Design and Analysis of Algorithms for Combinatorial Multi-arm Bandit Problems under Complex Environments

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

Master of Science in
Computer Science and Engineering
by Research

by

KUMAR ABHISHEK
201502172
kumar.abhishek@research.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500 032, INDIA
June 2021

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled **"Design and Analysis of Algorithms for Combinatorial Multi-arm Bandit Problems under Complex Environments"** by **Kumar Abhishek**, has been carried out under my supervision and is not submitted elsewhere for a degree.

_____ _____

Date Adviser: Prof. Sujit Gujar

To my Family.

Acknowledgments

I owe my deepest gratitude to my research supervisor, Prof Dr. Sujit Gujar. Without his continuous enthusiasm, positivity, and belief in me, this thesis would not have been possible. His inputs and view on problems have always been really insightful. I feel incredibly fortunate to have him as my advisor. I am grateful to him for guiding me in various kinds of academic and professional activities. I sincerely thank sir, for regular technical/non-technical discussions either on boards or over a cup of tea. I also had the privilege to play cricket with you.

I thank Prof. Y Narahari (IISc Bengaluru) for giving me the opportunity to work as a research intern. There, Dr. Ganesh Ghalme was a mentor to me. I had some great discussions, and his passion for work was inspirational. I am also grateful to my co-authors Ayush Deva, Prof. Shweta Jain, Dr. Ganesh Ghalme, and Prof. Y Narahari, a pleasure to work with. I want to thank Kartikey, Aakash KT, Sankarshan, and Manisha, who helped me by being a constant reviewer.

No words can describe my gratitude to my parents and brother, who have been continuously supportive and a source of inspiration.

I want to thank my friends Amardeep, Kartikey, Aakash KT, Yash Goyal, Aayush Tiwari, Mayank Garg, Lakshya, Yaghyavardhan, Sankarshan, Swapnil Nayan, Aashish, Ayush Rai, Anirudh, Priyansh, and Gokul, without whom life at IIIT would not be as much fun. I want to express my gratitude to my friends from school, still supporting and cheering me up to keep the mojo going, Adarsh, Raj Ratan, Rishav, Shashi, Aditya Sahi, Shakti, and Atish.

Abstract

Online learning problems are among the most abundant machine learning problems, such as advertisement selection, movie recommendation, and node or link prediction in evolving networks. In online learning, one must make online, real-time decisions and continuously improve performance with the sequential arrival of data. The multi-armed bandit (MAB) problem is one of the fundamental online learning problems that captures the classic exploration vs. exploitation dilemma. As per the applications' requirements, new variants of MAB have been developed to tackle the problem. For example, strategic bidding in the online learning environment (MAB mechanism); availability of extra information to make the decision (Contextual MAB); selecting multiple arms instead of the single-arm (Combinatorial MAB); a varying set of available arms (Sleeping MAB), etc. We identify some of the problems and appropriately address them by proposing efficient algorithms. We provide theoretical guarantees for the performance of the proposed algorithms and showcase their efficacy through simulations. Following are the problems we address in this thesis:

- Designing truthful contextual multi-armed bandits with the backdrop of sponsored search auction [3]. We tackle the drawbacks of the current state-of-the-art.
- Propose an efficient algorithm for sleeping combinatorial multi-arm bandit problem when we consider general reward function (satisfying mild smoothness conditions) [2].
- We take a multi-arm bandit approach to subset selection under constraints in an unknown environment [46].

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The last two decades have seen a surge in real-world applications where the *Multi-Armed bandit* (MAB) techniques have found their applicability. Typically, a planner has to choose (aka pull) an arm (aka choice) from a fixed set of arms at each round. The planner receives a reward by pulling an arm and does not receive any other arm's reward. The planner aims to maximize its cumulative reward through multiple rounds. Here, the challenge arises as to the quality of the arm in terms of rewards is unknown to the planner. Hence, the planner must intelligently choose the agents through the rounds to identify the best rewarding arm from the other arms by using the least number of rounds. This situation leads to the classic exploration/exploitation dilemma and has been modeled as *Multi-Armed Bandit* (MAB) problem.

Applications like internet advertising, involve learning an unknown parameter that can be learned through repetitive interactions. In the modern advertising paradigm, the platforms on the internet show the advertisements to the users arriving on the platform. The advertisers pay a certain amount to the platform if the advertiser's ad obtains a click from the user; otherwise, they pay nothing. This mode of payment is also known as *pay-per-click* model. Hence, to maximize the platform's revenue, it needs to select the ad which is most likely to get the click. Here, notice that the probability of an ad getting a click is unknown to the platform. The probability of getting a click is also known as *click-through rate* (CTR). The platform needs to learn it through sequential interactions with the users. Hence, we can model this problem as a MAB problem. Similarly, this scenario arises in other problems like crowdsourcing – the quality of work done by workers is unknown to the platform; smart grids – in demand response, the platform needs to select the people who are willing to reduce their energy consumption to manage the peak demand. The probability of the person to positively reduce the energy consumption is unknown to the platform.

In many applications, the problem is not completely encapsulated within the classic MAB setting. To make the MAB problem come closer to real-world applications, researchers have developed new MAB settings on top of the stochastic MAB setting. The variants are often modified based on the particular requirements of the application in consideration. Each additional variant makes the model more complex

and requires a careful approach to design the algorithm to address the problem. Additionally theoretical analysis of the algorithms becomes further challenging.

1.1 Problems Addressed in the Thesis

In this thesis, we identify three real world problems which also pose theoretical challenges to solve them. We model these problems as MAB problems. Note that the problems do not fit the traditional MAB framework and thus, require more complex settings to address them. We abstract out the situation so that the technique developed is useful in many similar situations. We motivate the problem through a specific application for better exposition. Though such a situation arises in other applications as well. Following are the problems we address in the thesis:

- *P1 – Internet advertising*: As mentioned above, the platform requires learning the CTRs of the ads through sequential interactions with its users. Further, in the case of internet advertising, typically, the platform conducts an auction to select the ads where the advertisers bid their valuations of getting a click. Here the agents have a private valuation for the click, which is unknown to other advertisers and the platform. This auction is known as *Sponsored Search Auction* (SSA). The advertisers being rational and strategic players want to maximize their utility. Hence, they may falsely report their true valuations. For the effective selection of ads, the platform needs to elicit true valuations. Additionally, some extra information (context) is often available to the advertising platform to personalize the ad to the user. This personalizing leads to higher clicks on the ads, subsequently, higher revenues. In summary, we address the problem where we require to learn the unknown parameter in an online personalized manner with the available context in the presence of strategic advertisers.

- *P2 – Crowdsourcing*: The online crowdsourcing platforms are emerging to be a crucial marketplace for employers to get human computation tasks performed in a timely, scalable and cost-effective manner. Simultaneously, this provides an opportunity for the workers to make money without much least amount of overhead. For example, crowdsourcing is used widely in the AI community to collect labels for large-scale learning problems. In such cases, the platform usually selects multiple workers to perform one task to determine the correct labels, also known as the crowd's wisdom. The employers desire that the task are completed with high accuracy. However, as workers' qualities are unknown to the platform, they can either explore the workers to learn their qualities or choose to exploit the best set of workers identified. Note that, here, we consider selecting a set of multiple agents instead of a single agent. Additionally, the assumption of all workers being available at each task is not realistic. In our problem, we relax this assumption and allow the set of available agents to varying from one round to the next. In summary, here, we address the problem of sequential selection of multiple agents in an unknown environment, where the availability of the agents may vary.

- *P3 – Items Procurement*: E-commerce platforms, like Amazon and Alibaba, have several sellers registered on their platform. For each product, the platform needs to select a subset of sellers to display on its page. Each seller has a cost and quality associated with the product it offers. The platform wants to ensure that it avoids low-quality sellers and does not display only the searched product's high-cost variants. We model this problem as a subset selection problem, where the platform needs to select a subset of these sellers. The platform wants to maximize its utility (i.e., revenue - cost) while ensuring that the procured units' average quality is above a certain threshold to guarantee customer satisfaction and retention. We refer to the constraint to maintain that the average quality procured products satisfy a certain threshold as quality constraints (QC). Often the platform does not know its sellers' quality at the time of registration until its products are procured and sold in the market. In summary, we address the problem of subset selection problem such that we minimize the cost while satisfying QC where the qualities of sellers are unknown.

1.2 Contributions and Thesis Outline

We model the above problems in their appropriate variant of the MAB problem and provide novel solutions. Following is the organization of the Thesis Chapter 2 provides an overview of MAB and its derivatives. In Chapter 3, we review the important results in mechanism design theory. Each of the Chapters 4-6 is organized as an independent paper. The state of the art related to each of the above problems is reviewed in the corresponding chapter. Each chapter proceeds with the model and presents our results. The thesis is concluded with a summary in Chapter 7. Our contributions are as follows:

- Chapter 4: In this chapter we address the problem from scenario arising in P1. We model the problem as designing a truthful contextual multi-armed bandit mechanism to leverage the contextual information while learning the unknown parameters in strategic agents' presence. Prior to this contextual information in SSA is considered only in [56]. The authors proposed a novel, theoretically sound, deterministic, exploration-separated mechanism that offers strong game-theoretic properties. However, it faces multiple practical challenges. Hence we design and analyze the mechanisms in the presence of strategic agents, random context-arrivals, and stochastic clicks. Our goal is to design a non-exploration-separated, ex-post truthful mechanism that (i) learns CTRs efficiently (minimizes regret), (ii) may not need prior knowledge of T, and (iii) does not have free rounds. We provide theoretical guarantees of the mechanism and show its efficacy through simulations.

- Chapter 5: In this chapter we tackle the P2. We model the problem as a sleeping combinatorial MAB problem, a combination of sleeping and combinatorial stochastic bandits. The sleeping variant of MAB captures where the set of available agents vary from round to round. Combinatorial bandits capture the case where instead of selecting one agent, we have to select multiple agents from the set of available agents. We consider a general reward function (under mild smoothness

constraints), whereas the existing literature assumes lot of structure and restrictions on the reward function. We illustrate how to adapt existing algorithm for this problem. The key challenge is analyzing it in the new setting . We develop new theoretical framework for the same. We also provide validation of our theoretical results through experiments.

- Chapter 6: In this chapter we tackle the P3. We take the MAB approach for subset selection while considering the constraint in an unknown environment. In this case, each agent is associated with quality (unknown to the platform) and cost. The platform wants to maximize its utility while ensuring that the selected agents' average quality is above a certain threshold. To the best of our knowledge, we are first to address the problem where both the constraint and the utility function depend on the unknown parameter. We propose algorithms to address the problem and show its correctness in terms of algorithm satisfying the constraints. We also provide a bound on its performance and show its efficacy through experiments.

Chapter 2

Multi-Armed Bandits and its Derivatives: An Overview

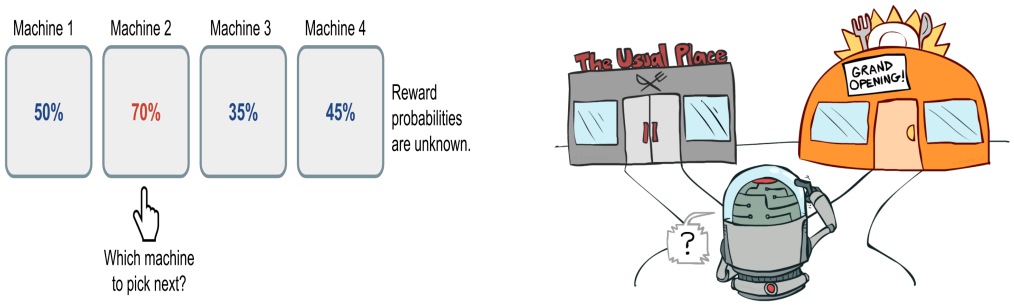


Figure 2.1: Examples of exploration/exploitation dilemma

Multi-armed bandits (MAB) is a rich, multi-disciplinary area studied since 1933 [108]. MAB has attracted the interests of many researchers in the past 10-15 years. The stochastic multi-armed bandit (MAB) problem is one of the fundamental online learning problems that captures the famous exploration vs. exploitation dilemma.

An algorithm operating in an uncertain environment must optimally trade-off the gathering of new information about the environment with the optimal use of information-at-hand to choose an action that performs well against a given performance metric. In a classical stochastic MAB setup, an algorithm has to choose (or pull) a single action (i.e., arm) at each time instant and receive a reward corresponding to a chosen arm. Each arm's reward is an independent sample of a fixed, but unknown stochastic distribution [79]. The goal is to minimize the expected *regret*, i.e., the difference between the expected cumulative reward of the best offline algorithm with known distributions and the expected cumulative reward. It is called "regret" because this is how much the algorithm "regrets" not knowing the best arm.

The term "Multi-Armed Bandit" comes from the example where a gambler can play across different similar-looking slot machines. Each machine provides a random reward from a probability distribution specific to that machine that is unknown to the gambler. To maximize his earnings, he has to decide which machine to play, how many times to play each machine, and in which order to play them. In most real-life cases, the gambler loses money while playing. Hence the slot machines depict the nature bandits. Kindly refer to [34, 81, 104] for comprehensive survey of MAB literature.

2.1 Applications

Through the years MAB has been found its applicability in many different domains. It has been used by many tech giants for examples Amazon [63], Facebook, Google [101, 102], LinkedIn [5, 6], Microsoft [61], Netflix [11].

The classic MAB model is an abstraction for a recurring feature observed in many of the applications. Following is the sample of such applications:

- Clinical trials: The original application of MAB was in clinical trials. When a patient visits a doctor, the doctor can prescribe one of several possible treatments. The patient observes and provides feedback on the effectiveness of the prescribed treatment to the doctor. The objective of the doctor here is to attain useful scientific data while minimizing harm to the patients. Here each treatment can be considered as an arm, and effectiveness of the treatment as the reward. Here, similar exploitation vs. exploration dilemma arises, i.e., whether to prescribe the best treatment based on information at hand (exploitation) or prescribe other sub-explored treatment to gather more information about its efficacy.

Figure 2.2: Major Applications

- Internet advertisement: With the rise of the internet/web, the advertisement mode changed. The main source of income of many big tech giants like Google, Facebook, etc., is through online advertisement. When a user visits a web page in web advertising, its underlying algorithm needs to choose some ads from many possible ads to display. If the user clicks on the displayed ad, the advertiser needs to pay some amount to the company. This payment is popularly known as *pay-per-click* model. Hence, the objective of the company is to maximize the number of clicks. Here, each ad serves as an arm in the MAB problem and the clicks as rewards. Note that the click can only be observed if and only if the ad is chosen to be displayed. Here arises the famous exploitation vs. exploration dilemma, as the probability of an ad getting click is unknown.

