$$

2.2.2.2 α-approximation Regret

Sometimes, *even if we know the mean value of every arm beforehand, it is still computationally hard to find the optimal superarm.* In general, it is an NP-hard problem. Therefore, instead of trying to compute the optimal superarm, it is sometimes a good idea to use a greedy algorithm to find a superarm that performs reasonably well.

Definition 2.5 (**α-Approximation Oracle**)**.** *An oracle that takes a reward vector $\mu_1, \mu_2, \ldots, \mu_n$ and returns a superarm that generates a reward at least α times that generated by the optimal superarm is called an α-approximation oracle.*

If that is the case, regret is defined with respect to the reward generated by the superarm returned by the oracle instead of the optimal superarm. This gives us an *α-approximate regret*.

Chen *et al.* [8] proposed the algorithm CUCB (Algorithm 2) with computation oracle to solve CMAB problems.

```
Algorithm 2: CUCB with computation oracle
For each arm $i$, maintain: (1) variable $N_i(t)$ as the total number of times arm $i$ has played so
far till end of round $t$ (2) variable $\bar{\mu}_i$ as the mean of all outcomes $X_i$'s of arm $i$ observed so
far.
For each arm $i$, play an arbitrary super arm $s \in \mathcal{S}$ such that $i \in s$ and update variables $N_i(t)$
and $\bar{\mu}_i$.
Set $t = n$.
▷ Since $n$ rounds have already been played
while TRUE do
    $t = t + 1$
    For each arm $i$, set $\hat{\mu}_i = \bar{\mu}_i + \sqrt{\frac{3 \ln t}{2N_i(t)}}$.
    ▷ $\sqrt{\frac{3 \ln t}{2N_i(t)}}$ is the exploration term when the mean value is $\bar{\mu}_i$
    $s = Oracle(\bar{\mu}_1, \bar{\mu}_2, \ldots, \bar{\mu}_n)$.
    ▷ $s$ is the optimal superarm, according to the oracle, if the mean estimate of each
    arm were the respective true value.
    Play $s$ and update all $N_i(t)$'s and $\bar{\mu}_i$'s.
end while
```

2.2.3 Budgeted Multi-Armed Bandits (BMAB)

Often, pulling an arm incurs additional costs. In a BMAB (Madani *et al.* [24], Tran-Thanh *et al.* [32], Tran-Thanh *et al.* [34]), we *have a spending budget, and each arm has a cost it incurs on being pulled.* In the simpler case where we know the actual value of every arm, it reduces to a knapsack problem. Hence, it is only natural to talk about the knapsack problem before proceeding to BMAB.

2.2.3.1 The Knapsack problem

Problem Definition

The knapsack problem (*unbounded knapsack*, to be specific) states that we have a set of items, each having a weight and value, and the objective is to select items such that their total value is as high as possible, but their total weight should not be more than a certain pre-determined limit. Each item can be selected as many (non-negative integer) times as required.

LP solution

Mathematically speaking, we have n items, where the i^{th} item has weight w_i and value v_i. The maximum weight limit is W. The goal is to determine the number of times each item is selected. Let t_i be the number of times we select item i. This can be found by solving the following LP:

$$\begin{aligned} \max Z \quad &= \sum_{i=1}^{n} x_i v_i \\ \text{subject to} \quad & \sum_{i=1}^{n} x_i w_i \leq W \\ & x_i \in \mathbb{Z}^+ \end{aligned} \tag{2.4}$$

Greedy algorithm

There is no known polynomial time solution for the knapsack problem. However, there is a greedy algorithm (Algorithm 3) that gives a set of items whose total value is at least $\frac{V^*}{2}$, where V^* is the value attained by an optimal algorithm.

Algorithm 3: Greedy algorithm for Unbounded Knapsack

```
Parameters: Weights $w_i$ $\forall\, i \in \{1, 2, \ldots, n\}$, value $v_i$ $\forall\, i \in \{1, 2, \ldots, n\}$, weight limit $W$
Sort based on non-increasing $\frac{v_i}{w_i}$ and renumber them as $1', 2', \ldots, n'$.
While not exceeding budget: pick as many items as possible, going in the order of
    $1', 2' \ldots n'$.
```

We can easily do this in $O(n \log n)$.

2.2.3.2 Setting

A BMAB consists of n arms. Each arm i has a fixed but unknown reward distribution associated with it, with an unknown mean value of μ_i. Each arm i also has a cost c_i that it incurs when pulled. Over the course of T rounds, we select an arm in every round, subject to the constraint that the cost of the selected arm does not exceed our budget B.

2.2.3.3 Regret

In the non-budgeted version of single pull MAB, we tend to converge on the optimal arm i^*. Here, *we do not necessarily have an optimal arm but an optimal sequence of pulls.*

For BMAB, we define the expected regret of algorithm $\mathcal{A}$ as follows:

$$\mathbb{E}[Reg(\mathcal{A})] = \sum_{t=1}^{T} (\mu_{i_t^*} - \mu_{i_t}) \tag{2.5}$$

The sequence $i_1^*, i_2^*, \ldots, i_T^*$ gives us the optimal sequence of pulls.

2.2.3.4 KUBE

One of the first approaches towards solving a BMAB problem gave the KUBE (knapsack–based upper confidence bound exploration and exploitation) algorithm (Tran-Thanh *et al.* [34]) (Algorithm 4). The key idea is to *solve a knapsack problem with the UCB values at every round and randomly pull an arm with a probability directly proportional to its count as per the knapsack solution.* The knapsack problem with UCB values is given as follows:

$$\begin{aligned} \max \quad & = \sum_{i=1}^{n} m_{i,t} \left(\hat{\mu}_{i,N_i(t)} + \sqrt{\frac{2 \ln t}{N_i(t)}} \right) \\ \text{subject to} \quad & \sum_{i=1}^{n} m_{i,t} c_i \leq B_t \\ & m_{i,t} \in \mathbb{Z}^+ \; \forall \; i, t \end{aligned} \tag{2.6}$$

Here, $\hat{\mu}_{i,N_i(t)}$ is the current estimate of arm i's expected reward (calculated as the average reward received so far from pulling arm i), $N_i(t)$ is the number of pulls of arm i until round t, and $\sqrt{\frac{2 \ln t}{N_i(t)}}$ is the size of the upper confidence interval. The goal, then, is to find integers $m_{i,t}$ such that Equation 2.6 is maximized, with respect to the residual budget B_t.

Algorithm 4: KUBE

```
Initialisation: Round $t = 1$, residual budget $B_t = B$
while pulling is feasible do
    if $B_t < \min_i c_i$ then
        STOP! (Pulling is not feasible)
    end if
    if $t \leq n$ then
        Initial phase: play arm $i_t = t$.
    else
        Take $B_t$ as the residual budget and calculate $m^*_{i,t}$ $\forall\, i$ using Equation 2.6.
        Randomly pull arm $i_t$ with probability $P(i_t = i) = \frac{m^*_{i,t}}{\sum_{j=1}^{n} m^*_{j,t}}$.
        ▷ Explore with probability $P(i_t = i)$ and exploit otherwise.
    end if
    Update the estimated upper bound of arm $i_t$.
    $B_{t+1} = B_t - c_{i_t}$
    $t = t + 1$
end while
```

Theorem 2.2. *KUBE has a $O(n \log n)$ computational cost per time step and achieves a regret of $O(\log B)$.*

2.2.3.5 Multiple Budget Constraints

Badanidiyuru *et al.* [4] proposed an interesting problem setting was proposed in the paper Bandits with Knapsacks. They considered a setting where *pulling an arm may consume multiple resources, and each resource has a budget limitation.* They consider a single pull setting (non-CMAB), and arms are pulled as long as the budget permits.