- Crowdsourcing: In online crowdsourcing platforms the requester posts the tasks completed where various registered workers are willing to perform the tasks. Each worker has a fixed quality associated with them, which indicates the workers' ability to complete the task successfully. The requester wishes to assign the tasks to workers with the highest quality. However, these qualities

are typically unknown—the requester quests to maximize the expected number of tasks successfully. To maximize the quality, the requester assigns the tasks to the workers one by one to learn their qualities. Here, the workers serve as arms and the completion of the tasks allocated to the worker as a reward. Observe similar exploitation vs. exploration dilemma arises, i.e., whether to allocate the task to the worker to the best available task (exploitation) or to the worker who has not been sufficiently allocated to obtain better estimates of the quality of the workers (exploration).

The above applications have a common underlying challenge, which involves sequential decision-making in an uncertain environment. They suffer from similar exploitation vs. exploration dilemmas. These kinds of problems naturally fall into Multi-Armed Bandit (MAB) problems. Following are the problems which the researchers have discovered to lie in MAB paradigm:

- Healthcare: Initially the study of MAB was inspired from the problem of clinical trials [59, 108]. Bandit-like designs for medical trials belong to the realm of adaptive medical trials [43], which can also include other "adaptive" features such as early stopping, sample size re-estimation, and changing the dosage.
- Web Related:
  - Ad placement [80, 95, 96]
  - Web Search [98]
  - Recommendation system
    - News [33, 84, 85, 86]
    - advertisement [3, 56, 100]
    - Music [89]
  - Website optimization [63]
- Economics related:
  - Design of crowdsourcing platforms [27, 68]
  - Design of auctions [3]
  - Design of incentives and information structures - Smart grids (demand response) [71, 69]
  - Optimizing seller's product offerings, a.k.a. dynamic assortment [8]
  - Optimizing seller's prices, a.k.a. dynamic pricing or learn-and-earn [88]
- Networking:
  - Internet routing and congestion control [48, 72]
  - Radio Networks [12, 18, 78]
- Improving Machine learning [31]:

- Algorithm Selection [51]
- Hyperparameter Optimization [73]
- Feature Selection [32, 113]
- Bandit for Active Learning [30]
- Clustering [105]

2.2 MAB setting

Let $[n] = \{1, 2, \cdots, n\}$ represent the set of n base arms. In a classical stochastic multi-armed bandits (MAB) problem, at each discrete round t, a planner pulls a single arm $i_t \in [n]$ and observes a random reward $X_{i_t,t}$. Rewards are not observed for the arms $j \neq i_t$ at round t. The random variables $(X_{i,t})_t$ are identical and independently distributed according to a distribution $\mathcal{D}_i(\mu_i)$. Here, μ_i is the mean of distribution $\mathcal{D}_i$. We consider that $\mathcal{D}_i$ is supported over $[0, 1]$. The reward distributions $(\mathcal{D}_i)_{i\in[n]}$ are unknown to the algorithm. To compare performance of different algorithm's, the notion of regret is introduced. Basically, it captures the loss in expected reward due to lack of knowledge of $\mu_i's$. Formally, expected regret is defined as, $\mathcal{R}_{\text{ALG}}(T) = \mathbb{E}[\sum_{t=1}^{T}(X_{i^\star,t} - X_{i_t,t})]$. Here, $i^\star = \arg\max_i \mu_i$ denotes the best arm. Equivalently, we can write the expected regret as $\mathcal{R}_{\text{ALG}}(T) = \sum_i \Delta_i \cdot \mathbb{E}[N_{i,T}]$, where $\Delta_i = \mu_{i^\star} - \mu_i$ and $N_{i,t}$ represents the number of times arm i is pulled till t.

There have been works in the literature that have established a lower bound on regret, which applies to the bandit algorithms. A lower bound indicates the minimum regret every bandit algorithm will suffer; that is, no bandit algorithm can achieve better regret in the setting. For the classic stochastic MAB, the established instance-independent lower bound is $\Omega(\sqrt{T})$ and the instance-dependent lower bound is $\Omega(\log T)$, where T is the total number of rounds.

An ideal algorithm will be such whose upper bound on regret matches the established lower regret bound of the problem in consideration. We call the algorithm to be near-optimal if it is optimal up to polylogarithmic factors or, better, for example, constant factors. With abuse of notations, we use optimal and near-optimal interchangeably. As stated before, there is an intrinsic *trade-off* between *exploiting* the current knowledge to select the arm that seems to yield the highest rewards empirically, and *exploring* further the other arms to identify with better precision which arm is the best. The research done in this quest indicates that the key to obtaining a good strategy for this problem has been recognized to be, in a certain sense, to simultaneously perform exploration and exploitation. The two popular algorithms to do so efficiently has been, UCB1 (Upper Confidence Bound 1) and Thompson sampling, are two popular algorithms that are optimal in stochastic MAB problem.

2.2.1 UCB1

The algorithm UCB1 [16] follows the idea of a simple heuristic principle known as optimism in the face of uncertainty. UCB1 does not have to assume any preliminary knowledge about the reward

distributions (apart from the fact that their support is $[0, 1]$). This idea is very general and applies to many sequential decision-making problems in uncertain environments. The algorithm maintains an optimistic upper bound on each arm's mean reward and selects the arm with the maximal upper bound.

Algorithm 1 UCB1

Initialization: $N_{i,0} = 0, \overline{\mu}_{i,0} = 1, X_{i,0} = 0, \forall i \in [n]$
for $t = 1, 2, \ldots$ **do**
  if $\exists j \in [n], s.t., N_{i,t} = 0$ **then**
    Select $i_t = j$
  else
    Evaluate $\forall i, \overline{\mu}_{i,t} = \hat{\mu}_{i,t} + \sqrt{\frac{2 \ln t}{N_{i,t}}}$
    Select $i_t = \arg\max_i \overline{\mu}_{i,t}$, and observe reward $X_{i_t,t}$
    Update, $N_{i_t,t} \leftarrow N_{i_t,t-1} + 1$; $X_{i_t,1:t} \leftarrow X_{i_t,1:t-1} + X_{i_t,t}$; $\hat{\mu}_{i_t,t} \leftarrow \frac{X_{i_t,1:t}}{N_{i_t,t}}$

The equation $\overline{\mu}_{i,t}$, is referred as upper confidence bound (UCB) estimate of arm i at time t. UCB is defined as the sum of two terms. The first term is the empirical mean of the arm till time t. The second term is related to the size of the one-sided confidence interval for the empirical reward within which the true expected reward falls with overwhelming probability. Following, we provide intuition for the working of UCB1. An arm j is pulled on time t if it has the highest UCB value, which can happen due to two causes: the first term (empirical mean) is large, in which case this arm is likely to have a high reward and/or the second term (confidence interval) is large in which case this arm has not been explored much. Observe that the first term advocates exploitation whereas the second term exploration, hence the summation of both terms provides a way to resolve the exploration/exploitation dilemma.

Theorem 1. *[16] The expected regret incurred by UCB1 after T rounds is at most,*

$$\left[\sum_{i:\mu_i<\mu^\star} \left(\frac{\ln T}{\Delta_i}\right)\right] + \left(1 + \frac{\pi^2}{3}\right)\left(\sum_{j=1}^{n} \Delta_j\right).$$

2.2.2 Thompson Sampling

Thompson sampling is a Bayesian algorithm that was first proposed in 1933 [108]; for allocating experimental effort in two-armed bandit problems arising in clinical trials. It is a member of the family of randomized probability matching algorithms. The idea of Thompson sampling is to randomly draw each arm according to its probability of being optimal. For ease of exposition, we focus on the Thompson Sampling algorithm for the Bernoulli bandit the problem, i.e., when the rewards are either 0 or 1, which is extendable for general reward distribution with support $[0, 1]$ [10].

In Thompson sampling algorithm, as information is gathered, beliefs about action rewards are carefully tracked. By sampling actions according to the posterior probability that they are optimal, the algorithm continues to sample all actions that could plausibly be optimal while shifting sampling away

Algorithm 2 Thompson Sampling

1: **Initialization:** $a_i = 0, b_i = 0, \forall i \in [n]$
2: **for** $t = 1, 2, \ldots$ **do**
3: $\forall i$, sample κ_i from $Beta(a_i + 1, b_i + 1)$
4: Select arm $i_t = \arg\max \kappa_i$, and observe reward $X_{i_t,t}$
5: If $X_{i_t,t} = 1$, then $a_{i_t} = a_{i_t} + 1$ else $b_{i_t} = b_{i_t} + 1$

from those unlikely to be optimal. Roughly speaking, the algorithm tries all promising actions while gradually discarding those believed to be sub-optimal.

Theorem 2. *[10] For n-armed stochastic bandit problem, Thompson sampling algorithm has expected regret,*

$$\mathcal{R}_{\text{ALG}}(T) \leq O\left(\left(\sum_{i:\mu_i<\mu^\star} \frac{1}{\Delta_i^2}\right)^2 \ln T\right)$$

in T rounds, where $\Delta_i = \mu^\star - \mu_i$.

2.2.3 MAB variants

Note that many applications mentioned in Section 2.1 do not immediately fit in the classic MAB template and require additional dimensions of consideration. Through the years, many works have considered various "dimensions" along which the models can be made more expressive and closer to real-world problems, which has matured Multi-armed bandits into a rich and diverse problem space that keeps growing. Each dimension gave rise to a prominent line of work. Following are some of the high-level dimensions considered by the researchers, which can be further branch into other sub-dimensions:

- Structured actions: In this variant, the algorithm may require pulling a subset of arms instead of a single-arm that follows some structure.
  - Single pull: When the algorithm is required to pull exactly one arm.
  - Combinatorial pull: When algorithm can pull more than one arm.
- Rewards model: Rewards model signifies how the rewards are generated.
  - Stochastic rewards: The reward for each arm is drawn independently from a fixed distribution that depends on the arm.
    - Stationary distribution
    - Non-stationary distribution

    Note that in this thesis, we have restricted our work to the stationary distribution.

  - Adversarial rewards: The rewards in this model can be arbitrary in the sense that it seems to be chosen as if an "adversary" trying to fool the algorithm to chose a suboptimal arm. Based on the capacity of the adversaries capacity to observe the working of the algorithm, there can be two types of adversary
    * Oblivious adversary: Here, the adversary chooses the rewards independent of the forecaster's actions, i.e., being oblivious to the algorithm's performance. For example, when the adversary fixes each round's rewards for all arms before the start of the sequential decision making.
    * Non-oblivious adversary: When the adversary may adapt to the algorithm's past behavior, it is referred to as a non-oblivious adversary. For instance, in the rigged casino, the owner may observe the way a gambler plays to design even more evil sequences of gains.
  - Markovian rewards: Here, the reward processes are neither stochastic nor adversarial. More precisely, arms are associated with their respective Markov processes, each with its own State-space and rewards.

- Feedback: There can be different levels of how much information in terms of rewards is available after each round to improve the arms' estimation. In bandit literature majorly three levels of feedback have been considered:
  - Bandit feedback: When the algorithm observes the chosen arm's reward and no other feedback about other arms is observed.
  - Partial feedback: When some information is revealed, in addition to the reward of the chosen arm.
  - Full feedback: When the algorithm observes the rewards for all arms that could have been chosen.

- Contextual information: A natural extension of the multi-armed problem is where there is some available associating side information with each arm. The algorithm's objective changes in this setting from finding a single optimal arm to learning a policy to map contexts to its respective optimal arm. For example, in the News recommendation website, user location and demographics can be used as contexts for users and news articles as arms. Here the user living in India would prefer to read news about India than the news of any other countries.

- Constraints: The algorithm can be subject to global constraints that bind across arms and rounds. For examples:
  - Elicitation of true valuation: Setting where there is private information associated with arms along with an unknown parameter required to be estimated. In this setting, the planner

generally desires to elicit true private information, which constraints how the unknown parameter is estimated. This class of problems lies in the intersection of mechanism design and MAB. (Example auction in unknown setting)

- the gambler has budgetary constraints on playing the slot machines, in crowdsourcing the requester has budget and time constraints for completing the tasks [27] etc.

- Availability of arms: In many practical scenarios, the subset of arms available to the algorithm for the selection varies. For example, in network routing problem, some of the routes are unavailable at some point in time due to router or link crashes, a gambler playing slot machines, some of the slot machines might be occupied by other players at any given time [75].

By no means the above list is exhaustive. There are many other dimensions which have been addressed in the literature appropriately as per applications' requirement. We refer the reader to [34, 81, 104] for further details on multi-armed bandits and their applications. In the following section, we provide an overview of contextual MAB that is an interesting and popular extension of MAB.

2.3 Combinatorial MAB

A well-explored variant of MAB is the setting where multiple arms (a subset of arms) are pulled simultaneously instead of a single arm. This setting has been studied and is referred to as Combinatorial MAB (CMAB) [35, 39, 41, 45, 52, 53, 60, 77, 83, 94, 114, 115]. In the context of combinatorial MAB, the set of arms is also referred to as base arms and the subset of base arms form and is referred to as super-arm. This variant provides an abstraction to many real-world decision problems. For instance, in an online advertising setup, the platform selects multiple ads to display at any point in time [52]; in crowdsourcing, the requester chooses multiple crowd workers at the same time [112] and in network routing algorithm has to choose a path instead of a single edge [76, 106].

There are three prominent reward feedback types: bandit feedback, semi-bandit feedback, and full information setting. Semi-bandit feedback is the most popular of the lot where the rewards are observed for each arm pulled by the algorithm along with a single reward for pulling the subset of arms. This single reward is the output of an underlying function that takes the pulled subset of arms as input. This CMAB setting can be branched where the reward function is linear with respect to the individual rewards received for each pulled base arm [77] or the reward function can be non-linear [41, 52] for example, submodular function. [77] proposed a UCB-like algorithm CombUCB1 for a stochastic combinatorial semi-bandit setting where a learning agent chooses a subset of base arms at each round subject to constraints and then observes stochastic reward for the chosen arms and receives their sum as a reward.

Theorem 3. *[77] In any $(n, K^{'})$ stochastic combinatorial semi-bandit, the regret of CombUCB1 is bounded by*

$$nK^{'}\frac{534}{\Delta_{\min}}\log T + \left(\frac{\pi^2}{3}+1\right)n^2K^{'}$$

where, $K^{'}$ is maximum number of chosen arms; $\Delta_{\min}$ is minimum gap of sub-optimal solutions.

Algorithm 3 CUCB

Initialization:
for $i \in [k]$ **do**
    $N_{i,0} = 0$, $\overline{\boldsymbol{\mu}}_{i,0} = 1$, $X_{i,0} = 0$
for $t = 1, 2, 3, \ldots$ **do**
    if $\exists j \in [n]$, such that, $N_{j,t} = 0$ **then**
        Select super-arm $S_t \subseteq [n]$, such that $j \in S_t$
    else
        $S_t = \text{ORACLE}(A_t, \overline{\boldsymbol{\mu}}_t)$
    Observe: Semi-bandit feedback as $X_{j,t} \in \{0, 1\}, \forall j \in S_t$ and $R_{S_t}(\boldsymbol{\mu})$;
    Update:

- $N_{i,t} = \begin{cases} N_{i,t-1} & \text{if } \forall i \notin S_t \\ N_{i,t-1}+1 & \text{if } \forall i \in S_t \end{cases}$
- $\overline{\mu}_{i,t} = \frac{X_{i,1:t}}{N_{i,t}} + \sqrt{\frac{3\log(t)}{2N_{i,t}}}$
- $X_{i,1:t} = \begin{cases} X_{i,1:t-1} & \text{if } \forall i \notin S_t \\ X_{i,1:t-1} + X_{i,t} & \text{if } \forall i \in S_t \end{cases}$

[41] consider the general CMAB framework where reward function can also possibly be non-linear (satisfying some mild smoothness constraints). They assume the availability of an (γ, β)-ORACLE.

Definition 1. *[41]* **(γ, β)-ORACLE** *– For some $\gamma, \beta \leq 1$, the* ORACLE *with success probability β outputs a super-arm whose expected reward is at least γ fraction of the optimal expected reward.*

For the general CMAB framework, they provide the CUCB (combinatorial upper confidence bound) algorithm an extension of UCB1 algorithm.

Theorem 4. *[41] The $(\gamma, \beta)-$approximation regret of the CUCB algorithm in T rounds using an $(\gamma, \beta) -$ ORACLE is at most*

$$\left(\frac{6\ln T}{(f^{-1}(\Delta_{\min}))^2} + \frac{\pi^2}{3} + 1\right) \cdot n \cdot \Delta_{\max}.$$

where $f(\cdot)$ is the bounded smoothness function.

2.4 Contextual MAB

An exciting and useful variant of MAB is Contextual MAB (ConMAB) [1, 15, 80, 84], where some additional information (context) is available to the algorithm before selecting the arm to pull. This setting is more challenging than the standard MAB setting. It might be possible that different arms are optimal for different contexts against the standard MAB setting, where typically a single arm is optimal. ConMAB has found its applications primarily in recommender systems, for example, news recommendation, ad selection, etc. The primary motivation is that a user with a known "user profile" arrives in each round, and the context is the user profile. The algorithm can personalize the user's experience. Natural application scenarios include choosing which news articles to showcase, which ads to display, which products to recommend, or which webpage layouts to use. Rewards in these applications are often determined by user clicks, possibly in conjunction with other observable signals correlated with revenue and/or user satisfaction. Naturally, rewards for the same action may be different for different users.

In ConMAB, the definition of regret in the naive stochastic MAB setting is not sufficient to capture the algorithms' performance. Hence, the notion of "contextual regret" is introduced where optimality is defined with respect to the best policy (i.e., mapping from contexts to arms) rather than the best arm. Often, both arms and contexts are represented by sets of features. The quality of an arm is a function of the feature vector of the arm and the context. It is important to emphasize that the arms' feature vector is unknown, which needs to be estimated in an online manner. In general, we assume the feature vector of the arms and context are linearly related. Given the effectiveness of UCB methods for context-free bandit algorithms, the researchers have explored similar algorithms for contextual bandit problems. LinRel [15], LinUCB [84] and SupLinUCB-BaseLinUCB [44] are some popular algorithms which tackles the linear contextual MAB problem.

2.4.1 ConMAB setting

A contextual $n-$armed Multi-Armed Bandit (MAB) problem proceeds in discrete rounds $t = 1, 2, \ldots, T$. At each round t:

1. Observe a context $x_t \in [0, 1]^d$ at round t.

2. Based on the history, h_t, of allocations, observed rewards, and the context x_t, algorithm select an arm I_t.

3. A reward r_{I_t} is observed.

4. Update $h_t = h_{t-1} \cup \{x_t, \{I_t\}, \{r_{I_t}\}\}$.

5. The algorithm then improves its arm-selection strategy with new observation $(x_t, \{I_t\}, \{r_{I_t}\})$. No reward is observed for the arms that are not selected.

2.4.2 LinUCB

Algorithm 4 LinUCB

Inputs: $\alpha \in \mathbb{R}_+$
Initialization:
for all $i \in [n]$ **do**
    $A_i \leftarrow I_d$ (d-dimensional identity matrix)
    $c_i \leftarrow 0_{d\times 1}$ (d-dimensional zero vector)
for $t = 1, 2, 3, \ldots, T$ **do**
    Observe context of user as x_t
    for all $i \in [n]$ **do**
        $\hat{\theta}_i \leftarrow A_i^{-1} c_i$, $\mu_{i,t}^{+} \leftarrow \hat{\theta}_i^{\intercal} x_t + \alpha\sqrt{x_t^{\intercal} A_i^{-1} x_t}$
    $I_t = argmax_{i\in[n]}\, \mu_{i,t}^{+}$, Observe $r_{I_t} \in \{0, 1\}$
    $A_{I_t} \leftarrow A_{I_t} + x_t x_t^{\intercal}$, $c_{I_t} \leftarrow c_{I_t} + r_{I_t} x_t$

Li *et al.* [84] took contextual-bandit approach to personalized web-based services such as news article recommendation. ***LinUCB*** [84] is a generic ConMAB algorithm that efficiently learns the quality of an arm where the reward model is linear in terms of context and the unknown parameters. ***LinUCB*** captures the contextual information in ConMAB setting by using A_i and c_i for each agent i, where A_i summarizes the information about contexts and c_i corresponding clicks. Motivated by ***UCB*** [16], ***LinUCB*** maintains upper confidence bound (UCB) for each agent i as $\mu_{i,t}^{+} \leftarrow \hat{\theta}_i^{\top} x_t + \alpha\sqrt{x_t^{\top} A_i^{-1} x_t}$ where $\hat{\theta}_i \leftarrow A_i^{-1} c_i$ and α is learning parameter. At round t, the algorithm selects the agent I_t with the highest UCB $\mu_{i,t}^{+}$. The statistics for the selected agent I_t is updated as $A_{I_t} \leftarrow A_{I_t} + x_t x_t^{\intercal}$, $c_{I_t} \leftarrow c_{I_t} + r_{I_t} x_t$, where r_{I_t} is the reward. The main idea of LinUCB is to compute the expected reward of each arm by finding a linear combination of the previous rewards of the arm. LinUCB decomposes the feature vector of the current round into a linear combination of feature vectors seen on previous rounds and uses the computed coefficients and rewards on previous rounds to compute the expected reward on the current round.

2.4.3 SupLinUCB and BaseLinUCB

Li *et al.* [84] showed that the LinUCB algorithm is sufficient and effective in practise through experiments on real-world data. But they did not provide the theoretical guarantees if the algorithm due to the technical difficulty in analyzing it. The difficulty arises as we need the predicted set of rewards on the current round to be computed from a linear combination of rewards that are independent random variables, in order to apply the Azuma/Hoeffding inequality. However, LinUCB has the problem that predictions in later rounds are made using previous outcomes. To handle this problem, [44] modified the algorithm into BaseLinUCB (Algorithm 5) which assumes statistical independence among the samples, and then use a master algorithm SupLinUCB (Algorithm 6) to ensure the assumption holds.

Algorithm 5 BaseLinUCB

Inputs: $\alpha \in \mathbb{R}_+$, $\Psi_{i,t} \subseteq \{1, 2, \ldots, t-1\}$
$A_{i,t} \leftarrow I_d + \sum_{\tau \in \Psi_{i,t}} x_\tau^\intercal x_\tau$
$c_{i,t} \leftarrow \sum_{\tau \in \Psi_{i,t}} r_{i,\tau} x_\tau$
$\theta_{i,t} \leftarrow A_{i,t}^{-1} c_{i,t}$
Observe context vector as $x_t \in [0,1]^d$
for $i \in [n]$ **do**
    $w_{i,t}^s \leftarrow \alpha \sqrt{x_t^\intercal A_{i,t}^{-1} x_t}$
    $\hat{r}_{i,t}^s \leftarrow \theta_{i,t}^\intercal x_t$

Theorem 5. *[44] If SupLinUCB is run with* $\alpha = \sqrt{\frac{1}{2} \ln \frac{2Tn}{\delta}}$*, then with probability at least* $1 - \delta$*, the regret of the algorithm is*

$$O(\sqrt{Td \ln^3(nT \ln(T)/\delta)}).$$

2.5 Sleeping MAB

The usual assumption in online learning problems is that all arms are available at all rounds. In many applications, however, this assumption is not appropriate. For example, in network routing problems, some of the routes are unavailable at some point in time due to router or link crashes. In electronic commerce problems, items are out of stock, sellers are not available (due to maintenance or simply going out of business), and buyers do not buy all the time. Even in the setting that originally motivated the multi-armed bandit problems, a gambler playing slot machines, some of the slot machines might be occupied by other players at any given time. This variant of MAB is known as sleeping MAB [75], where the subset of arms available to the algorithm for the selection varies. This variant, sometimes also known as volatile bandits [29] or mortal bandits [36], models many real-world scenarios such as crowdsourcing [83], online advertising [36], and network routing [29, 75]. Kleinberg *et al.* [75] proposed an extension of the UCB1 algorithm for sleeping MAB setting. The algorithm is called *Awake Upper Estimated Reward* (AUER).

Theorem 6. *[75] The AUER algorithm incurs regret of at most*

$$(32 \ln T) \cdot \sum_{j=2}^{n} \sum_{i=1}^{j-1} \left(\frac{1}{\Delta_{i,j}^2} \right) \Delta_{i,i+1}$$

Having introduced a brief overview of MAB; in the next chapter, we introduce 'mechanism design' a game theoretic tool to ensure that the strategic agents cannot manipulate the system/underlying algorithms.

Algorithm 6 SupLinUCB

Initialization: $S \leftarrow \ln T$, $\Psi^s_{i,t} \leftarrow \phi$; $\Psi^s_{est} \leftarrow \phi$ for all $s \in [\ln T]$
for t = 1,2,..., T **do**
  $s \leftarrow 1$ and $\hat{A}_1 \leftarrow [n]$
  $j \leftarrow 1 + (t \bmod n)$
  repeat
    Use *BaseLinUCB-S* with $\{\Psi^s_{i,t}\}_{i \in \mathcal{N}}$ and context vector x_t to calculate the width $w^s_{i,t}$ and upper confidence bound $ucb^s_{i,t} = (\hat{r}^s_{i,t} + w^s_{i,t}), \forall i \in \hat{A}_s$
    if $w^s_{i,t} \leq \frac{1}{\sqrt{T}}, \forall i \in \hat{A}_s$ **then**
      Choose $I_t = \arg\max_{a \in \hat{A}_s} (\hat{r}^s_{i,t} + w^s_{i,t})$
      Keep the same index sets at all levels for I_t:
      $\Psi^{s'}_{I_t,t+1} \leftarrow \Psi^{s'}_{I_t,t}, \forall s' \in [S]$
    else if $w^s_{i,t} \leq 2^{-s}, \forall i \in \hat{A}_s$ **then**
      $\hat{A}_{s+1} \leftarrow \{i \in \hat{A}_s | b_i \cdot (\hat{r}^s_{i,t} + w^s_{i,t}) \geq \max_{a \in \hat{A}_s} b_a \cdot (\hat{r}^s_{a,t} + w^s_{a,t}) - 2^{1-s}\}$
    else
      Select $I_t \in \hat{A}_s$ such that $w^s_{I_t,t} > 2^{-s}$.
      Update the index sets at all levels:
      $$\Psi^{s'}_{I_t,t+1} \leftarrow \begin{cases} \Psi^{s'}_{I_t,t} \cup \{t\} & \text{if } s = s' \\ \Psi^{s'}_{I_t,t} & \text{otherwise} \end{cases}$$
  until I_t is selected

Algorithm 7 AUER

Initialization: $N_{i,0} = 0$, $\overline{\mu}_{i,0} = 1$, $X_{i,0} = 0, \forall i \in [n]$
for $t = 1, 2, \ldots$ **do**
  Observe the set of agents available for selection in round t as $A_t \subseteq [n]$.
  if $\exists j \in A_t, s.t., N_{i,t} = 0$ **then**
    Select $i_t = j$.
  else
    Evaluate $\forall i$, $\overline{\mu}_{i,t} = \hat{\mu}_{i,t} + \sqrt{\frac{2 \ln t}{N_{i,t}}}$.
    Select $i_t = \arg\max_{i \in A_t} \overline{\mu}_{i,t}$, and observe reward $X_{i_t,t}$.
    Update, $N_{i_t,t} \leftarrow N_{i_t,t-1} + 1$; $X_{i_t,1:t} \leftarrow X_{i_t,1:t-1} + X_{i_t,t}$; $\hat{\mu}_{i_t,t} \leftarrow \frac{X_{i_t,1:t}}{N_{i_t,t}}$.

Chapter 3

Mechanism Design: An Overview

Often, the role of the arms is played by strategic agents. These strategic agents holds some private information which can hamper the selection by the planner. As these agents are rational, they can misreport their private information to manipulate the decision making in order to maximize its utility. Mechanism design provides the tools that offers incentives to the agents and aligns their maximization problem to the system's optimization problem.

3.1 Mechanism Design

In this chapter, we provide an overview on *mechanism design*. The content of this section is taken from [67, 91, 54, 55]. The following provides an abstraction to mechanism design problems:

- Let there be n agents. We assume each agent is rational and intelligent.
- χ is a set of outcomes. The agents are required to converge to an outcome from the set χ.
- Each agent i observes a parameter θ_i which determines the preference of agent i over the outcomes χ. The value of θ_i may be unknown to other agents. Here, θ_i is called a private value or type of agent i.
- For agent i, Θ_i denote the set of private values. The set of all type profiles is given by $\Theta = \Theta_1 \times \ldots \times \Theta_n$. A sample type profile is represented as $\theta = (\theta_1, \cdots, \theta_n)$.
- Generally we assume that there is a common prior distribution $\mathbb{P} \in \Delta(\Theta)$. For consistency of beliefs individual belief functions can be derived form the common prior, i.e., $p_i : \Theta_i \to \Delta(\Theta_{-i})$.
- The preference of an agent i over the outcomes is indicated by an utility function $u_i : \chi \times \Theta_i \to \mathbb{R}$. For agent i the value $u_i(x, \theta_i)$ denotes the payoff for agent i when outcome is x and agents' type is θ_i. In general, u_i not only depends on outcome and type of player i but also on the types of other players as well, i.e., $u_i : \chi \times \Theta \to \mathbb{R}$.

- Set of agents $[n]$, outcomes χ, types Θ_i, prior distribution $\mathbb{P}$ and the utility functions u_i are assumed to be common knowledge among all the agents. Note that here the type θ_i of agent i is private information to the agent i.

As the agents' preferences depend on the realization of their types θ, the outcome depends on θ. This leads to social choice function.