One of their approaches to solving this problem leads to the algorithm PrimalDualBwK (Algorithm 5). The basic idea is very natural and intuitive: *greedily select arms with the highest estimated "bang-per-buck", i.e., reward per unit of resource consumption.* One of the main difficulties with this idea was that there is no such thing as a known “unit of resource consumption”: there are d different resources, and it is unclear how to trade off the consumption of one resource versus another. An optimal solution η^* can be interpreted as a vector of unit costs for resources, such that for every arm, the expected reward is less than or equal to the expected cost of resources consumed. Then the bang-per-buck ratio for a given arm i, with resource consumption vector C_i, can be defined as $\frac{\mu_i}{\eta^* \cdot C_i}$, where the denominator represents the expected cost of pulling this arm. To estimate the bang-per-buck ratios, the algorithm learns optimal vector η^* in tandem with learning the latent structure μ.

The algorithm uses the *multiplicative weights update method* to learn the optimal vector. This method raises the cost of a resource exponentially as it is consumed, ensuring that heavily demanded resources become costly and thereby promoting balanced resource consumption.

Algorithm 5: PrimalDualBwK

```
Initialisation:
In the first $m$ rounds, pull each arm once
$v_1 = \mathbf{1} \in [0,1]^d$
$v_t$ is the round $t$ estimate of the optimal solution $\eta^*$.
    ▷ Interpret $v_t(j)$ as an estimate of the (fictional) unit cost of resource $j$, for each $j$
Set $\epsilon = \sqrt{\frac{\ln d}{B}}$
for rounds $t = n+1, n+2, \dots,$ until resource budget exhausted do
    for each arm $i \in \mathcal{N}$ do
        Compute UCB estimate for the expected reward, $\hat{\mu}_i(t) \in [0,1]$
        Expected cost for one pull of arm $i$ is estimated by $EstCost_i = C_i v_t$
    end for
    Pull arm $i = i_t \in \mathcal{N}$ that maximizes $\frac{\hat{\mu}_i(t)}{EstCost_i}$, the optimistic bang-per-buck ratio.
    Update estimated unit cost for each resource $j$:
        $v_{t+1}(j) = v_t(j)(1+\epsilon)^{C_i(j)}$
end for
```

Theorem 2.3. *PrimalDualBwK achieves a regret of* $O\left(\sqrt{\log T}\frac{OPT}{\sqrt{B}} + \log^2 T\right)$.

2.2.4 Contextual Multi-Armed Bandits (ConMAB)

A ConMAB (Li *et al.* [22], Lu *et al.* [23],) as the name suggests, is *a MAB with additional context*. Usually, we have some additional information about the arms that can be useful if used properly.

Take the example of drug testing. While we could test drugs on patients without any prior assumptions about them and treat them uniformly, we should factor in their medical history for better-directed treatment. Grouping patients based on the similarity in their medical history is better for more efficient learning of how a drug interacts with a type of patient.

2.2.4.1 Setting

In ConMAB, in round t, we *receive a d-dimensional context vector* $x_t \in \mathcal{X}$, where $\mathcal{X}$ is the context set. Here, $x_t \in [0,1]^d$. Each arm i also has a d-dimensional context vector $x^i \in [0,1]$ associated with it.

In the case of ConMAB, we do not have a single arm that we can call optimal. However, we do have an optimal policy.

Definition 2.6 (Policy). *A policy is a mapping $g : \mathcal{X} \to \mathcal{N}$ that tells us which arm to play on seeing context $x \in \mathcal{X}$.*

2.2.4.2 Regret

We define expected regret for algorithm $\mathcal{A}$ in the case of ConMAB as follows:

$$\mathbb{E}[Reg(\mathcal{A})] = \max_{g:\mathcal{X}\to\mathcal{N}} \mathbb{E}[\sum_{t=1}^{T} X_{g(x_t)_t} - \sum_{t=1}^{T} X_{i_t}] \tag{2.7}$$

2.2.4.3 S-Exp3

The simplest approach to solve a ConMAB problem is to *run a separate instance of the $Exp3$ (Exponential weights algorithm for exploration and exploitation) algorithm on each context.* This is known as the $S - Exp3$ algorithm.

Exp3 algorithm is given below:

```
Algorithm 6: S-Exp3
Parameters: Exploration probability $\gamma \in [0, 1]$
Initialisation: Weights $w_i(1) = 1 \; \forall \; i \in \mathcal{N}$
for $t = 1, 2, \ldots, T$ do
    Set probability of playing arm $i$ as
    $p_i(t) = (1 - \gamma)\frac{w_i(t)}{\sum_{i \in \mathcal{N}} w_j(j)} + \frac{\gamma}{n} \; \forall \; i \in \mathcal{N}$.
    Play $i_t$ randomly according to the probabilities $p_1(t), p_2(t), \ldots, p_n(t)$.
    Receive reward $X_{i_t}(t) \in \{0, 1\}$.
    for $j \in \mathcal{N}$ do
        Set:
        $\hat{X}_{j_t} = \begin{cases} \frac{X_j(t)}{p_j(t)}, \text{ if } j = i_t \\ 0, \text{ otherwise} \end{cases}$
        ▷ Update only for the arm that was played and observed.
        $w_j(t+1) = w_j(t) \exp(\gamma \frac{\hat{X}_{jt}}{n})$
    end for
end for
```

Theorem 2.4. *The $S - Exp3$ algorithm achieves a regret of $O(\sqrt{T|S|n \log n})$.*

2.2.5 Types of feedback

In a MAB, the reward we observe may differ from setting to setting. There are three main types of feedback.

2.2.5.1 Full feedback

We are said to receive full feedback when you can *observe the effect of every arm,* regardless of the arm (or superarm) you pull. An example is that even if a student takes 5 out of 20 courses, he can still get to know how well all the other courses were taught at the end of the semester based on a peer review.

2.2.5.2 Bandit feedback

We are said to receive bandit feedback when you can *only observe the effect of the played arm* (or superarm) as a whole. When taking multiple medicines for a disease, we generally see the effect of the medicines as a whole. Thus, we only receive bandit feedback.

2.2.5.3 Semi-bandit feedback

We are said to receive semi-bandit feedback when you can *observe the effect of the superarm as a whole and certain effects of the arms in the superarm.* As a manager, you will likely be able to observe how the selected workers perform on the task and also see the results of the group as a whole.

Henceforth, we will assume semi-bandit feedback unless specified otherwise.

2.2.6 Important Notations

For ease of reference, Table 2.1 lists some important common notations used in this thesis.

Symbol	**Description**
n	Number of arms
$[n] = \{1, 2, ..., n\}$	Set of arms
T	Number of rounds
$\mathcal{S}$	Set of superarms
s	Subset of arms
s_t	Subset selected in round t
i_t	Single arm selected in round t
X_{s_t}	Reward observed for pulling subset s_t in round t
X_{i_t}	Reward observed for pulling arm i_t in round t
i_t^*	Optimal arm to play in round t
i_t^A	Arm selected by the algorithm A to play in round t
$Reg_t(A)$	Regret for Algorithm A for round t
$Reg(A)$	Regret for Algorithm A over T rounds
μ_i	Mean reward for arm i
μ^*	Mean reward for optimal arm
$\hat{\mu}_i$	UCB estimate of arm i
$\bar{\mu}_i$	Emperical mean of arm i
$N_i(t)$	Number of times arm i was pulled in the first t rounds
$\mathcal{X}$	Context set
x_t	Context vector obtained in round t
x^i	Context associated with arm i
B	Budget constraint
c_{i_t}	Cost of pulling arm i_t in round t
$c_{i_t}^j$	Cost associated with the j^{th} resource for pulling arm i_t in round t

Table 2.1: Commonly used Notations

2.3 The BCMAB Problem

The main problem setting addressed in this thesis is that of a multi-armed bandit with stochastic reward setting (Section 2.2.1.1) having combinatorial framework (Section 2.2.2) with budget constraints (Section 2.2.3) and with semi-bandit feedback (Section 2.2.5.3). We call this the BCMAB problem.

In the next section, we will review related papers and discuss their approaches, and see how there is a need for new techniques to solve the BCMAB problem.