Definition 2. ***Social Choice Function*** *– Let $[n]$ be a set of agents with type $\{\Theta_i\}_{i\in[n]}$. Given χ, a social choice function is a mapping $f : \Theta_1 \times \cdots \times \Theta_n \to \chi$, that assigns each possible type profile $(\theta_1, \theta_2, \ldots, \theta_n)$ to an outcome in χ. The outcome corresponding to a type profile is called a social choice or collective choice for that profile.*

Mechanism design is a process to a type of optimization problem where the private information needs to be first elicited, and then the underlying optimization problem is solved. There are two major approaches to elicit the agents' type information truthfully, called direct and indirect mechanisms.

Definition 3. ***Direct Mechanism*** *– Suppose $f : \Theta_1 \times \ldots \times \Theta_n \to \chi$ be a social choice function. A direct mechanism (also known as a direct revelation mechanism) corresponding to f consist o the tuple $(\Theta_1, \ldots, \Theta_n, f(\cdot))$.*

In a direct mechanism, the mechanism seeks the type information from the agents directly, i.e., by asking them to report their true types.

Definition 4. ***Indirect Mechanism*** *– An indirect mechanism (also known as indirect revelation mechanism) consists of a tuple $(Z_1, \ldots, Z_n, f(\cdot))$ where Z_i is a set of possible actions for agent i and $f : Z_1 \times \ldots \times Z_n \to \chi$ is a function that maps each action profile to an outcome.*

Mechanism design involves implementing system-wide solution that will satisfy certain desirable properties. Following are some of the desirable properties which we want a social choice function to satisfy.

Definition 5. ***Dominant Strategy Incentive Compatibility (DSIC)*** *– A social choice function $f : \Theta_1 \times \ldots \times \Theta_n \to \chi$ is said to be dominant strategy incentive compatible if the indirect revelation mechanism $\mathcal{D} = ((Z_i(\Theta_i))_{i\in[n]}, f(\cdot))$ has a weak dominant strategy equilibrium $z^*(\cdot) = (z_1^*(\cdot), \ldots, z_n^*(\cdot))$ in which $s_i^*(\theta_i) = \theta_i$, $\forall\theta_i \in \Theta_i, \forall i \in [n]$ i.e.*

$$u_i(f(\theta_i, z_{-i}(\theta_{-i})), \theta_i) \geq u_i(f(z_i(\theta_i), z_{-i}(\theta_{-i})), \theta_i),$$

$\forall z_i(\theta_i) \in \Theta_i, \forall z_{-i}(\theta_{-i}) \in \Theta_{-i}$. Here, $z_{-i}(\theta_{-i})$ denotes the strategy of all agents other then agent i with type profile θ_{-i}.

Individual rationality is a phenomenon are better off not participating in the mechanism. Formally, individual rationality of a social choice function essentially means that each agent gains a non-negative utility by participating in a mechanism.

Definition 6. ***Ex-Post Individual Rationality*** *– After all agents have announced and an outcome $X \in \chi$ is selected, let $\overline{u}_i(\theta_i)$ be the utility that agent i receives by withdrawing from the mechanism when its type is θ_i. The mechanism is ex-post individual rational if $\forall i \in [n]$*

$$u_i(f(\theta_i, \theta_{-i}), \theta_i) \geq \overline{u}_i(\theta_i) \qquad \forall(\theta_i, \theta_{-i}) \in \Theta$$

3.1.1 Characterizing Truthful Mechanism

Myerson et al. [90] provide result for characterizing DSIC social choice function in quasi-linear environment. The result relies on an important property called monotonicity of an allocation that yields truthfulness. Here, we define the monotonicity followed by the result from [90].

Definition 7. ***Monotone Allocation Rule*** *– Consider two bids b_i and $b_i^{'}$ for agent i such that $b_i \geq b_i^{'}$. Allocation rule $\mathcal{A}$ is called monotone if $\forall i \in [n]$, we have $\mathcal{A}_i(b_i, b_{-i}) \geq \mathcal{A}_i(b_i^{'t}, b_{-i})$ $\forall b_{-i} \in \Theta_{-i}$ $\forall \theta_i \in \Theta_i$.*

Theorem 7. *[90] A mechanism $\mathcal{M} = \mathcal{A}, \mathcal{P}$ is truthful if and only if:*

1. *Allocation rule $\mathcal{A}$ is monotone*
2. *The payment rule $\mathcal{P}$ for any player i satisfies*

$$p_i(b_i, b_{-i}) = p_i^0(b_{-i}) + b_i \cdot \mathcal{A}_i(b_i, b_{-i}) - \int_{-\infty}^{b_i} \mathcal{A}_i(u, b_{-i})du,$$

where $p_i^0(b_{-i})$ does not depend on b_i.

3.2 MAB Mechanism Design

MAB and mechanism design has been studied through the years extensively. Combining both concepts captures many applications, primarily in settings where we require to estimate/learn some quantities in an auction setting. For example, sponsored search auctions (SSA), crowdsourcing, demand response in smart grids, etc. MAB mechanisms [26, 58, 87, 100, 25] model the settings where we have to maximize the social welfare across a time horizon in an uncertain environment (unknown stochastic reward) and strategic agents who want to maximize its utility even if required to misreport its true type. Hence, to maximize social welfare, the mechanism designer needs to learn the stochastic reward while eliciting the agents' private information. Here, we majorly focus on stochastic MAB mechanisms. Usually, while designing MAB mechanisms, we desire it to satisfy some properties.

Definition 8. ***Ex-Post Monotone Allocation Rule*** *– $\mathcal{A}$ is ex-post monotone if for every sequence of reward realizations, for each agent $i \in [n]$, $\forall b_{-i}$, $\forall t$ and two possible bids of i, $b_i^+ \geq b_i$, we have*

$$\mathcal{A}_i(b_i^+, b_{-i}; t) \geq \mathcal{A}_i(b_i, b_{-i}; t)$$

where, $\mathcal{A}_i(b_i, b_{-i}; t) = \sum_{t'=1}^{t} \mathcal{A}_i^{t'}(b_i, b_{-i}; t')$.

Definition 9. ***Ex-Post Incentive Compatible Mechanism*** *– A mechanism $\mathcal{M}$ is ex-post incentive compatible if by misreporting the bid, no agent can gain its total utility more than that it would have obtained by bidding truthfully, i.e., $\forall i, \forall b_{-i}, \forall \theta_i, \forall b_i$,*

$$\sum_{t=1}^{T} u_i(\mathcal{A}_i(\theta_i, b_{-i}; t), \mathcal{P}(\theta_i, b_{-i}; t), \theta_i; t) \geq u_i(\mathcal{A}_i(b_i, b_{-i}; t), \mathcal{P}(b_i, b_{-i}; t), \theta_i; t)$$

There are majorly two types of dominant strategy incentive compatible MAB mechanism: deterministic and randomized. The authors in [21] provide a characterization result for any *deterministic*, dominant strategy incentive compatible MAB mechanism. In [21], they show that any deterministic truthful MAB mechanism has to be exploration-separated. In the exploration-separated mechanism for first τ rounds, the mechanism does exploration, i.e., allocating each agent for $\lfloor \tau/T \rfloor$ rounds (non-adaptively in some deterministic order). Only the exploration rounds the mechanism exploits the agents based on the empirical estimate.

Theorem 8. *[47] For a mechanism $\mathcal{M} = (\mathcal{A}, \mathcal{P})$, let $\mathcal{A}$ be an exploration-separated deterministic allocation rule. Then the regret of mechanism $\mathcal{M}$ running across T rounds is at least $\Omega(T^{2/3})$.*

3.2.1 Generic Transformation for Truthful Mechanisms

Babaioff *et al.* [19] provide a generic transformation that produces a truthful and individually rational mechanism given a monotone allocation rule. This result reduces the problem of designing a truthful mechanism to designing of monotone allocation rule. The generic procedure takes the monotone allocation rule and creates a truthful randomized mechanism in expectation, and attains the same outcome as the original allocation rule with high probability.

The transformation mechanism takes a parameter $\phi \in [0, 1]$ and a monotone allocation rule $\mathcal{A}$ as inputs and outputs a truthful mechanism $\tilde{\mathcal{M}} = (\tilde{\mathcal{A}}, \tilde{\mathcal{P}})$, such that the following properties satisfies:

- It converts the bid vector $\boldsymbol{b}$ to a randomized bid vector $\tilde{\boldsymbol{b}}$ and invokes $\mathcal{A}(\tilde{\boldsymbol{b}})$ once.
- For any bid vector $\boldsymbol{b}$ and fixed parameter μ, $\mathcal{A}(\boldsymbol{b}) = \mathcal{A}(\tilde{\boldsymbol{b}})$ are identical with probability at least $1 - n\phi$.
- $\tilde{M}$ is ex-post individually rational and any agent i pays no more than $\mathcal{A}_i(\tilde{\boldsymbol{b}}) \left(b_i \left(\frac{1}{\phi} - 1 \right) - \frac{\underline{b}}{\phi} \right)$. Here, $\underline{b}$ is the lower bound on the bid i.e. $\forall i, b_i \geq \underline{b}$.

Let s be reward realization of arm i in any round t. For truthful property of monotone allocation rule, it is sufficient to have the following payment rule:

$$\mathcal{P}_i(b_i, b_{-i}; s) = b_i \mathcal{A}_i(b_i, b_{-i}; s) - \int_{-\infty}^{b_i} \mathcal{A}_i(u, b_{-i}; s) du$$

The above integral is challenging to compute. Hence, a sampling procedure was introduced by [19] as *self-resampling procedure*.

Algorithm 8 [19] Self-resampling procedure

Input: Bid $b_i \in [\underline{b}, \infty)$, parameter $\phi \in (0, 1)$.
Output: (x_i, y_i) where $0 \leq x_i \leq y_i \leq b_i$.
with probability $1 - \phi$
    $x_i \leftarrow b_i, y_i \leftarrow b_i$.
else
    Choose $b_i^{'} \in [0, b_i]$ uniformly at random.
    $x_i \leftarrow$ RECURSIVE($b_i^{'}$), $y_i \leftarrow b_i^{'}$
function RECURSIVE(b_i)
    with probability $1 - \phi$
        return b_i
    else
        Choose $b_i^{'} \in [0, b_i]$ uniformly at random.
        return RECURSIVE($b_i^{'}$).

Algorithm 9 [19] Generic Transformation

Input: Bid vector $\boldsymbol{b}$, parameter $\phi \in (0, 1)$, allocation rule $\mathcal{A}$.
Obtain the modified bid for each agent using the self-resampling procedure as $\boldsymbol{x} = (x_1(b1), \ldots, x_n(b_n))$, $\boldsymbol{y} = (y_1(b_1), \ldots y_n(b_n))$.
Do allocation as $\mathcal{A}(\boldsymbol{x})$
The payment from each agent i is $b_i \cdot \mathcal{A}_i(\boldsymbol{x}) - Re_i$, where Re_i is the rebate

$$Re_i \leftarrow \begin{cases} \frac{1}{\phi} \cdot \frac{\mathcal{A}_i(x)}{F_i^{'}(y_i, b_i)} & \text{if } y_i < b_i \\ 0 & \text{otherwise} \end{cases}$$

Definition 10. *[19] **Self-resampling Procedure** – Let I be a nonempty interval in $\mathbb{R}$. A self-resampling procedure with support I ans resampling parameter $\phi \in (0, 1)$ is a randomized algorithm with input $b_i \in I$ and output $x_i(b_i), y_i(b_i) \in I$, that satisfies the following properties $\forall i \in [n]$:*

1. *$x_i(b_i), y_i(b_i)$ are non-decreasing functions of b_i.*
2. *With probability $(1 - \phi)$, $x_i(b_i) = y_i(b_i) = b_i$ and $x_i(b_i) \leq y_i(b_i) < b_i$ with probability ϕ.*

3. *Given $y_i(b_i) = b_i^{'} < b_i$, the conditional distribution of $x_i(b_i)$ is same as conditional distribution of $x_i(b_i^{'})$, i.e.*

$$\mathbb{P}[x_i(b_i) < a_i | y_i(b_i) = b_i^{'}] = \mathbb{P}[x_i(b_i^{'}) < a_i], \quad \forall a_i \leq b_i^{'} < b_i.$$

4. *Consider the function $F(a_i, b_i) = \mathbb{P}[y_i(b_i) < a_i | y_i(b_i) < b_i]$, which is called distribution function of the self-resampling procedure. For each b_i, the function $F(\cdot, b_i)$ must be differentiable and strictly increasing on the interval $I \cap (-\infty, b_i)$.*

We present an example of a self-resampling procedure in Algorithm 8.

Theorem 9. *[19] Let $\mathcal{A}$ be any monotone allocation rule. Then the transformed mechanism $\mathcal{M} = (\tilde{\mathcal{A}}, \tilde{\mathcal{P}})$ satisfy the following property:*

- *$\mathcal{M}$ is truthful, ex-post individually rational.*
- *For n agents and any bid vector b allocations $\tilde{\mathcal{A}}(b)$ and $\mathcal{A}(b)$ are identical with probability at least $1 - n\phi$.*
- *If all the private types are positive then the mechanism $\mathcal{M}$ never pays any advertiser i more than $b_i \cdot \mathcal{A}_i(\boldsymbol{x}) \cdot \left(\frac{1}{\phi} - 1\right)$.*

We have covered the preliminaries for the thesis. In the next chapters we introduce the problems and our approach to tackle the problems in depth.

Chapter 4

Designing Truthful Contextual Multi-Armed Bandits based Sponsored Search Auctions

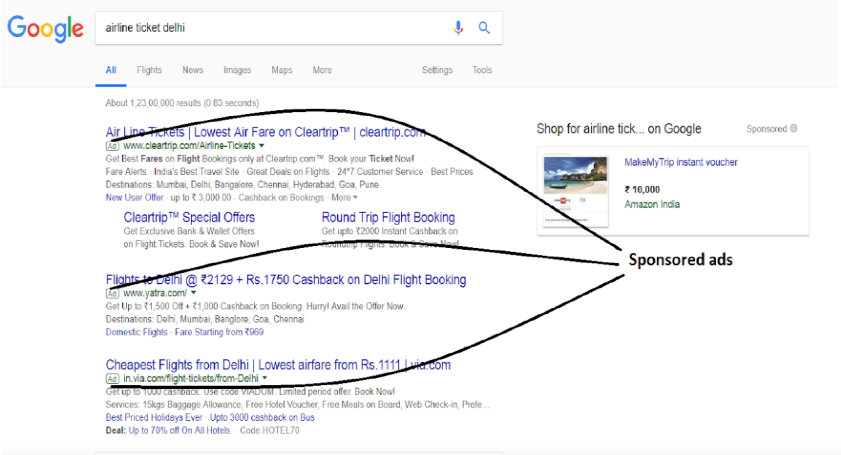


Figure 4.1: Result of sponsored search by Google search engine

We consider the contextual multi-armed bandit problem in the presence of strategic agents in the context of sponsored search auction. In this setting, an advertising platform (*center*) runs a repeated auction to select the best-suited ads relevant to the user's query. The center aspires to select an ad that has a high expected value that depends on the probability of getting a click and the value it derives from a click of the ad. The probability of getting a click (CTR) is unknown to the center and depends on the user's profile (*context*) posting the query. Further, the value derived for a click is the private information to the advertiser and needs to be elicited truthfully.

We first design an elimination-based algorithm *ELinUCB-SB* that is ex-post monotone, which is a sufficient condition for truthfulness. Thus, *ELinUCB-SB* can be naturally extended to ex-post incentive compatible and ex-post individually rational mechanism *M-ELinUCB-SB*. We show via simulations that *M-ELinUCB-SB* outperforms the existing mechanism in this setting. Theoretically, however, the mechanism may incur linear regret in non-frequent instances. We then propose a theoretically stronger mechanism, *M-SupLinUCB-S*. We prove that *M-SupLinUCB-S* is ex-post incentive compatible, ex-post

individually rational, and achieves the regret of $O(n^2\sqrt{dT\log T})$ as against $O(n\sqrt{dT\log T})$ for non-strategic settings, where $O(n)$ is price of truthfulness. We demonstrate the efficacy of our mechanisms via simulation and establish superior performance than the existing literature.

4.1 Introduction

Internet advertising is one of the booming and rapidly increasing industry with revenue volume in billions of dollars [66]. The majority of the revenue generated by search engines like Google, Yahoo, and Bing comes from advertisements displayed on their platform. Typically, for any search query by a user, a search engine/the advertising platform, henceforth a *center*, displays ads along with the relevant results via an auction mechanism known as *sponsored search auction* (SSA). The fundamental difference between traditional advertising and Internet advertising is the payment model. In the former, the advertisers (henceforth *agents*) pay based on the number of impressions, whereas in the latter, the agents pay only if their ad receives a click.

Typically an individual user tends to click some specific ads more often than the other ads, which depends upon the profile of an individual user of the platform. This profile generally serves as a feature vector that provides the context to personalize the ads to increase clicks. Thus, the probability of an ad getting clicked, referred to as *click-through rate* (CTR), also depends on the context. The CTR of an ad is unknown to the center but can be learned over a period of time by displaying the ad repeatedly. Each agent also has a private valuation for its ad, representing its willingness to pay for a click. This valuation needs to be elicited from the agents truthfully.

In the absence of contexts with agents reporting true valuations, the problem can be modeled as a Multi-Armed Bandit (MAB) problem [79, 57] with agents playing the role of the arms. As the agents are strategic, they may misreport their valuations to maximize their utility. To ensure truthfulness, researchers have used Mechanism Design [7, 93]. However, such mechanisms fail to avoid manipulations by the agents when learning is involved. In such cases, the researchers have modeled this problem as a MAB mechanism [21, 47, 56, 70, 100]. The authors designed *ex-post truthful* mechanisms wherein the agents cannot manipulate even when the random clicks are known to them. When the CTRs of the ads depend on the specific context at a particular round, we can model the problem as a *Contextual MAB* (ConMAB) problem [1, 15, 80, 84]. However, a naive implementation of ConMAB is not adequate in the presence of strategic agents.

To the best of our knowledge, contextual information in SSA is considered only in [56]. The authors proposed a novel, theoretically sound, deterministic, exploration-separated mechanism that offers strong game-theoretic properties. However, it faces multiple practical challenges: (i) it incurs the high cost of learning (*regret*), (ii) the center needs to know the number of rounds for which it needs to execute SSA, and (iii) the initial rounds being free, a malicious agent may drop off after free rounds; in some cases, all the rounds could be free.

Contributions

In the presence of strategic agents, random context-arrivals, and stochastic clicks, our goal is to design a non-exploration-separated, ex-post truthful mechanism that (i) learns CTRs efficiently (minimizes regret), (ii) may not need prior knowledge of T, and (iii) does not have free rounds. We leverage popular algorithms *LinUCB* [84] and *SupLinUCB* [44] that perform well in estimating CTRs in the contextual setting to build our randomized mechanisms to avoid manipulations by strategic agents. In particular, our contributions are:

- We adapt *LinUCB* to design an ex-post monotone allocation rule *ELinUCB-S* for a single-slot SSA (Theorem 11). We further optimize *ELinUCB-S* by introducing batch update to propose *ELinUCB-SB* and using resampling procedure by [20], we develop an ex-post truthful mechanism *M-ELinUCB-SB*, which is also ex-post individually rational (Theorem 12). Unlike existing ConMAB mechanism, *M-ELinUCB-SB* does not need to know T.

- For stronger theoretical guarantees, we adapt *SupLinUCB* to design an ex-post monotone allocation rule *SupLinUCB-S* (Theorem 15). We prove that *SupLinUCB-S* has regret $O(n^2\sqrt{dT\log T})$ (Theorem 13) as against $O(n\sqrt{dT\log T})$ for the non-strategic settings; we attribute $O(n)$ as price of truthfulness. Using resampling procedure, we develop *M-SupLinUCB-S* which is ex-post truthful and ex-post individually rational. *M-SupLinUCB-S*, however, needs to know T upfront.

- We study *M-ELinUCB-SB* and *M-SupLinUCB-S* with the existing mechanism *M-Reg*, on simulated data and provide empirical analysis. We establish that *M-ELinUCB-SB* and *M-SupLinUCB-S* significantly outperforms *M-Reg*.

4.2 Preliminaries

4.2.1 Model and Notation

There is a fixed set of agents $[n] = \{1, 2, \ldots, n\}$, where each agent has exactly one ad to display and the center has one slot for allocation. A contextual $n-$armed Multi-Armed Bandit (MAB) mechanism $\mathcal{M}$ proceeds in discrete rounds $t = 1, 2, \ldots, T$. At each round t:

1. $\mathcal{M}$ observes a context $x_t \in [0, 1]^d$ which summarizes the profile of the user arriving at round t.

2. Based on the history, h_t, of allocations, observed clicks, and the context x_t, $\mathcal{M}$ chooses an agent $I_t \in [n]$ to display it's ad.

3. A click r_{I_t} is observed which is 1 if it gets clicked and 0 otherwise.

4. Mechanism $\mathcal{M}$ decides the positive payment $p_{I_t,t}$ to be made by the agent I_t to the center. The payment by any other agent is 0.

5. Update $h_t = h_{t-1} \cup \{x_t, \{I_t\}, \{r_{I_t}\}\}$.

6. The mechanism then improves its arm-selection strategy with new observation $(x_t, \{I_t\}, \{r_{I_t}\})$. No click is observed for the agents that are not selected.

Each agent i is thus characterized by two quantities: (i) private valuation $v_i \in [0, 1]$, which represents the willingness to pay for the click received and is constant throughout the rounds, and (ii) click-through rate (CTR) of its ad $\mu_i(x_t) \in [0, 1]$ which is an unknown parameter and is dependent on the context x_t.

We assume that the bids are constant across the rounds (since v_i's are constant). Typically in SSA, each week, millions of queries occur; hence even for a small duration of a week, keeping the constant valuation and constant bid makes sense. Hence our assumptions are in line with theoretical as well as practical practices. Further, the assumption about the same valuation at each round is standard, as is considered in [47, 56, 70, 71]. Each agent i submits the valuation of getting a click on its ad as bid b_i. Note that the submitted bid at the start of the auction is constant throughout the rounds but may not be the same as the valuation. Hence the truthfulness property needs to be assured by carefully designing the MAB setting mechanism, particularly when contextual information is available. We assume that the CTR of an agent i is linear in d-dimensional context x_t with some unknown coefficient vector θ_i [84]. Thus, the problem reduces to learning the $d-$dimensional vector θ_i for each agent i. The probability of getting a click on the ad of agent i at any given round t is given as:

$$\mathbb{P}[r_{i,t}|x_t] = \mu_i(x_t) = \theta_i^\intercal x_t$$

Thus, the expected valuation of agent i for context x_t at time t is $v_i\mu_i(x_t)$. Let b_{-i} be the bid vector of all the agents other than i and b denote the bid vector of all the agents. The utility of an agent i in round t with history h_t is given as:

$$u_{i,t}(b_i, b_{-i}, x_t; h_t; v_i) = \mathbb{1}\{I_t(b, x_t; h_t) = i\} r_{i,t}(v_i - p_{i,t}(b; h_t))$$

and the utility of center is given as:

$$u_t^c(b, x_t, h_t) = \sum_{i=1}^{n} \{I_t(b, x_t; h_t) = i\} r_{i,t} p_{i,t}$$

When the CTRs are not known, the efficiency of any mechanism is measured by its rate of learning or regret. In this work, we aim to minimize social welfare regret similar to [21, 70]. The social welfare at round t is evaluated as sum of utilities of the agents and the center. Thus, social welfare regret is given as:

$$\mathbb{R}_T(\mathcal{M}) = \sum_{t=1}^{T} [\theta_{i_t^*}^T x_t \cdot v_{i_t^*} - \theta_{I_t}^T x_t \cdot v_{I_t}] \tag{4.1}$$

Here, $i_t^*(x_t)$ denote the highest expected valuation (based on bids) i.e., $i_t^*(x_t) = argmax_k\{v_k \cdot (\theta_k^T x_t)\}$.

In the following section, we define game-theoretic properties relevant to this work.

4.2.2 Game Theoretic Properties

A mechanism $\mathcal{M} = (\mathcal{A}, \mathcal{P})$ (where $\mathcal{A}$ is the allocation rule and $\mathcal{P}$ is the payment rule) is ex-post incentive compatible (formally called EPIC) if and only if $\mathcal{A}$ is ex-post monotone [90, 13]. That is, for all instances of possible click realizations and context-arrivals, by increasing its bid to $b_i^+ > b_i$, agent i should obtain at least same number of clicks at bid b_i if not more. Formally,

Definition 11. ***Ex-post monotonicity:*** *Let $\nu_i(b_i, t)$ denote total number of clicks on the ad of agent i in first t rounds. Then, $\mathcal{A}$ is ex-post monotone if for every possible sequence of context-arrivals and click realizations, for each agent $i \in [n], \forall t, \forall b_{-i}$ and two possible bids of i, $b_i^+ \geq b_i$ we have*

$$\nu_i(b_i^+, t) \geq \nu_i(b_i, t).$$

Definition 12. ***EPIC:*** *A mechanism $\mathcal{M} = (\mathcal{A}, \mathcal{P})$ is said to be* ex-post incentive compatible *$(EPIC)$ if by misreporting the bid, no agent can gain its total utility more than that it would have obtained by bidding truthfully, i.e., $\forall i, \forall b_{-i}, \forall v_i, \forall h_t, \forall b_i$,*

$$\sum_{t=1}^{T} u_{i,t}(v_i, b_{-i}, x_t; h_t; v_i) \geq \sum_{t=1}^{T} u_{i,t}(b_i, b_{-i}, x_t; h_t; v_i).$$

EPIC implies even if an agent has observed all the contexts and all the click realizations, it is in the agent's best interest to report true valuation. Note that, if a mechanism does any randomization, $u_{i,t}(\cdot)$ is replaced by $\mathbb{E}[u_{i,t}(\cdot)]$, where the expectation is taken w.r.t. randomization in the mechanism. Such a mechanism is still truthful for every realization of external randomnesses, such as click realizations and context-arrivals.

One alternative notion of IC which may seem suitable in our model is dynamic IC [24]. We would like the reader to note that in our model, the bids and valuations are constant throughout the rounds and not dependent on any round t. Thus, private information and communication with the mechanism are not dynamic. Hence, a robust game-theoretic property of ex-post IC is more apt in our model.

Definition 13. ***EPIR:*** *A mechanism $\mathcal{M} = (\mathcal{A}, \mathcal{P})$ is said to be* ex-post individually rational *$(EPIR)$ if every agent has a non-negative utility with truthful bidding irrespective of the bids of other agents i.e., $\forall i, \forall x_t, \forall v_i, \forall t, \forall h_t$,*

$$u_{i,t}(v_i, b_{-i}, x_t; h_t; v_i) \geq 0.$$

A randomized context-free MAB mechanism that is ex-post truthful and has regret $O(\sqrt{T})$ was proposed in [20]. Thus, by introducing randomness in the mechanism, it is possible to bypass the impossibility result in [21], which states that any deterministic, truthful MAB mechanism has to be exploration-separated and hence must suffer a regret of $\Omega(T^{2/3})$. The main result of [20] involves designing the black box mechanism using the self-resampling procedure in Algorithm 10, which provides ex-post truthful and IR mechanism if given an ex-post monotone allocation rule (Theorem 10).

Algorithm 10 Non-recursive self-resampling procedure

Input: bid $b = \{b_1, \ldots, b_n\}$, parameter $\delta \in (0, 1)$
Output: modified bid $y = \{y_1, \ldots, y_n\}$, $\eta = (\eta_1, \ldots, \eta_n)$
Independently for each agent $i \in [n]$
    Sample: ϵ_i uniformly at random from $[0, 1]$
    with probability $1 - \delta$
        $\eta_i = 1$
    else
        $\eta_i = \epsilon_i^{1/(1-\delta)}$
Construct the vector of modified bids $y = (y_1, \ldots, y_n)\,,$ where $y_i = \eta_i b_i$

Theorem 10. *(Theorem 4.5, [20]) Let $\mathcal{A}$ be ex-post monotone allocation rule. Applying the transformation in Algorithm 10 to $\mathcal{A}$ with parameter δ, we obtain a mechanism $\mathcal{M}$ such that $\mathcal{M}$ is EPIC and EPIR.*

Above theorem is a special case of Theorem 9, which is abstracted from the notion of truthfulness the allocation rule provides. In our case, we consider ex-post truthfulness of the mechanism.

There does not exist any previous work which provides an ex-post monotone allocation rule in the contextual setting. We address this problem and design two allocation rules, which are ex-post monotone, though they have different properties.

Concerning ConMAB mechanisms for SSA, two works are closely related to our work [56] and [84]. The former considers the strategic agents in ConMAB [56] by proposing a mechanism that we will call *M-Reg*. The mechanism is *exploration-separated* [47, 21] which is deterministic and induces EPIC property. The regret achieved by *M-Reg* is quite high $O(T^{2/3})$ as compared to $O(\sqrt{T})$ regret in the traditional ConMAB problem. The latter introduces $LinUCB$ algorithm, which is particularly of interest to us. Hence we describe it below.