2.4 Attempting to solve BCMAB: A Look at Related Problems

In this section, we will review a few selected papers involving complex variants of MABs to see how the subtopics from Section 2.2 interact. We will also see what their approaches lack to solve the BCMAB problem as described in Section 2.3.

2.4.1 Combinatorial Multi-Armed Bandit Based Unknown Worker Recruitment in Heterogeneous Crowdsensing

2.4.1.1 Problem Addressed

In this paper Gao *et al.* [12], the focus is on the unknown worker recruitment problem in mobile crowdsensing, where workers' sensing qualities are unknown a priori. They consider the scenario of recruiting workers to complete some continuous sensing tasks.

The whole process is divided into multiple rounds. In each round, every task may be covered by more than one recruited worker, but its completion quality only depends on these workers' maximum sensing quality. Each recruited worker will incur a cost, and each task has attached a weight to indicate its importance. The objective is to determine a recruiting strategy to maximize the total weighted completion quality under a limited budget. They extend the problem to the case where the workers' costs are also unknown.

2.4.1.2 Proposed Solution

- They turn the unknown worker recruitment problem for heterogeneous Mobile CrowdSensing systems into a K-arm CMAB problem

- They propose an extended UCB-based arm-pulling strategy to solve the CMAB problem and design the corresponding unknown worker recruitment online algorithm (UWR)
- They also study an extended case where both the sensing qualities and the costs of workers are unknown and devise another algorithm (EUWR)
- They conduct extensive simulations on real-world traces to evaluate the significant performance of the algorithms

Proposed Algorithms - UWR and EUWR

UWR

The platform selects the first options of all workers with the minimum cost to initialize several parameters, such as $N_i(t)$ and $\bar{\mu}_i(t)$ (average learned quality value for i until the t^{th} round). Then, the platform selects K workers according to the UCB-based qualities and the proposed selection criterion. To meet the constraint that at most one option of a worker can be selected in a round, let $\mathcal{P}^{t'}$ denote the set of not satisfying options. Then, the option with the largest ratio of the marginal UCB-based quality function value and the cost is selected from the set $\mathcal{P} \setminus \mathcal{P}^{t'}$. The platform decides whether to terminate the algorithm based on the remaining budget. If the remaining budget is enough, the recruited workers perform the corresponding tasks and send the sensing results to the platform. The platform updates the worker profiles. The remaining budget and total achieved weighted quality are accordingly updated.

Theorem 2.5. *UWR has a regret bound of $O(nLK^3 \ln B)$, where L is the number of options of each of the K workers.*

EUWR

EUWR is very similar to UWR. The key difference between UWR and EUWR is that in EUWR, the selection criterion takes the obtained quality and cost values (as a ratio of quality to cost) into consideration simultaneously (whereas in UWR, only the quality was under consideration).

Theorem 2.6. *EUWR has a regret bound of $O(nLK^3 \ln (nTB))$.*

Why does the proposed solution fail for BCMAB?

They assume a fixed superarm size K of CMAB, whereas BCMAB handles a more general case with no superarm size limitations.

2.4.2 Efficient Budget Allocation with Accuracy Guarantees for Crowdsourcing Classification Tasks

2.4.2.1 Problem Addressed

In this paper Tran-Thanh *et al.* [35], the authors address the problem of budget allocation for redundantly crowdsourcing a set of classification tasks where a key challenge is to find a trade-off between the total cost and the accuracy of estimation.

2.4.2.2 Proposed Solution

- They introduce the problem of budget allocation for crowdsourcing classification tasks, in which the goal is to minimize the error of the estimated answers for a finite number of tasks with respect to a budget limit.

- They develop CrowdBudget, an algorithm that, combined with a fusion method, provably achieves an efficient bound on the estimation error, which significantly advances the best-known results.

- They compare the performance of CrowdBudget with existing algorithms through extensive numerical evaluations of real-world data taken from a prominent crowdsourcing system.

The CrowdBudget Algorithm

Let N_t denote the number of users the agent aims to assign to task t. It first pre-sets to $N_t = \left\lfloor \frac{B}{c_t^2 \sum_{i=1}^{n} \frac{1}{c_i}} \right\rfloor$. The agent also maintains B^r that denotes the residual budget, initially set to be B. After each pre–set of N_t, B^r is decreased by $N_t c_t$. where c_t is the cost that user t has to be paid. Next, if $B^r > 0$, the agent sequentially increases the number of allocated users for each task t by 1 until the original budget is exceeded. This phase guarantees that the budget is fully used. Following this, the agent redundantly submits the tasks to the system, and once it receives the responses from the users, it uses a Majority-Voting–efficient fusion method to estimate the answers to each task.

Theorem 2.7. *CrowdBudget achieves at most* $\max\{0, \frac{T}{2} - O(\sqrt{B})\}$ *estimation error with high probability.*

Why does the proposed solution fail for BCMAB?

They consider a setting where the workers are homogeneous, and a superarm gets a reward in $\{0, 1\}$. We consider a more general setting where the workers (arms) may differ, and a superarm gets a reward in $[0, 1]$.

2.4.3 Budget-Constrained Multi-Armed Bandits with Multiple Plays

2.4.3.1 Problem Addressed

The paper Zhou & Tomlin [39] studies the multi-armed bandit problem with multiple plays and a budget constraint for both the stochastic and the adversarial setting. At each round, exactly K out of n possible arms must be played. In addition to observing the individual rewards for each arm played[1], the player also learns a vector of costs which has to be covered with an a priori defined budget B. The objective is to derive algorithms that minimize regret for these settings, given that the reward and cost distribution of the arms is unknown.

2.4.3.2 Proposed Solution

- The authors provide an algorithm (UCB-MB) for the stochastic setting and prove its regret bound.
- They also provide an algorithm (Exp3.M.B) for the adversarial setting and prove its regret bound.
- They adjust Exp3.M.B to handle budget constraints and thereby provide algorithm Exp3.1.M.B, and prove its regret bound.

Proposed Algorithms - UCB-MB, Exp3.M.B and Exp3.1.M.B

UCB-MB

Given a bandit with n distinct arms, each arm indexed by $i \in \mathcal{N}$ is associated with an unknown reward and cost distribution with unknown mean $0 < \mu_i \leq 1$ and $0 < c_{min} \leq c_i \leq 1$, respectively. Realizations of costs $c_{i_t} \in [c_{min}, 1]$ and rewards $\mu_{i_t} \in [0, 1]$ are independently and identically distributed. At each round t, the decision maker plays exactly K arms (the K arms associated with the K largest confidence bounds) and subsequently observe the individual costs and rewards only for the played arms, which corresponds to the semi-bandit setting. Before the game starts, the player is given a budget $0 < B \in \mathbb{R}_+$

[1] The paper does not address how the individual rewards combine. However, the setting is semi-bandit feedback

to pay for the materialized costs $\{c_{i_t} \mid i \in a_t\}$, where a_t denotes the indices of the K arms played at time t. The game terminates as soon as the sum of costs at round t, namely $\sum_{j \in a_t} c_{j_t}$ exceeds the remaining budget.

Theorem 2.8. *UCB-MB achieves a regret of $O(nK^4 \log B)$.*

Exp3.M.B

The adversarial case makes no assumptions on the reward and cost distributions whatsoever. They consider the extension of the classic setting, where the decision-maker has to play exactly $1 \le K \le n$ arms. For each arm i played at round t, the player observes the reward $X_{i_t} \in [0, 1]$ and, additionally the cost $0 < c_{min} < c_{i_t} < 1$. The player is given a budget $B > 0$ to pay for the costs incurred, and the algorithm terminates after $\tau_{\mathcal{A}}(B)$ rounds when the sum of materialized costs in round $\tau_{\mathcal{A}}(B)$ exceeds the remaining budget.