LinUCB

LinUCB [84] is a generic ConMAB algorithm that efficiently learns the CTR of an agent where the CTR model is linear in terms of context and the unknown parameters. *LinUCB* is shown to be efficient in approximating the CTRs of news articles in news recommendation, however, it does not capture strategic manipulations. *LinUCB* captures the contextual information in ConMAB setting by using A_i and c_i for each agent i, where A_i summarizes the information about contexts and c_i corresponding clicks. Motivated by *UCB* [16], *LinUCB* maintains upper confidence bound (UCB) for each agent i as $\mu_{i,t}^{+} \leftarrow \hat{\theta}_i^{\top} x_t + \alpha\sqrt{x_t^{\top} A_i^{-1} x_t}$ where $\hat{\theta}_i \leftarrow A_i^{-1} c_i$ and α is learning parameter. At round t, the algorithm selects the agent I_t with the highest UCB $\mu_{i,t}^{+}$. The statistics for the selected agent I_t is updated as $A_{I_t} \leftarrow A_{I_t} + x_t x_t^{\intercal}$, $c_{I_t} \leftarrow c_{I_t} + r_{I_t} x_t$, where r_{I_t} is the indicator variable of receiving click.

Motivated by *LinUCB*, we build randomized EPIC mechanisms for SSA by developing an ex-post monotone allocation rule, *ELinUCB-SB*, and using the resampling procedure (Algorithm 10) to design a randomized EPIC and EPIR mechanism *M-ElinUCB-SB* [20]. ELinUCB-SB has linear regret. We

present the algorithm as the key ideas to adapt *LinUCB* to design truthful mechanisms are useful and carry forward when we design a more complicated mechanism, *M-SupLinCUB-S*, based on *SupLinUCB* by [44].

4.3 *M-ELinUCB-SB*: Truthful ConMAB Mechanism 1

We first propose a single-slot allocation rule *ELinUCB-S* based on *LinUCB*. We next provide a further optimized algorithm *ELinUCB-SB* that incorporates mini-batch learning, which makes the algorithm efficient both in terms of regret and computation.

4.3.1 *ELinUCB-S*: LinUCB-Based Single-Slot SSA

Algorithm 11 *ELinUCB-S*: LinUCB-based allocation for single-slot SSA

Inputs: n, $\alpha \in \mathbb{R}_+$, bid vector b
Initialization: $S_{act} = [n]$
for all $i \in [n]$ **do**
    $A_i \leftarrow I_d$ (d-dimensional identity matrix)
    $c_i \leftarrow 0_{d\times 1}$ (d-dimensional zero vector)
    $\mu_i^+ \leftarrow b_i; \mu_i^- \leftarrow 0$
for $t = 1, 2, 3, \ldots$ **do**
    Observe context as x_t
    $I_{t'} \leftarrow 1 + (t \mod n)$
    if $I_{t'} \in S_{act}$ **then**
        Allocate agent $I_{t'}$, i.e., $I_t \leftarrow I_{t'}$
        Observe click as $r_{I_t} \in \{0, 1\}$
        $A_{I_t} \leftarrow A_{I_t} + x_t x_t^\intercal$, $c_{I_t} \leftarrow c_{I_t} + r_{I_t} x_t$, $\hat{\theta}_{I_t} \leftarrow A_{I_t}^{-1} c_{I_t}$
        {Update confidence bound}
        if $\mu_{I_t}^- < \mu_{I_t}^+$ **then**
            $(\gamma_{I_t}^-, \gamma_{I_t}^+) \leftarrow b_{I_t}(\hat{\theta}_{I_t} x_t \mp \alpha\sqrt{x_t^\intercal A_{I_t}^{-1} x_t})$
            if $\max(\mu_{I_t}^-, \gamma_{I_t}^-) < \min(\mu_{I_t}^+, \gamma_{I_t}^+)$ **then**
                $(\mu_{I_t}^-, \mu_{I_t}^+) \leftarrow (\max(\mu_{I_t}^-, \gamma_{I_t}^-), \min(\mu_{I_t}^+, \gamma_{I_t}^+))$
            else
                $\left(\mu_{I_t}^-, \mu_{I_t}^+\right) \leftarrow \left(\frac{\mu_{I_t}^- + \mu_{I_t}^+}{2}, \frac{\mu_{I_t}^- + \mu_{I_t}^+}{2}\right)$
    else
        $I_t \leftarrow argmax_i\, b_i \cdot (\hat{\theta}_i^T x_t), \ \ni I_t \in S_{act}$
        Observe click as $r_{I_t} \in \{0, 1\}$
    for all agent $i \in S_{act}$ **do**
        if $\mu_i^+ < \max_{k\in S_{act}} \mu_k^-$ **then**
            Remove i from S_{act}

ELinUCB-S (Algorithm 11) for single-slot allocation maintains a set of active agents S_{act}. At each round, algorithm evaluates whether an agent should be retained in S_{act} or not. Once an agent is evicted

from S_{act}, it can not be added back. For better understanding about the working of the algorithm, we virtually divide the *LinUCB-S* into 4 subroutines: i) Initialization (lines[1-7]) ii) Exploration (lines[11-20]) iii) Exploitation (lines[22-23]) iv) Elimination (lines[24-26]).[1] For each agent i, the algorithm maintains lower confidence bound (LCB) and upper confidence bound (UCB) as μ_i^- and μ_i^+ respectively.

At each round t, the algorithm observes context x_t. It determines the index of agent $I_{t'}$ whose turn is to display the ad based on round robin order, as stated in line[9]. The algorithm then checks if $I_{t'} \in S_{act}$. If it evaluates to true the algorithm runs Exploration subroutine else Exploitation. In Exploration subroutine the algorithm allocates the slot to $I_{t'}$, observes click $r_{I_t'}$ and updates its parameters. The confidence bounds are updated if and only if the size of confidence interval decreases (line[18]). In Exploitation subroutine, the agent with the maximum estimated expected valuation among the agents in S_{act} is allocated the slot and observes click r_{I_t}. It is important to note that no parameter is updated during Exploitation subroutine which is crucial for the ex-post monotonicity property. At the end, Elimination subroutine is executed which removes the agents $j \in S_{act}$ from S_{act} if UCB of agent j is less than LCB of any other agent in S_{act}.

The intuition behind the algorithm is after sufficient exploration the confidence interval becomes sufficiently small, hence the agents close to optimal continues to remain in S_{act} and sub-optimal agents are eliminated.

4.3.2 Regret Analysis of *ELinUCB-SB*

Although the algorithm *ELinUCB-S* seems promising, the dynamic and varying nature of contexts and its arrival order may lead to the elimination of an optimal agent. Hence, it may continue to allocate sub-optimal agents in subsequent rounds leading to high regret on specific instances, which is evident from our simulation of the algorithm (Fig.4.2c). The updates in μ_i^+, μ_i^- depend upon the context so that μ_i^+ is non-increasing and μ_i^- is non-decreasing, as stated and proved in Claim 2. These updates being irreversible, needs to be carefully handled to optimize regret. To counter this problem, we design *ELinUCB-SB* (Algorithm 12) in which we have introduced a subtle yet important use of mini-batch. The algorithm *ELinUCB-SB* allocates an agent for bs number of rounds instead of one round. It follows similar rules for allocating agents and maintaining the active set S_{act}. It updates the bounds of agents by taking the average over the contexts arrived in bs rounds. Updating the bounds over the average of context after batch allocation handles the variance in contexts and arrivals, thus reducing regret significantly.

It can be shown that eventually, *ELinUCB-SB* will eliminate all but one arm. The remaining arm will be the dominant arm in most of the contexts. However, we can construct examples where this arm is not the best for at least one context, which has a non-zero probability, leading to $O(T)$ regret. Even though *ELinUCB-SB* incurs linear regret theoretically, it performs well in experiments and has interesting monotonicity properties; We next prove the ex-post monotonicity of *ELinUCB-SB*.

[1]Note that this is virtual division and proposed algorithms is not actually exploration-separated where initial rounds are only exploration as well as free and then exploitation where no update happens.

Algorithm 12 *ELinUCB-SB*: LinUCB-based batch allocation rule for single-slot SSA

Inputs: $n, T, \alpha \in \mathbb{R}_+$, bid vector b, batch size bs
Initialization: $S_{act} = [n]$, $x^{'} \leftarrow 0_{d\times 1}, T^{'} = \lfloor \frac{T}{bs} \rfloor$
for all $i \in [n]$ **do**
    $A_i \leftarrow I_d$ (d-dimensional identity matrix)
    $c_i \leftarrow 0_{d\times 1}$ (d-dimensional zero vector)
    $\mu_i^+ \leftarrow b_i; \mu_i^- \leftarrow 0$
for $t^{'} = 1, 2, 3, \ldots, T^{'}$ **do**
    $I_{t'} \leftarrow 1 + (t^{'} - 1) \mod n$
    if $I_{t'} \in S_{act}$ **then**
        for $t = (t^{'} - 1)bs, \ldots, (t^{'} \cdot bs - 1)$ **do**
            Observe context as x_t
            $I_t \leftarrow I_{t'}$,
            $x^{'} \leftarrow ((t-1)x^{'} + x_t)/t$ (averaging over contexts)
            Observe click as $r_{I_t} \in \{0, 1\}$
            $A_{I_t} \leftarrow A_{I_t} + x_t x_t^{\intercal}, c_{I_t} \leftarrow c_{I_t} + r_{I_t} x_t, \hat{\theta}_{I_t} \leftarrow A_{I_t}^{-1} c_{I_t}$
        if $\mu_{I_t}^- < \mu_{I_t}^+$ **then**
            $(\gamma_{I_t}^-, \gamma_{I_t}^+) \leftarrow b_{I_t}(\hat{\theta}_{I_t}^{\intercal} x^{'} \mp \alpha\sqrt{(x^{'})^{\intercal} A_{I_t}^{-1} x^{'}})$
            if $\max(\mu_{I_t}^-, \gamma_{I_t}^-) < \min(\mu_{I_t}^+, \gamma_{I_t}^+)$ **then**
                $(\mu_{I_t}^-, \mu_{I_t}^+) \leftarrow (\max(\mu_{I_t}^-, \gamma_{I_t}^-), \min(\mu_{I_t}^+, \gamma_{I_t}^+))$
            else
                $\left(\mu_{I_t}^-, \mu_{I_t}^+\right) \leftarrow \left(\frac{\mu_{I_t}^- + \mu_{I_t}^+}{2}, \frac{\mu_{I_t}^- + \mu_{I_t}^+}{2}\right)$
    else
        for $t = (t^{'} - 1)bs, \ldots, (t^{'} \cdot bs - 1)$ **do**
            Observe x_t
            $I_t \leftarrow argmax_i\, b_i \cdot (\hat{\theta}_i^T x_t), \ \ni I_t \in S_{act}$
            Observe click as $r_{I_t} \in \{0, 1\}$
    for all agent $i \in S_{act}$ **do**
        if $\mu_i^+ < \max_{k \in S_{act}} \mu_k^-$ **then**
            Remove i from S_{act}

4.3.3 Monotonicity of *ELinUCB-S*

For a fixed sequence of context-arrivals $\{x_t\}_t$, and click realization ρ, let $S_{act}(b, t)$ be the set of active agents in the beginning of round t when agents bid $b = (b_i, b_{-i})$. For each agent i, let $\mu_i^-(b, t)$ and $\mu_i^+(b, t)$ be the values of μ_i^- and μ_i^+ in the round t and similarly when agents bid b'. We prove ex-post monotonicity with the following claims.

Claim 1. *For fixed context-arrivals $\{x_t\}_t$ and click realization ρ, let two bid vectors be* b *and b', then for each round t, if $i \in S_{act}(b, t) \cap S_{act}(b^{'}, t)$, then:*

$$\mu_i^-(b, t)/b_i = \mu_i^-(b', t)/b'_i \text{ and } \mu_i^+(b, t)/b_i = \mu_i^+(b', t)/b'_i$$

Proof. μ_i^+ and μ_i^- are updated only in Exploration subroutine which is based on round-robin order and hence does not depend on bid. Thus, the claim follows from the fact that contexts and click realizations are fixed. □

Claim 2. *For a fixed bid vector b, and each agent $i : \mu_i^- \leq \mu_i^+$, then for all $(t-1, t)$ consecutive rounds μ_i^- is non-decreasing and μ_i^+ is non-increasing.*

Proof. From lines[15-20] of the Algorithm 11, we have: $\mu_i^-(b, t-1) \leq \mu_i^-(b, t) \leq \mu_i^+(b, t) \leq \mu_i^+(b, t-1)$. Hence the claim holds. □

Claim 3. *For any two bid vectors b^+ and b, where $b_i^+ \geq b_i$, $b_j^+ = b_j \; \forall j \neq i$ and $i \in S_{act}(b^+, \tau) \cap S_{act}(b, \tau)$, then $\forall \tau \in \{1, 2, \ldots, T\}$, $S_{act}(b^+, \tau) \subseteq S_{act}(b, \tau)$ holds.*

Proof. The condition $i \in S_{act}(b^+, \tau) \cap S_{act}(b, \tau)$ implies that if $i \in S_{act}(b^+, \tau)$, then $i \in S_{act}(b, \tau)$, hence satisfying the claim for i. For $j \neq i$, we will use induction on t. The claim trivially holds for $t = 1$. Let, $t \leq \tau$ be the last round such that $S_{act}(b^+, t) = S_{act}(b, t)$ and $S_{act}(b^+, t+1) \neq S_{act}(b, t+1)$. In this case, we prove that: $\forall j \neq i, j \in S_{act}(b^+, t+1) \implies j \in S_{act}(b, t+1)$.
Since, $j \neq i$, $\mu_j^+(b^+, t+1) = \mu_j^+(b, t+1)$, $\mu_z^-(b^+, t+1) = \mu_z^-(b, t+1) \; \forall z \neq i$, and $\mu_i^-(b^+, t+1) \geq \mu_i^-(b, t+1)$ from Claim 1. Thus,

$$\begin{aligned} \mu_j^+(b^+, t+1) &> \max_{z \in S_{act}(b^+, t)} \mu_z^-(b^+, t+1) \\ &\geq \max_{z \in S_{act}(b^+, t)} \mu_z^-(b, t+1) \\ \implies \mu_j^+(b, t+1) &> \max_{z \in S_{act}(b, t)} \mu_z^-(b, t+1) \end{aligned}$$

(Since $S_{act}(b, t) = S_{act}(b^+, t)$). Hence, $j \in S_{act}(b, t+1)$. From, induction hypothesis, $\forall t'$ s.t. $\tau > t' \geq t$, $S_{act}(b^+, t') \subseteq S_{act}(b, t')$. We will now prove that $S_{act}(b^+, t'+1) \subseteq S_{act}(b, t'+1)$.

Consider any $j \in S_{act}(b^+, t') \cap S_{act}(b, t')$: we will prove that if $j \in S_{act}(b^+, t'+1)$, then $j \in S_{act}(b, t'+1)$.

Since $j \in S_{act}(b^+, t') \cap S_{act}(b, t')$, $\mu_j^+(b^+, t'+1) = \mu_j^+(b, t'+1)$. Also, $\forall z \in S_{act}(b, t') \cap S_{act}(b^+, t')$ and $z \neq i$, $\mu_z^-(b^+, t'+1) = \mu_z^-(b, t'+1)$. Further, $\forall z$ such that $z \in S_{act}(b, t')$ but $z \notin S_{act}(b^+, t')$, $\exists l \in S_{act}(b^+, t')$ such that $\mu_z^+(b^+, t'+1) < \mu_l^-(b^+, t'+1) \implies \mu_z^-(b^+, t'+1) < \mu_l^-(b^+, t'+1)$. Thus, $\max_{z \in S_{act}(b, t'+1)} \mu_z^-(b^+, t'+1) \leq \max_{z \in S_{act}(b^+, t'+1)} \mu_z^-(b^+, t'+1)$. Thus, $j \in S_{act}(b^+, t'+1)$

implies

$$\begin{aligned}
&\mu_j^+(b^+.t'+1) \geq \max_{z \in S_{act}(b^+,t')} \mu_z^-(b^+, t'+1) \\
\implies &\mu_j^+(b, t'+1) \geq \max_{z \in S_{act}(b,t')} \mu_z^-(b^+, t'+1) \\
\implies &\mu_j^+(b, t'+1) \geq \max_{z \in S_{act}(b,t')} \mu_z^-(b, t'+1)
\end{aligned}$$

□

Claim 4. *For fixed context-arrivals, fixed click realizations, and fixed bids of the agents except i, that is, for a fixed b_{-i}, if $i \in S_{act}(t, b)$, then $i \in S_{act}(t, b^+)$ where $b = (b_i, b_{-i})$ and $b^+ = (b_i^+, b_{-i})$; $b_i^+ > b_i$.*

Proof. Let $\tau^* \geq 1$ be the last round for i s.t. it is in active set with both bids. From Claim 1, $\mu_i^+(b^+, \tau^*) = \frac{b_i^+}{b_i}\mu_i^+(b, \tau^*) > \mu_i^+(b, \tau^*)$ as $i \in S_{act}(b, \tau^*) \cap S_{act}(b^+, \tau^*)$. As the context-arrivals, click realizations and bids of the remaining agents are fixed, if agent i becomes inactive with bid b_i^+ then

$$\begin{aligned}
&\mu_i^+(b^+, \tau^*) < \max_{k \in S_{act}(b^+, \tau^*-1)} \mu_k^-(b^+, \tau^*) \\
\implies &\mu_i^+(b, \tau^*) < \max_{k \in S_{act}(b^+, \tau^*-1)} \mu_k^-(b^+, \tau^*) \\
\implies &\mu_i^+(b, \tau^*) < \max_{k \in S_{act}(b, \tau^*-1)} \mu_k^-(b, \tau^*).
\end{aligned}$$

The last line follows from Claim 3. Thus, i is also inactive with bid b_i. □

Theorem 11. *The allocation rule induced by ELinUCB-S (Algorithm 11) is ex-post monotone.*

Proof. For a fixed context-arrivals $\{x_t\}_t$, click realization ρ, bids of agents except i, i.e., b_{-i} and two possible bids $b_i^+ > b_i$, let τ and τ^+ be the last round for i s.t. i is in active set with bids b_i and b_i^+ respectively. From Claim 4, $\tau^+ \geq \tau$. Thus, i will receive more number of rounds with bid b_i^+ as compared with bid b_i. □

Proposition 1. *The allocation rule induced by ELinUCB-SB (Algorithm 12) is ex-post monotone.*

Proof. The difference between Algorithm 11 and Algorithm 12 is the introduction of batch allocation. In Algorithm 12 the unit of one round is equivalent to bs rounds in Algorithm 11. Hence, by replacing variable t with $t^{'}$ where $t \in \{1, 2, \ldots, T\}$ and $t^{'} \in \{1, 2, \ldots, \lfloor \frac{T}{bs} \rfloor\}$ will satisfy all the claims (Claim 1-4). Thus, *ELinUCB-SB* (Algorithm 12) is still ex-post monotone. □

Algorithm 13 *M-ELinUCB-SB*: LinUCB-based ex-post truthful mechanism

1: **Input:** bid vector b, resampling parameter δ
2: Run self-resampling procedure on bid vector b, obtain modified bid vector $y = (y_1, \ldots, y_n)$, $\eta = (\eta_1, \ldots, \eta_n)$
3: Allocate according to $\mathcal{A}(y, t)$
4: For each agent i, assign payment $p_{i,t} = b_i \cdot \mathcal{A}_{i,t}(y, t) \cdot \begin{cases} 1 & \text{if } \eta_i = 1 \\ 1 - \frac{1}{\delta} & \text{if } \eta_i < 1 \end{cases}$

M-ELinUCB-SB

We now propose the following mechanism *M-ELinUCB-SB* for the single-slot SSA. A mechanism is defined as $\mathcal{M} = (\mathcal{A}, \mathcal{P})$. The outline of both mechanisms is defined in Mechanism 13. For both the mechanisms, we apply the resampling procedure [20] on the bids and the allocation in both cases are based on the modified bids, where δ is resampling parameter. For *M-ELinUCB-SB*, $\mathcal{A}$ is given by *ELinUCB-SB*. The payment $\mathcal{P}$ at round t, corresponding context x_t, $\forall i \in [n]$ is given by $p_{i,t} = b_i \cdot (\mathbb{1}\{I_t = i\})$ if $\eta_i = 1$ and $p_{i,t} = b_i \cdot (\mathbb{1}\{I_t = i\}) \cdot (1 - \frac{1}{\delta})$ if $\eta_i < 1$.

Theorem 12. M-ELinUCB-SB *is ex-post incentive compatible (EPIC) and ex-post individually rational (EPIR) mechanism.*

Proof. The result follows from Theorem 10 and by ex-post monotonicity of $\mathcal{A}$ defined in Algorithm 12. □

4.4 *M-SupLinUCB-S*: Truthful ConMAB Mechanism 2

As the mechanism with allocation rule in Algorithm 12 can incur linear regret, we next propose M-SupLinUCB-S that achieves sub-linear regret. First, we explain how we adapt SupLinUCB [44] for SSA to derive an ex-post monotone allocation algorithm SupLinUCB-S in the next subsection. In Section 4.4.2, we prove the regret bound on SupLinUCB-S. Then we prove the monotonicity of it.

4.4.1 *SupLinUCB-S*

First let us emphasize the major differences between SupLinUCB ([44]) and SupLinUCB-S (Algorithm 14). (i) SupLinUCB considers a common θ to be learned across all the agents. For each round t the contexts are different for each agent whereas in our setting independent θ_i are to be learned for each agent i while context being the same across each agent. (ii) SupLinUCB is non-strategic and hence does not require ex-post monotonicity.

Note that SupLinUCB-S uses BaseLinUCB-S as subroutine Line [6] at each round (multiple times in a round based on s). SupLinUCB maintains 2 sets of rounds each of cardinality S. First set maintains S sets of rounds for each agent i, $\Psi^1_{i,t}, \ldots, \Psi^S_{i,t}$, where $\Psi^s_{i,t} \subseteq \{1, \ldots, T\}$ contains the rounds for which selection was done at stage s in the original SupLinUCB algorithm. Second set also maintains S arises

due to the adaptation of SupLinUCB to the allocation rule truthful and for each stage is denoted by Ψ^s_{est}, where $\Psi^s_{est} \subseteq \{1, \ldots, T\}$.

SupLinUCB-S working: SupLinUCB-S has similar framework to ELinUCB in order to adapt SupLinUCB to make it truthful. The sets $\Psi^s_{i,\cdot}$ are arranged such that width $w^s_{i,\cdot}$ of each agent i is at most 2^{-s}. $\hat{A}_s$ denote the set of active agents available for selection at stage s. At each round t, stage s is initialized with 1 and $\hat{A}_s$ with all agents $[n]$. At each round t, SupLinUCB-S chooses provisional agent j (Line [4]) to be selected based on round robin order. Following process is repeated until an agent is selected:

- Line [6]: Evaluate $ucb^s_{i,t}$ for each agent i at stage s using $BaseLinUCB - S$
- Lines [7-9]: If the width wi, t^s of provisional agent j is more than 2^{-s} and j belongs to the active set $\hat{A}_s$, then j is selected and exploration is done as the width of the upper confidence bound is big
- Lines [10-12]: This checks if for all the agents in the active set $\hat{A}_s$ the width of the upper confidence bound is sufficiently small (less than $\frac{1}{\sqrt{T}}$, i.e., if it is sure that the expected reward of this alternative is close to the optimal expected reward. In this case exploitation takes place and agent with highest empirical valuation is selected $I_t = argmax_{i \in \hat{A}_s} b_i \cdot (\hat{r}^s_{i,t} + w^s_{i,t})$
- Lines [13-15]: This checks if for all the agents in the active set $\hat{A}_s$ the width of the upper confidence bound is smaller than the permissible limit for that stage, i.e., 2^{-s}. In this case elimination takes place where the only those agents which are sufficiently close to the optimal agent are passed to the next stage $s + 1$, $\hat{A}_{s+1}$
- Lines [16-18]: This case does not exist in SupLinUCB algorithm by [44]. This case arises with our adaptation to design a truthful allocation rule. Note that in Lines[10-12], the algorithm allows exploration for only the provisional agent j (based on round robin order, crucial to ensure truthfulness). As there can be other agent k such that $w^s_{k,t} > 2^{-s}$ where $k! = j$, we allow exploitation in this case without any update in statistics for learning.

We next provide the regret analysis of *SupLinUCB-S*, highlighting the steps differing from that of *SupLinUCB*.

4.4.2 Regret Analysis of *SupLinUCB-S*

For convenience, let $s_{i,t} = \sqrt{x_t^\intercal A^{-1}_{i,t} x_t}$, $ucb^s_{i,t} = (\hat{r}^s_{i,t} + w^s_{i,t})$ and $0 < b_i <= 1, \forall i \in [n]$. For all round t, stage s and given context x_t, $i^*_t(x_t) = argmax_{i \in \hat{A}_s} b_i \cdot \mathbb{E}[r_{i,t}|x_t]$. The regret analysis is along the similar lines with [44] with changes deemed necessary to incorporate in our setting. In Lemmas 3, 4, and 5, we need to work for each agent as we have different θ_is for different agents. In our setting, rewards of the arms also have bid component, thus we need the following lemma.

Algorithm 14 *SupLinUCB-S:(Adapted from SupLinUCB by [44] to satisfy monotonicity property)*

Initialization: $S \leftarrow \ln T$, $\Psi^s_{i,t} \leftarrow \phi$; $\Psi^s_{est} \leftarrow \phi$ for all $s \in [\ln T]$
for t = 1,2,..., T **do**
  $s \leftarrow 1$ and $\hat{A}_1 \leftarrow [n]$
  $j \leftarrow 1 + (t \bmod n)$
  repeat
    Use *BaseLinUCB-S* with $\{\Psi^s_{i,t}\}_{i\in[n]}$ and context vector x_t to calculate the width $w^s_{i,t}$ and upper confidence bound $ucb^s_{i,t} = (\hat{r}^s_{i,t} + w^s_{i,t}), \forall i \in \hat{A}_s$
    if $j \in \hat{A}_s$ and $w^s_{j,t} > 2^{-s}$ **then**
      Select $I_t = j$
      Update the index sets at all levels:
      $\Psi^{s'}_{I_t,t+1} \leftarrow \begin{cases} \Psi^{s'}_{I_t,t} \cup \{t\} & \text{if } s = s' \\ \Psi^{s'}_{I_t,t} & \text{otherwise} \end{cases}$
    else if $w^s_{i,t} \leq \frac{1}{\sqrt{T}}, \forall i \in \hat{A}_s$ **then**
      Select $I_t = argmax_{i\in\hat{A}_s} b_i \cdot (\hat{r}^s_{i,t} + w^s_{i,t})$
      Update index sets at all levels for I_t:
      $\Psi^{s'}_{I_t,t+1} \leftarrow \Psi^{s'}_{I_t,t}, \forall s' \in [S]$
    else if $w^s_{i,t} \leq 2^{-s}, \forall i \in \hat{A}_s$ **then**
      $\hat{A}_{s+1} \leftarrow \{i \in \hat{A}_s | b_i \cdot (\hat{r}^s_{i,t} + w^s_{i,t}) \geq \max_{a\in\hat{A}_s} b_a \cdot (\hat{r}^s_{a,t} + w^s_{a,t}) - 2^{1-s}\}$
      $s \leftarrow s + 1$
    else
      Select $I_t = argmax_{i\in\hat{A}_s} b_i \cdot (\hat{r}^s_{i,t} + w^s_{i,t})$
      Update index sets at level s, Ψ^s_{est}
  until I_t is selected

Lemma 1. *With probability* $1 - \kappa S$*, for any* $t \in [T]$ *and any* $s \in [S]$*, the following hold:*

1. $b_i \cdot ucb^s_{i,t} - 2 \cdot w^s_{i,t} \leq b_i \cdot \mathbb{E}[r_{i,t}] \leq b_i \cdot ucb^s_{i,t}, \forall i$
2. $i^*_t(x_t) \in \hat{A}_s$
3. $b_{i^*_t(x_t)} \cdot \mathbb{E}[r_{i^*(x_t),t}] - b_i \cdot \mathbb{E}[r_{i,t}] \leq 2^{3-s}$

Proof. From Lemma 15 of [15], we have, $ucb^s_{i,t} - 2 \cdot w^s_{i,t} \leq \mathbb{E}[r_{i,t}] \leq ucb^s_{i,t}$ for all i. As $b_i > 0$, after multiplying with b_i inequality still holds, i.e., $b_i \cdot ucb^s_{i,t} - 2 \cdot b_i w^s_{i,t} \leq b_i \cdot \mathbb{E}[r_{i,t}] \leq b_i \cdot ucb^s_{i,t}$. From our assumption $b_i \leq 1$, hence the first part holds.