Algorithm Exp3.M.B maintains a set of time-varying weights $\{w_i(t)\}_{i=1}^n$ for all arms, from which the probabilities for each arm being played at time t are calculated. The probabilities $\{p_i(t)\}_{i=1}^n$ sum to K (because exactly K arms need to be played), which requires the weights to be capped at a value $v_t > 0$ such that the probabilities $\{p_i(t)\}_{i=1}^n$ are kept in the range [0, 1]. In each round, the player draws a set of distinct arms at of cardinality $|a_t| = K$, where each arm has probability $p_i(t)$ of being included in a_t. At the end of each round, the observed rewards and costs for the played arms are turned into estimates $\hat{\mu}_{i_t}$ and $\hat{c}_{i_t}$. Arms with $w_i(t) < v_t$ are updated according to $(\hat{\mu}_{i_t} - \hat{c}_{i_t})$, which assigns larger weights as $\hat{\mu}_{i_t}$ increases and $\hat{c}_{i_t}$ decreases. Regret achieved by Exp3.M.B depends on the upper bound for OPT, where OPT is the optimal cumulative reward under the optimal set.

Theorem 2.9. *Exp3.M.B achieves a regret of $O(\sqrt{Bn \log \frac{n}{K}})$.*

Exp3.1.M.B

If no upper bound on OPT exists, Algorithm Exp3.M.B. needs to be modified. The weights have to be updated differently. As in Algorithm Exp3.1 in Auer *et al.* [2], the authors use an adaptation of Algorithm Exp3.M.B, which they call Exp3.1.M.B. Algorithm Exp3.1.M.B utilizes Algorithm Exp3.M.B as a subroutine in each epoch until termination.

Theorem 2.10. *Exp3.1.M.B achieves a regret of $O(\sqrt{nKT \log \frac{n}{K}} + \frac{n-K}{n-1} \log(\frac{nT}{\delta}) + K^2 \sqrt{nT \frac{n-K}{n-1} \log(\frac{nT}{\delta})})$ with a probability of $1 - \delta$.*

Why does the proposed solution fail for BCMAB?

They assume a fixed superarm size K of CMAB, whereas BCMAB handles a more general case with no superarm size limitations.

2.4.4 Combinatorial Semi-Bandits with Knapsacks

2.4.4.1 Problem Addressed

The paper Sankararaman & Slivkins [29] unifies two prominent lines of work on multi-armed bandits: bandits with knapsacks (BwK) Badanidiyuru *et al.* [4] and combinatorial semi-bandits Gyorgy *et al.* [15]. It presents a combinatorial MAB setting with budget constraints.

2.4.4.2 Proposed Solution

The authors design an algorithm SemiBwK-RRS (Algorithm 7), achieving regret rates comparable with those for BwK and combinatorial semi-bandits.

The SemiBwK-RRS Algorithm

The algorithm builds on an arbitrary random rounding scheme (RRS) for $\mathcal{S}$, where $\mathcal{S}$ is the set of all possible superarms. It is parameterized by this RRS and a number $\epsilon > 0$. In each round t, it recomputes the upper/lower confidence bounds and solves the following linear program:

$$\begin{aligned} \max \quad & \mu_t^+ x \\ \text{subject to} \quad & C_t^- x \le \frac{B(1-\epsilon)}{T}, \ j \in [d] \end{aligned} \tag{2.8}$$

This linear program defines a linear relaxation of the original problem, which is *optimistic* in the sense that it uses upper confidence bounds for rewards and lower confidence bounds for consumption. The linear relaxation is also *conservative* in the sense that it rescales the budget by $1 - \epsilon$. Essentially, this ensures that the algorithm does not run out of budget with high probability. Parameter ϵ will be fixed throughout. For ease of notation, they denote $B_\epsilon := (1 - \epsilon)B$. The LP solution x can be seen as a probability vector over the atoms. Finally, the algorithm uses the RRS to convert the LP solution into a feasible superarm.

```
Algorithm 7: SemiBwK-RRS
Input: an RRS for action set $\mathcal{S}$, $\epsilon > 0$
for $t = 1, 2, \ldots, T$ do
    Obtain fractional solution $x_t \in [0,1]^n$ by solving the linear program 2.8
    Obtain a feasible superarm $s_t \in \mathcal{S}$ by invoking the RRS on vector $x_t$
    Semi-bandit Feedback: observe the rewards/consumption for all superarms $s \in S_t$.
end for
```

Theorem 2.11. *Regret* $\leq O(\log(ndT))(\sqrt{n})(\frac{OPT}{\sqrt{B}} + \sqrt{T + OPT})$, *where d is the number of resources.*

Theorem 2.12. *The per-round running time is polynomial is n, and near-linear in n for some important special cases.*

Why does the proposed solution fail for BCMAB?

They equally distribute the budget in each round. We do not put such restrictions on our setting; hence our BCMAB setting is more general than the setting they solve in the paper.

2.5 More Relevant Work

2.5.1 Vanilla MAB

Lai & Robbins [21] 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 the UCB1 algorithm.

2.5.2 Combinatorial MAB

Stochastic combinatorial multi-armed bandits (CMABs) were first introduced by Chen *et al.* [8]. The authors used an (α, β)-approximation oracle that takes the distributions of arms and outputs a subset of arms with probability β generates an α fraction of the optimal expected reward. The concept of (α, β)-approximation oracle has been used in many CMAB related works such as Chen *et al.* [7]. This setting was extended to linear rewards by Wen *et al.* [37]. Recently, the regret of Thompson sampling-based algorithm was derived for CMAB by Wang & Chen [36]. 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.* [18] considers a setting where at each round the subset of arms needs to be selected to satisfy a quality constraint. Other examples of settings that have considered combinatorial bandits include [5, 10, 16, 30].

2.5.3 Budgeted MAB

Budgeted multi-armed bandits (BMABs) were introduced in Tran-Thanh *et al.* [32], and the concept of Knapsack in BMABs was later elaborated on in Tran-Thanh *et al.* [34]. Works like Nhat *et al.* [26], Tran-Thanh *et al.* [35], Zhang *et al.* [38], Tran-Thanh *et al.* [33], Jain *et al.* [18], Reddy *et al.* [28] 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.

2.5.4 Interesting MAB Variants

Lately, interesting new variants have been proposed to solve more complex problems. The solutions to these problems often require thinking from a different perspective as the previous approaches do not work well enough.

Smart grids are being implemented worldwide to reduce electricity consumption costs. Chandlekar *et al.* [6] propose a MAB-based online algorithm to learn the rate of reduction which is then used to design an optimal algorithm for their setting. Reddy *et al.* [28] propose a bounded integer min-knapsack MAB algorithm that optimizes the loss to the distributor company.

Solanki *et al.* [31] address the problem of agents competing with each other in a federated learning environment. They propose a Privacy-preserving Federated Combinatorial Bandit algorithm.

Deva *et al.* [10] explore CMAB under quality constraints. Sharma *et al.* [30] and Abhishek *et al.* [1] model sponsored search as a MAB problem. Ravindranath *et al.* [27] model human experts as arms for a bandit and use it for face recognition. Jain *et al.* [18] develop an assured accuracy bandit for binary labeling tasks in a cost-optimal way. Ghalme *et al.* [14] explore the case when the set of available arms grows over time. Bhat *et al.* [5] propose an algorithm for the case where we learn both the cost and the capacity of the arms.

Manisha & Gujar [25] use a data-driven approach and Ghalme *et al.* [13] use a randomized payment rule for designing a Thompson Sampling based MAB mechanism. These mechanisms satisfy certain game-theoretic properties and have applications in some interesting scenarios. Jain *et al.* [17] develop a deterministic MAB mechanism with logarithmic regret in social welfare that can be applied to a general crowdsourcing setting.

2.5.5 Recent MAB work

Even though CMABs and BMABs have been explored quite intensively, not much literature exists for both of them together. Chen *et al.* [7], Zhou & Tomlin [40], Gao *et al.* [12] provide a setting that assumes a fixed size of super-arm. The closest work to ours is by Sankararaman & Slivkins [29]. They consider a similar setting to BCMAB with an overall budget and no restriction on the size of superarm. However, they distribute the budget uniformly over all rounds and incur 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 a uniform budget over all rounds, and our CBwK-LP-UCB algorithm incurs a regret of $O(\log^2 T)$, which is a substantial improvement.

2.6 A Need for New Approaches

In Section 2.4, we saw that even though a lot of work has been done in BMABs and CMABs with budget constraints, they do not consider the BCMAB setting which does not place any restriction on having a fixed budget per round or fixed size of superarm per round. Assuming such restrictions and using the previously existing approaches for our BCMAB setting can lead to arbitrarily inefficient algorithms, as we show in Lemma 3.1 and Proposition 3.1 in the next chapter. Hence, there is a need for new approaches to solve BCMAB.

Chapter 3

Designing Algorithms

This chapter commences with a detailed elaboration of the additive case as applied to our BCMAB problem. Following this, we delve into the deterministic setting, exploring a dual-pronged approach encompassing both a greedy technique and a Linear Programming (LP) strategy. The latter part of this chapter shifts our attention to the stochastic setting, where we extrapolate our previous approaches and propose relevant algorithms, thereby offering a comprehensive understanding of the problem at hand.

3.1 Preliminaries - BCMAB

Let $\mathcal{N} = \{1, 2, \ldots, n\}$ 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 $\mu = \{\mu_1, \mu_2, \ldots, \mu_n\}$. 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 μ_i being the 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 algorithm 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:

$$\begin{aligned} \max_{\{S_t\}_{t=1}^{T}} \; REW_{ALG} \quad &= \sum_{t=1}^{T} \sum_{i \in S_t} \mu_i \\ \text{subject to} \quad & \sum_{i=1}^{n} N_i(T) c_i \leq B \\ & 0 \leq N_i(T) \leq T \end{aligned} \tag{3.1}$$

Optimization Problem in Equation 3.1 is hard even when the mean rewards μ_is are known. For $T = 1$, the problem reduces to a Knapsack problem. In the unknown reward setting, Sankararaman & Slivkins [29] assumes a fixed budget per round, which approximates Optimization Problem 3.1 as T independent Knapsack problems. However, this may perform arbitrarily badly, as we prove below.

Lemma 3.1. *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 ϵ_1 and ϵ_2. Since the 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 μ_1 close to 1 and μ_2 close to 0), fixing the budget can perform arbitrarily bad. □

Proposition 3.1. *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. 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 $REW_{ALG1} = (\mu_1 + \mu_2)\frac{B}{(c_1+c_2)}$ whereas 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 ϵ can be arbitrarily small, this ratio can be arbitrarily bad. □

Lemma 3.1 and Proposition 3.1 illustrate that fixing the 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 the

Optimization Problem 3.1. Since the problem is hard even for a known reward setting, we first discuss the algorithms that can be used for a known reward setting.

Note: Optimization Problem 3.1 can be modeled 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 the knapsack problem, we propose a greedy approach in the next subsection which select the arms with respect to the bangperbuck ratio.

3.1.1 Deterministic Setting with Known μ - Greedy Approach (CBwK-Greedy)

Since modeling Optimization Problem 3.1 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 a knapsack problem where each arm has T copies and the goal is to select 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 at most 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 a higher bangperbuck ratio the maximum number of times possible before the arms with a 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 8.

Algorithm 8: CBwK-Greedy

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

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

Remark 1: Since CBwK-Greedy is similar to that of the 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.

3.1.2 Deterministic Setting with Known μ - Modelling it as LP (CBwK-LP)

We can also write the optimization problem in Equation 3.1 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 not to 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 the C_i vector will be $n+1$, with the first n components corresponding to n additional resource and the 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 the rest 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:

$$M_{ji} = \begin{cases} c_i \cdot B'/B & \text{if } j = N+1 \\ 1 \cdot B'/T, & \text{if } i = j \\ 0, & \text{otherwise} \end{cases} \tag{3.2}$$

Here, M_{ji} indicates the cost of resource j if we pull arm i. Now, we write this as a relaxed LP as shown in Equation 3.3. We also write its dual in Equation 3.4.

$$\begin{aligned} &\max \sum_{i \in \mathcal{N}} \zeta_i \mu_i, \quad \zeta_i \in \mathbb{R}, \\ &\text{s.t.} \sum_{i \in \mathcal{N}} \zeta_i M_{ji} \leq B' \quad \forall j \in \{1, 2, \ldots, n+1\} \\ &\zeta_i \geq 0, \quad \forall i \in \mathcal{N} \end{aligned} \tag{3.3}$$

The variables ζ_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.

$$\begin{aligned} &\min B' \sum_j \eta_j, \quad \eta_j \in \mathbb{R} \\ &\text{s.t.} \sum_j \eta_j M_{ji} \geq \mu_i, \quad \forall i \in \mathcal{N} \\ &\eta_j \geq 0 \quad \forall j \in \{1, 2, \ldots, n+1\} \end{aligned} \tag{3.4}$$

The dual variables η_j can be interpreted as a unit cost for the corresponding resource j. We refer to the algorithm to solve the 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 the reward from arm pulls is stochastic. We prove this in the following Lemma.

Lemma 3.2. *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 $\eta^* = (\eta_1^*, \dots, \eta_d^*)$ denote an optimal solution to Equation 3.4. Interpret each η_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 $1, \dots, t$ of the optimal dynamic policy, plus the cost of the remaining resource endowment after round t, then the stochastic process $Z_0, Z_1, \dots, Z_T'$ 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. □

3.1.3 Regret

Let the optimal algorithm for Optimization Problem 3.1 be ALG^*. Its reward is given by REW_{ALG^*}. We define regret incurred by an algorithm ALG as

$$REG_{ALG}(\mathcal{N}, T, B) = REW_{ALG^*}(\mathcal{N}, T, B) - REW_{ALG}(\mathcal{N}, T, B) \tag{3.5}$$

3.2 Proposed Approaches - BCMAB

3.2.1 CBwK-Greedy-UCB

Our first algorithm CBwK-Greedy-UCB in unknown reward setting is presented in Algorithm 9. The algorithm basically extends the greedy algorithm presented in section 3.1.1 so as to select a subset of arms at each round t.

Algorithm 9: 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 $B_r = B$.
    ▷ Initialization
 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
    ▷ Treat CBwK-Greedy as the oracle.
 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 3.1.1 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 a 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 not to 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.

3.2.2 CBwK-LP-UCB

The *intuitive idea for the CBwK-LP-UCB algorithm is to select arms with the highest estimated bangperbuck ratio greedily.* 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 10.

Algorithm 10: 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 the 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 η^* after a set of rounds t.

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

Key Differences between CBwK-LP-UCB and PrimalDualBwK

- The *fundamental difference between CBwK-LP-UCB and PrimalDualBwK 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 a combinatorial setting, and why it is not a trivial extension with the 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 not to 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 the 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 suit our combinatorial setting and present the regret analysis for CBwK-LP-UCBand prove a regret bound of the form:

$$OPT_{LP} - REW_{\text{CBwK-LP-UCB}} \leq f(OPT_{LP}) \tag{3.6}$$

where $f(\cdot)$ is a linear function that depends only on parameters (B', n, T'). Regret bound (3.6) implies the claimed regret bounds relative to REW_{ALG^*} because

$$REW_{\text{CBwK-LP-UCB}} \geq OPT_{LP} - f(OPT_{LP}) \geq REW_{ALG^*} - f(REW_{ALG^*}) \tag{3.7}$$

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 3.2. From Equation 3.7 and Equation 3.5,

$$REG_{\text{CBwK-LP-UCB}} = REW_{ALG^*} - REW_{\text{CBwK-LP-UCB}} \leq f(REW_{ALG^*}) \tag{3.8}$$

3.2.3 Differences with SemiBwK-RRS

As discussed in Section 2.4.4, even though Sankararaman & Slivkins [29] 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 3.1 that this can perform arbitrarily badly. We use the budget in each round adaptively, thus leading to better regret bounds.

In the next chapter, we bound our regret and prove Equation 3.8.

Chapter 4

Theoretical Analysis

This chapter explores the regret analysis of our algorithm, named CBwK-LP-UCB. The most important theory in our thesis is that the regret of CBwK-LP-UCBis O($log^2 T$), with T being the total number of rounds played by the BCMAB. To start, we look at the deterministic reward setting and prove some key points. Using these points as a basis, we then shift our focus to the stochastic reward setting. This progression allows us to prove our main theory. Our systematic approach helps us evaluate the effectiveness of our algorithm in different situations.

4.1 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 4.1. 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 4.1. *The regret of algorithm CBwK-LP-UCB with parameter* $\epsilon = \sqrt{\frac{\ln d}{B'}}$, *for* $d = n + 1$, *satisfies*

$$OPT_{LP} - REW_{\text{CBwK-LP-UCB}} \leq O\left(\sqrt{\log(n^2 T)}\right)\left(\sqrt{n \cdot OPT_{LP}} + OPT_{LP}\sqrt{\frac{n}{B'}}\right) + O(n)\log(n^2 T)\log(T) \tag{4.1}$$

Proof. (Overview) Using useful result adapted from Kleinberg [19], in Lemma 4.1 (Section 4.2), we bound the term $OPT_{LP} - f(OPT_{LP})$ for the deterministic (but unknown) setting. In Section 4.3, we extend this for the unknown stochastic setting, as described in Equation 3.8, completing the proof. □

Step 12 in Algorithm 10 uses the multiplicative weights update technique by Freund & Schapire [11]. 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 [19].

Proposition 4.1. *Fix any parameter $\epsilon \in (0,1)$ and any stopping time τ . For any sequence of payoff vectors $\pi_1, \ldots, \pi_\tau \in [0,1]^d$, we have*

$$\forall y \in \Delta[d] \qquad \sum_{t=1}^{\tau} y_t^T \pi_t \geq (1-\epsilon) \sum_{t=1}^{\tau} y^T \pi_t - \frac{\ln d}{\epsilon} \tag{4.2}$$

4.2 Deterministic Rewards

In this section, we consider the setting where pulling an arm i deterministically generates reward μ_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 10 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 $REW_{\text{CBwK-LP-UCB}} = \sum_{t=2}^{T} rew_t$, where rew_t is the reward collected in the round t. We will prove in Lemma 4.1 that $REW_{\text{CBwK-LP-UCB}} \geq W \cdot \frac{OPT_{LP}}{B'}$. For this reason, maximizing W also helps maximize REW.

Let ζ^* 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 4.1. *We claim that there exists a z_t, such that*

$$z_t \in \operatorname*{argmax}_{z \in \Delta[N]} \frac{\mu^T z}{y_t^T M z} \tag{4.3}$$

Proof. z_t is a distribution that maximizes the bangperbuck ratio among all distributions z over arms. Indeed, the argmax in Equation (4.3) 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 ρ, the linear inequality $\rho y_t^T Mz \geq \mu^T z$ 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 4.1. $REW_{\text{CBwK-LP-UCB}} \geq W \cdot \frac{OPT_{LP}}{B'}$

Proof.

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

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

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

The last inequality follows because all components of $M\zeta^*$ are at most B' by the primal feasibility of ζ^*. Hence $REW_{\text{CBwK-LP-UCB}} \geq W \cdot \frac{OPT_{LP}}{B'}{}'$ □

Combining Lemma 4.1 and the regret bound from Proposition 4.1, we obtain $\forall y \in \Delta[d]$

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

The algorithm stops after round T. By this round, either B (from the original setting in Optimization Problem 3.1) has been fully comsumed, 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,

$$\begin{aligned} REW_{\text{CBwK-LP-UCB}} &\geq [(1-\epsilon)(B'-n) - \tfrac{\ln d}{\epsilon}] \cdot \tfrac{OPT_{LP}}{B'} \\ &\geq OPT_{LP} - [\epsilon B + n + \tfrac{\ln d}{\epsilon}] \cdot \tfrac{OPT_{LP}}{B'} \\ &= OPT_{LP} - O(\sqrt{B \ln d} + n) \cdot \tfrac{OPT_{LP}}{B} \\ &\text{where } \epsilon = \sqrt{\tfrac{\ln d}{B'}} \text{ and } d = n+1 \end{aligned}$$

4.3 Stochastic Rewards

Here, we use the techniques from the previous section and provide regret bound for the stochastic setting (Equation 3.8).

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 4.2, we claim there exists a z_t such that

$$z_t \in \underset{z \in \Delta \mathcal{N}}{\operatorname{argmax}} \frac{u_t^T z}{y_t^T M z} \tag{4.4}$$

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

We define a confidence radius $rad(x, N) = \sqrt{\frac{xC_{rad}}{N}} + \frac{C_{rad}}{N}$, where $C_{rad} = \theta(\log ndT)$. We adapt the following result from Babaioff *et al.* [3] and Kleinberg *et al.* [20].

Proposition 4.2. *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* $\forall C_{rad} > 0$

$$Pr[|x - \bar{x}| \leq rad(\bar{x}, N) \leq 3rad(x, N)] \geq 1 - \exp -\omega(C_{rad}) \tag{4.5}$$

Using Proposition 4.2 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* ofCBwK-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 4.2. *In a clean execution of Algorithm PrimalDualBwK with parameter $\epsilon = \sqrt{\frac{\ln d}{B}}$, the algorithm's total reward satisfies the bound*

$$OPT_{LP} - REW_{\text{CBwK-LP-UCB}} \leq 2 \cdot OPT_{LP} \left(\sqrt{\frac{\ln d}{B'}} + \frac{n}{B} \right) + n + \left| \sum_{t=2}^{T} \delta_t^T z_t \right| \tag{4.6}$$

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

Let ζ^* denote an optimal solution of the primal linear program given in Equation 3.3, 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, $\sum_{t=1}^{T} y_t^T M z_t \geq B'$. Once again, since the total cost for any resource is at most n for $t = 1$, $\sum_{t=2}^{T} y_t^T M z_t \geq B' - n$

Let $\bar{y} = \frac{1}{REW_{UCB}} \sum_{t=2}^{T} (u_t^T z_t) y_t$. Assuming a clean execution,

$$\begin{aligned} B' \geq \bar{y}^T M \zeta^* &= \frac{1}{REW_{UCB}} \sum_{t=2}^{T} (u_t^T z_t)(y_t M \zeta^*) \\ &\geq \frac{1}{REW_{UCB}} \sum_{t=2}^{T} (u_t^T \zeta^*)(y_t M z_t) \\ &\geq \frac{1}{REW_{UCB}} \sum_{t=2}^{T} (r^T \zeta^*)(y_t M z_t) \\ &\geq \frac{OPT_{LP}}{REW_{UCB}} \left[(1-\epsilon) y^T \left(\sum_{t=2}^{T} M z_t \right) - \frac{\ln d}{\epsilon} \right] \\ \implies REW_{UCB} &\geq \frac{OPT_{LP}}{B'} [B' - \epsilon B' - n - \frac{\ln d}{B}] \\ \implies REW_{UCB} &\geq OPT_{LP} [1 - \epsilon - \frac{n}{B} - \frac{\ln d}{\epsilon B}] \end{aligned}$$

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

$$\begin{aligned} REW_{\text{CBwK-LP-UCB}} &\geq REW_{UCB} - \sum_{t=2}^{T} (u_t - \mu)^T z_t \\ &= REW_{UCB} - \sum_{t=2}^{T} \delta_t^T z_t \end{aligned}$$

We use the following proposition from Badanidiyuru *et al.* [4].

Proposition 4.3. *(Badanidiyuru* et al. *[4]) Consider two sequences of vectors $a_1, \dots, a_\tau$ and $b_1, \dots, b_\tau$, 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,

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

Taking $a_t = \mu$, $b_t = u_t$ and vector $a_0 = \mu$ in Proposition 4.3, we get

$$\left| \sum_{t=2}^{T} \delta_t z_t \right| \le O(\sqrt{C_{rad} n REW_{\text{CBwK-LP-UCB}}} + C_{rad} n \log T). \tag{4.7}$$

We now combine the results to provide the proof for Theorem 4.1.

Proof. (Theorem 4.1) For $n \ge \frac{B'}{\log dT}$, the bound in Claim 4.2 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 4.2 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 4.7, along with Claim 4.2. □

Chapter 5

Experimental Analysis

This chapter centers around the practical comparison of our proposed algorithms through experimentation. Initially, we discuss the experimental setup for our BCMAB investigation, highlighting the procedure we've followed. To provide a comprehensive view, we examine four distinct scenarios in depth. Following this, we present our findings and engage in an empirical analysis of these results. This careful, methodical evaluation ensures a thorough understanding of how our algorithms perform in varied circumstances.

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

5.1 Experimental Set-up

For the 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-UCB as well as SemiBwK-RRS. We take $\epsilon = 0$ in SemiBwK-RRS, to maintain consistency with what Sankararaman & Slivkins [29] mentioned in their experiments. We report the average taken 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 the effect of the budget on regret.

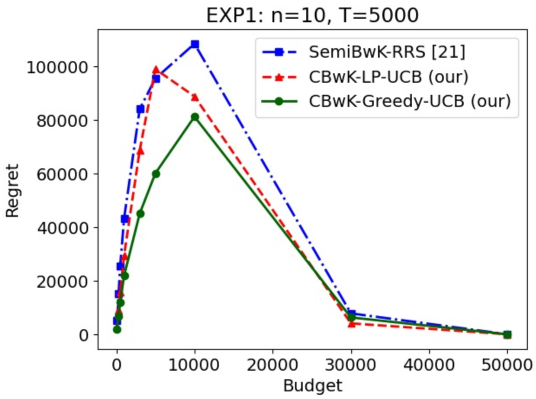


Figure 5.1: Regret with respect to B, for fixed T

EXP2

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

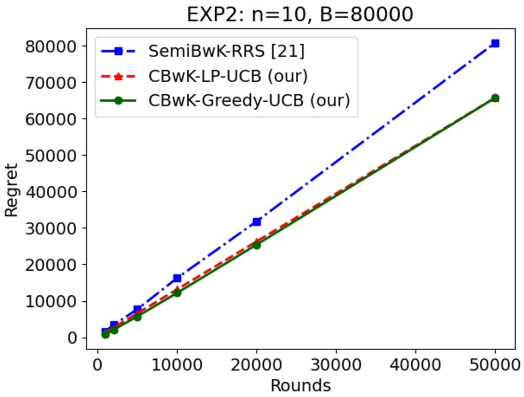


Figure 5.2: Regret with respect to T, for fixed B

EXP3

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

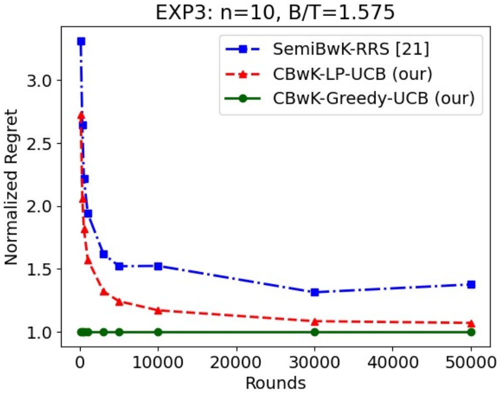


Figure 5.3: Regret with respect to T, for fixed $\frac{B}{T}$

EXP4

For non-i.i.d. arms (Figure 5.4), we vary $T = 100 \longrightarrow 2000$ for a fixed budget/round ratio $B/T = 1.575$ and $n = 4$ to study the 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, and very low rewarding arms.

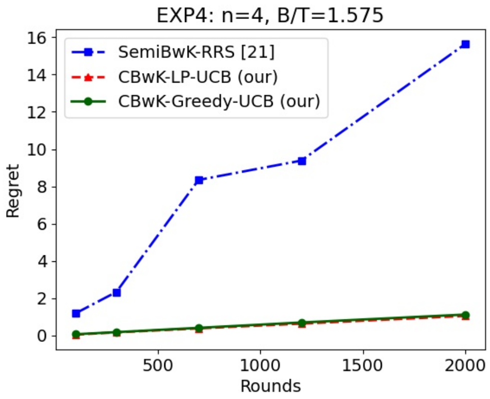


Figure 5.4: Regret with respect to T for fixed B/T, non-i.i.d. arms

5.2 Empirical Analysis

As can be seen from the figures *CBwK-Greedy-UCB achieves the lowest regret in all four experiments. CBwK-LP-UCB performs almost as well 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 OPT_{LP} increases with an increase in B as well as T. Thus, to study the relative performance of the algorithms as T increases, we plot regrets relative to CBwK-Greedy-UCB (CBwK-Greedy-UCB normalized to 1) for EXP3. It clearly indicates the 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.

Chapter 6

Conclusion

In this thesis, we delved into the fascinating world of Multi-Armed Bandits (MABs) and explored various intriguing variants of this problem. These variants included scenarios where we had to consider budget constraints, combine different options in a combinatorial setting, and take into account contextual information. We thoroughly examined relevant research papers that dealt with these complex variations of MABs, highlighting their significance and challenges.

Our main focus was on a specific variant called Budgeted Combinatorial Multi-Armed Bandit (BCMAB). We tackled this problem in both the offline setting, where arm rewards were known in advance, and the online setting, where rewards were unknown. Through our innovative algorithms, namely CBwK-Greedy-UCB and CBwK-LP-UCB, we achieved superior performance compared to existing approaches documented in the literature.

Furthermore, we extended our work to address more general submodular rewards setting known as BCMAB-S. Similar to the BCMAB variant, we tackled this problem in both offline and online settings. Our algorithms, G-EqualDistribution-UCB and G-Single-UCB, were designed to handle this setting effectively and provided promising results.

Overall, our research made significant contributions to understanding and solving the challenging problems posed by variants of Multi-Armed Bandits, particularly in the contexts of budget constraints, combinatorial settings, and submodular rewards.

Bibliography

[1] Abhishek, Kumar, Jain, Shweta, & Gujar, Sujit. 2020. Designing Truthful Contextual Multi-Armed Bandits Based Sponsored Search Auctions. *Page 1732–1734 of: Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*. AAMAS '20. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems.

[2] Auer, Peter, Cesa-Bianchi, Nicolò, & Fischer, Paul. 2002. Finite-Time Analysis of the Multiarmed Bandit Problem. *Mach. Learn.*, **47**(2–3), 235–256.

[3] Babaioff, Moshe, Dughmi, Shaddin, Kleinberg, Robert, & Slivkins, Aleksandrs. 2015. Dynamic Pricing with Limited Supply. *ACM Trans. Econ. Comput.*, **3**(1).

[4] Badanidiyuru, Ashwinkumar, Kleinberg, Robert, & Slivkins, Aleksandrs. 2018. Bandits with Knapsacks. *J. ACM*, **65**(3).

[5] Bhat, Satyanath, Jain, Shweta, Gujar, Sujit, & Narahari, Yadati. 2019. An Optimal Bidimensional Multi-Armed Bandit Auction for Multi-Unit Procurement. *Annals of Mathematics and Artificial Intelligence*, **3**(01).

[6] Chandlekar, Sanjay, Boroju, Arthik, Jain, Shweta, & Gujar, Sujit. 2023. A Novel Demand Response Model and Method for Peak Reduction in Smart Grids – PowerTAC. *Page 2520–2522 of: Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*. AAMAS '23. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems.

[7] Chen, Lixing, Xu, Jie, & Lu, Zhuo. 2018. Contextual Combinatorial Multi-armed Bandits with Volatile Arms and Submodular Reward. *In:* Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., & Garnett, R. (eds), *Advances in Neural Information Processing Systems*, vol. 31. Curran Associates, Inc.

[8] Chen, Wei, Wang, Yajun, & Yuan, Yang. 2013. Combinatorial Multi-Armed Bandit: General Framework and Applications. *Pages 151–159 of:* Dasgupta, Sanjoy, & McAllester, David (eds), *Proceedings of the 30th International Conference on Machine Learning*. Proceedings of Machine Learning Research, vol. 28, no. 1. Atlanta, Georgia, USA: PMLR.

[9] Das, Debojit, Jain, Shweta, & Gujar, Sujit. 2022. Budgeted Combinatorial Multi-Armed Bandits. *Page 345–353 of: Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*. AAMAS '22. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems.

[10] Deva, Ayush, Abhishek, Kumar, & Gujar, Sujit. 2021. A Multi-Arm Bandit Approach To Subset Selection Under Constraints. *In: International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2021*.

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

[12] Gao, Guoju, Wu, Jie, Xiao, Mingjun, & Chen, Guoliang. 2020. Combinatorial Multi-Armed Bandit Based Unknown Worker Recruitment in Heterogeneous Crowdsensing. *Pages 179–188 of: IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*.

[13] Ghalme, Ganesh, Jain, Shweta, Gujar, Sujit, & Narahari, Y. 2017. Thompson Sampling Based Mechanisms for Stochastic Multi-Armed Bandit Problems. *Page 87–95 of: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*. AAMAS '17. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems.

[14] Ghalme, Ganesh, Dhamal, Swapnil, Jain, Shweta, Gujar, Sujit, & Narahari, Y. 2021. Ballooning multi-armed bandits. *Artificial Intelligence*, **296**, 103485.

[15] Gyorgy, Andras, Linder, Tamas, Lugosi, Gabor, & Ottucsak, Gyorgy. 2007. *The on-line shortest path problem under partial monitoring*.

[16] Jain, Shweta, & Gujar, Sujit. 2020. 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.

[17] Jain, Shweta, Ghalme, Ganesh, Bhat, Satyanath, Gujar, Sujit, & Narahari, Y. 2016. A Deterministic MAB Mechanism for Crowdsourcing with Logarithmic Regret and Immediate Payments. AAMAS '16. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems.

[18] Jain, Shweta, Gujar, Sujit, Bhat, Satyanath, Zoeter, Onno, & Narahari, Y. 2018. A quality assuring, cost optimal multi-armed bandit mechanism for expertsourcing. *Artificial Intelligence*, **254**, 44–63.

[19] Kleinberg, Robert. 2007. *Notes from Week 2: Prediction algorithms and zero-sum games*.

[20] Kleinberg, Robert, Slivkins, Aleksandrs, & Upfal, Eli. 2008. Multi-Armed Bandits in Metric Spaces. *CoRR*, **abs/0809.4882**.

[21] Lai, T.L, & Robbins, Herbert. 1985. Asymptotically Efficient Adaptive Allocation Rules. *Adv. Appl. Math.*, **6**(1), 4–22.

[22] Li, Lihong, Chu, Wei, Langford, John, & Schapire, Robert E. 2010. A Contextual-Bandit Approach to Personalized News Article Recommendation. *CoRR*, **abs/1003.0146**.

[23] Lu, Tyler, Pal, David, & Pal, Martin. 2010. Contextual Multi-Armed Bandits. *Pages 485–492 of:* Teh, Yee Whye, & Titterington, Mike (eds), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Proceedings of Machine Learning Research, vol. 9. Chia Laguna Resort, Sardinia, Italy: PMLR.

[24] Madani, Omid, Lizotte, Daniel J., & Greiner, Russell. 2004. The Budgeted Multi-armed Bandit Problem. *Pages 643–645 of:* Shawe-Taylor, John, & Singer, Yoram (eds), *Learning Theory*. Berlin, Heidelberg: Springer Berlin Heidelberg.

[25] Manisha, Padala, & Gujar, Sujit. 2019. Thompson Sampling Based Multi-Armed-Bandit Mechanism Using Neural Networks. *Page 2111–2113 of: Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. AAMAS '19. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems.

[26] Nhat, Van Quoc Truong, Stein, Sebastian, Tran-Thanh, Long, & Jennings, Nick. 2019. What prize is right? How to learn the optimal structure for crowdsourcing contests. *Pages 85–97 of:* Nayak, Abhaya, & Sharma, Alok (eds), *PRICAI 2019: Trends in Artificial Intelligence*, vol. 1160. Springer.

[27] Ravindranath, Saurabh, Baburaj, Rahul, Balasubramanian, Vineeth N., Namburu, NageswaraRao, Gujar, Sujit, & Jawahar, C. V. 2020. Human Machine Collaboration for Face Recognition. *Page 10–18 of: Proceedings of the 7th ACM IKDD CoDS and 25th COMAD*. CoDS COMAD 2020. New York, NY, USA: Association for Computing Machinery.

[28] Reddy, Meghana, Singh, Akansha, Jain, Shweta, & Gujar, Sujit. 2021. Designing Bounded Min-knapsack Bandits Algorithm for Sustainable Demand Response. *In: 18th Pacific Rim International Conference on Artificial Intelligence (PRICAI-2021)*.

[29] Sankararaman, Karthik Abinav, & Slivkins, Aleksandrs. 2017. Semi-Bandits with Knapsacks. *CoRR*, **abs/1705.08110**.

[30] Sharma, Akash Das, Gujar, Sujit, & Narahari, Yadati. 2012. Truthful multi-armed bandit mechanisms for multi-slot sponsored search auctions. *Current Science*, 1064–1077.

[31] Solanki, Sambhav, Kanaparthy, Samhita, Damle, Sankarshan, & Gujar, Sujit. 2022. Differentially Private Federated Combinatorial Bandits with Constraints. *In: In the proceedings of European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD'22)*.

[32] Tran-Thanh, Long, Chapman, Archie C., de Cote, Enrique Munoz, Rogers, Alex, & Jennings, Nicholas R. 2010. Epsilon-First Policies for Budget-Limited Multi-Armed Bandits. *In: AAAI*.

[33] Tran-Thanh, Long, Stein, Sebastian, Rogers, Alex, & Jennings, Nicholas. 2012 (08). Efficient crowdsourcing of unknown experts using multi-armed bandits. vol. 214.

[34] Tran-Thanh, Long, Chapman, Archie C., Rogers, Alex, & Jennings, Nicholas R. 2012. Knapsack based Optimal Policies for Budget-Limited Multi-Armed Bandits. *CoRR*, **abs/1204.1909**.

[35] Tran-Thanh, Long, Venanzi, Matteo, Rogers, Alex, & Jennings, Nicholas R. 2013. Efficient Budget Allocation with Accuracy Guarantees for Crowdsourcing Classification Tasks. *Page 901–908 of: Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems*. AAMAS '13. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems.

[36] Wang, Siwei, & Chen, Wei. 2018. Thompson sampling for combinatorial semi-bandits. *Pages 5114–5122 of: International Conference on Machine Learning*. PMLR.

[37] Wen, Zheng, Kveton, Branislav, & Ashkan, Azin. 2015. Efficient Learning in Large-Scale Combinatorial Semi-Bandits. *Pages 1113–1122 of:* Bach, Francis, & Blei, David (eds), *Proceedings of the 32nd International Conference on Machine Learning*. Proceedings of Machine Learning Research, vol. 37. Lille, France: PMLR.

[38] Zhang, Hao, Ma, Yao, & Sugiyama, Masashi. 2015. Bandit-Based Task Assignment for Heterogeneous Crowdsourcing. *CoRR*, **abs/1507.05800**.

[39] Zhou, Datong P., & Tomlin, Claire J. 2017a. Budget-Constrained Multi-Armed Bandits with Multiple Plays. *CoRR*, **abs/1711.05928**.

[40] Zhou, Datong P., & Tomlin, Claire J. 2017b. *Budget-Constrained Multi-Armed Bandits with Multiple Plays*.