The lemma trivially holds for $s = 1$. For $s > 1$, $\hat{A}_s \subseteq \hat{A}_{s-1}$ and from the algorithm it is clear that $w^s_{i,t} \leq 2^{-(s-1)}$ and $w^s_{i^*_t(x_t)} \leq 2^{-(s-1)}$. From part 1 of the lemma and using the above fact, for any $j \in \hat{A}_s$ we have $b_{i^*_t(x_t)} ucb^{(s-1)}_{i^*_t(x_t),t} \geq b_{i^*_t(x_t)} \mathbb{E}[r_{i^*_t(x_t),t}]$ and $b_j \mathbb{E}[r_{j,t}] \geq b_j ucb^s_{j,t} - 2 \cdot 2^{-(s-1)}$. From

Algorithm 15 *BaseLinUCB-S: (Adapted from BaseLinUCB by [44])*

1: **Inputs:** $\alpha \in \mathbb{R}_+$, $\Psi_{i,t} \subseteq \{1, 2, \ldots, t-1\}$
2: $A_{i,t} \leftarrow I_d + \sum_{\tau \in \Psi_{i,t}} x_\tau^\intercal x_\tau$
3: $c_{i,t} \leftarrow \sum_{\tau \in \Psi_{i,t}} r_{i,\tau} x_\tau$
4: $\theta_{i,t} \leftarrow A_{i,t}^{-1} c_{i,t}$
5: Observe context vector as $x_t \in [0,1]^d$
6: **for** $i \in [n]$ **do**
7: $\quad w_{i,t}^s \leftarrow \alpha \sqrt{x_t^\intercal A_{i,t}^{-1} x_t}$
8: $\quad \hat{r}_{i,t}^s \leftarrow \theta_{i,t}^\intercal x_t$

definition, $b_{i_t^*(x_t)} \mathbb{E}[r_{i_t^*(x_t),t}] \geq b_j \mathbb{E}[r_{j,t}]$. Using this and above inequalities, agent $i_t^*(x_t)$ will belong to $\hat{A}_s, \forall s$ (i.e., will never be eliminated for context x_t) due to the rule defined in Line [14], Algorithm 14. Hence part 2 of the lemma is proved.

From Line [14] Algorithm 14, $b_i ucb_{i,t}^s \geq b_{i_t^*(x_t)} ucb_{i_t^*(x_t),t}^s - 2 \cdot 2^{-(s-1)}$ and part 1 of the lemma the proof of part 3 follows. □

Lemma 2. *(Lemma 6, [44]) For all $s \in [S]$ and $i \in [n]$,*

$$|\psi_{i,T+1}^s| \leq 5 \cdot 2^s (1+\alpha^2) \sqrt{d |\psi_{i,t+1}^s|}$$

Lemma 3. *(Lemma 2, [44]) For each $s \in [S]$ and $i \in [n]$, suppose $\psi_{i,t+1}^s = \psi_{i,t}^s \cup \{t\}$. Then, eigenvalues of $A_{i,t}$ can be arranged so that $\lambda_{i,t}^j \leq \lambda_{i,t+1}^j$, for all j and*

$$s_{i,t}^2 \leq 10 \sum_{j=1}^{d} \frac{\lambda_{i,t+1}^j - \lambda_{i,t}^j}{\lambda_{i,t}^j}$$

Lemma 4. *(Lemma 3, [44]) Using notation in* BaseLinUCB-S *and assuming $|\psi_{i,T+1}^s| \geq 2$, we have*

$$\sum_{t \in \psi_{i,T+1}^s} s_{i,t} \leq 5 \sqrt{d |\psi_{i,T+1}^s| \ln |\psi_{i,T+1}^s|}$$

Lemma 5. *(Lemma 4, [44]). For each $s \in [S]$, each $t \in [T]$, and any fixed sequence of feature vectors x_t, with $t \in \psi_{I_t,t}^s$, the corresponding rewards $r_{I_t,t}$ are independent random variables such that $\mathbb{E}[r_{I_t,t}] = \theta_i^\intercal x_t$.*

We now provide the main theorem.

Theorem 13. *With $\alpha = \sqrt{\frac{1}{2} \ln \frac{2nT}{\kappa}}$, SupLinUCB-S has regret $O(n^2 \sqrt{dT \ln T})$ with probability at least $1 - \kappa$.*

Proof. The proof is similar to the proof of Theorem 6 of [15] but requires additional terms to consolidate the difference between the algorithms, problem setting, and regret definition. We have restricted the learning during round-robin ordering only Lines [7-9], due to which we have an additional decision rule for agent selection as in Line [17]. Hence the main challenge is to bound the number of rounds agent selection, which is done using this decision rule. Let ψ_0 and ψ^s_{est} be the set of round which agent got in Lines [10-12] and in Lines [16-17] respectively. Further, $\psi^s_{T+1} = \bigcup_i \psi^s_{i,T+1}$.

Claim 5. *At each stage s,* $|\psi^s_{est}| <= (n-1) \cdot |\psi^s_{T+1}|$.

For any stage s, let us assume for each of the n consecutive rounds, selection of agent is done in Lines [16-17]. But note that selection of agent in this decision block is done if and only if there exist an agent k such that $j \neq k$ (where j is designated agent for the round) and $w^s_{k,\cdot} > 2^{-s}$. But after n consecutive rounds each agent has got its designated round once (Line [4]). Hence, if for some agent k, $w^s_{k,\cdot} > 2^{-s}$, then this agent k should be selected on its designated round. Hence our assumption of selection of agent in Lines [16-17] for n consecutive round is wrong. From above reasoning, it is clear that at least for one round out of n rounds, one of the agent must be selected at its designated round. Hence, at most for $n-1$ rounds agent is selected in Line [16-17] out of n rounds, until condition in Line [10] is achieved. Thus, we can say that at each stage s, $|\psi^s_{est}| <= (n-1) \cdot |\psi^s_{T+1}|$. As $2^{-S} \leq 1/\sqrt{T}$, we have $\{1, \ldots, T\} = \psi_0 \cup \bigcup_s \psi^s_{T+1} \bigcup_s \cup \psi^s_{est}$. Using the claim and the lemmas, we get regret

$$
\begin{aligned}
\mathbb{R}_T &= \sum_{t=1}^{T} [b_{i^*_t(x_t)} \mathbb{E}[r_{i^*_t(x_t),t}] - b_{I_t} \mathbb{E}[r_{I_t,t}]] \\
&= \sum_{t \in \psi_0} [b_{i^*_t(x_t)} \mathbb{E}[r_{i^*_t(x_t),t}] - b_{I_t} \mathbb{E}[r_{I_t,t}]] \\
&\quad + \sum_{s=1}^{S} \Bigg[\sum_{t \in \psi^s_{T+1}} [b_{i^*_t(x_t)} \mathbb{E}[r_{i^*_t(x_t),t}] - b_{I_t} \mathbb{E}[r_{I_t,t}]] \\
&\quad + \sum_{t \in \psi^s_{est}} [b_{i^*_t(x_t)} \mathbb{E}[r_{i^*_t(x_t),t}] - b_{I_t} \mathbb{E}[r_{I_t,t}]] \Bigg] \\
&\leq \frac{2}{\sqrt{T}} |\psi_0| + \sum_{s=1}^{S} n \cdot \sum_{t \in \psi^s_{T+1}} [b_{i^*_t(x_t)} \mathbb{E}[r_{i^*_t(x_t),t}] - b_{I_t} \mathbb{E}[r_{I_t,t}]]
\end{aligned}
$$

$$
\begin{aligned}
=&\frac{2}{\sqrt{T}}|\psi_0| + \sum_{s=1}^{S} n \sum_{i}^{|[n]|} \sum_{t \in \psi^s_{i,T+1}} [b_{i^*_t(x_t)} \mathbb{E}[r_{i^*_t(x_t),t}] - b_{I_t} \mathbb{E}[r_{I_t,t}]] \\
\leq&\frac{2}{\sqrt{T}}|\psi_0| + n \sum_{i}^{|[n]|} \sum_{s=1}^{S} 8 \cdot 2^{-s} \cdot |\psi^s_{i,T+1}| \\
\leq&\frac{2}{\sqrt{T}}|\psi_0| + n \sum_{i}^{|[n]|} \sum_{s=1}^{S} 40 \cdot (1 + \ln(2Tn/\kappa)) \cdot \sqrt{d|\psi^s_{i,T+1}|} \\
\leq&\frac{2}{\sqrt{T}}|\psi_0| + n \sum_{i}^{|[n]|} 40 \cdot (1 + \ln(2Tn/\kappa) \cdot \sqrt{STd} \\
\leq&2\sqrt{T} + 40n^2 \cdot (1 + \ln(2Tn/\kappa) \cdot \sqrt{STd}
\end{aligned}
$$

□

Monotonicity of SupLinUCB-S

Theorem 14. *The allocation rule induced by SupLinUCB-S (Algorithm 14) is ex-post monotone.*

Proof. The allocation rules Algorithm 11, and Algorithm 14 are similar in the way it learns and eliminates agents. Both algorithms learn if and only if a designated agent is selected based on a round-robin order, and the elimination is based on bids, *UCB*, and *LCB* estimates. The difference between elimination rules is the need for an agent's width to reach threshold 2^{-s} at stage s. Due to the similarities, the proof follows on the similar lines of the proof of Theorem 11, and hence we skip it. □

M-SupLinUCB-S

The mechanism *M-SupLinUCB-S* follows the same structure as that of mechanism *M-ELinUCB-SB*. The only change is that the allocation rule $\mathcal{A}$ is given by *SupLinUCB-S* (Algorithm 14).

4.4.3 *M-SupLinUCB-S*: Game-Theoretic Analysis

Theorem 15. M-SupLinUCB-S *is ex-post incentive compatible (EPIC) and ex-post individually rational (EPIR).*

Proof. The result follows from Theorem 10 and by ex-post monotonicity of $\mathcal{A}$ defined in Algorithm 14. □

We have presented the results for single-slot SSA for ease of exposition, which can be carefully extended to multi-slot SSA with new ancillary notation and steps. We have presented the algorithm *SupLinUCB-M* for multi-slot allocation. We also have provided the algorithm's construction and its game-theoretic and regret analysis.

4.5 Multi-slot SSA

Algorithm 16 *SupLinUCB-M:(Adapted from SupLinUCB by [44] to satisfy monotonicity property and multi-slot allocation)*

Initialization: $S \leftarrow \ln T$, $\Psi^s_{i,t} \leftarrow \phi$ for all $s \in [\ln T]$
for t = 1,2,..., T **do**
  $alloc\_agts \leftarrow [\,]$, $unalloc\_slts \leftarrow [\,]$
  $j \leftarrow 1 + (t \bmod n)$
  for $k = 1, 2, \ldots, K$ **do**
    $s \leftarrow 1$ and $\hat{A}_1 \leftarrow [n] \backslash alloc\_agts$
    $j_k \leftarrow (j + k - 1) \mod n$
    repeat
      Use *BaseLinUCB-S* with $\{\Psi^s_{i,t}\}_{i\in[n]}$ and context vector x_t to calculate the width $w^s_{i,t}$ and upper confidence bound $ucb^s_{i,t} = (\hat{r}^s_{i,t} + w^s_{i,t}), \forall i \in \hat{A}_s$
      if $j_k \in \hat{A}_s$ and $w^s_{j_k,t} > 2^{-s}$ **then**
        Select $I^k_t = j_k$
        $alloc\_agts \leftarrow alloc\_agts \cup \{I^k_t\}$
        Update the index sets at all levels:
        $$\Psi^{s'}_{I^k_t,t+1} \leftarrow \begin{cases} \Psi^{s'}_{I^k_t,t} \cup \{t\} & \text{if } s = s' \\ \Psi^{s'}_{I^k_t,t} & \text{otherwise} \end{cases}$$
      else if $w^s_{i,t} \leq \frac{1}{\sqrt{T}}, \forall i \in \hat{A}_s$ **then**
        Select $I^k_t = argmax_{i\in\hat{A}_s} b_i \cdot (\hat{r}^s_{i,t} + w^s_{i,t})$
        $alloc\_agts \leftarrow alloc\_agts \cup \{I^k_t\}$
        Update index sets at all levels for I^k_t:
        $\Psi^{s'}_{I^k_t,t+1} \leftarrow \Psi^{s'}_{I^k_t,t}, \forall s' \in [S]$
      else if $w^s_{i,t} \leq 2^{-s}, \forall i \in \hat{A}_s$ **then**
        $\hat{A}_{s+1} \leftarrow \{i \in \hat{A}_s | b_i \cdot (\hat{r}^s_{i,t} + w^s_{i,t}) \geq \max_{a\in\hat{A}_s} b_a \cdot (\hat{r}^s_{a,t} + w^s_{a,t}) - 2^{1-s}\}$
        $s \leftarrow s + 1$
      else
        $unalloc\_slts \leftarrow unalloc\_slts \cup \{k\}$
    until I^k_t is selected OR slot k added in $unalloc\_slts$ set
  $c = 1$
  for $k \in unalloc\_slts$ **do**
    $I^k_t = argmax^c_{i\in\{N\backslash alloc\_agts\}} b_i \cdot (\hat{r}^1_{i,t} + w^1_{i,t})$
    $c \leftarrow c + 1$

We first introduce the additional notations required for a multi-slot setting.

The center has a K number of slots available for allocation, and the slots are indexed as $1, 2, \ldots, K$. I^k_t denotes the index of the agent selected by the algorithm at round t and slot index k. We have considered semi-bandit feedback where the center receives the feedback for each agent selected $\{I^k_t\}_{k\in[1,\ldots,K]}$ at each round t. We have considered additive reward where the total reward (click) the center gets is the sum of individual reward (click) received by each selected agent. We have assumed that each slot has

equal relevance, .i.e., the probability of getting a click on the ad of any agent i when placed at slot k at any round t is given as:

$$\mathbb{P}[r_{i,t}^k | x_t] = \mu_i(x_t) = \theta_i^\intercal x_t$$

Thus, the expected valuation of agent i when displayed at k^{th} slot at round t is given as $v_i \mu_i(x_t)$, which is independent of the slot. We modify the regret definition to capture the multi-slot variant of the problem which we define as:

$$\mathbb{R}_T(\mathcal{M}) = \sum_{t=1}^{T} \sum_{k=1}^{K} [\mu_{i_{k,t}^*}(x_t) \cdot b_{i_{k,t}^*} - \mu_{I_t^k}(x_t) \cdot b_{I_t^k}] \tag{4.2}$$

Here, $i_{k,t}^*(x_t)$ denote the index of the k^{th} highest expected valuation (based on bids) i.e., $i_{k,t}^*(x_t) = argmax_{i \in [n]}^k \{b_i \mu_i^k\}$.[2]

4.5.1 SupLinUCB-M

The algorithm *SupLinUCB-M* works similarly as *SupLinUCB-S*, with the difference of multi-slot allocation. To adapt for multi-slot allocation the algorithm additionally maintains two sets $alloc\_agts$ and $unalloc\_slts$. At the start of each round t, a designated agent is assigned for each slot k in a round-robin order (line[7]). The inner *for* loop of slots follows similar steps as of *SupLinUCB-S*. It can be perceived as the algorithm *SupLinUCB-M* runs *SupLinUCB-S* for K number of times at each round t. If an agent gets selected in lines[11, 15], we add the agent into $alloc\_agts$ set to keep account of the selected agents to avoid any agent allocation more than one. If for any slot k, the algorithm reaches line[22], we delay the allocation on that slot after the termination of the inner for loop. The delay is done to preserve the round-robin allocation for multi-slot cases (As once an agent is selected, the algorithm removes it from the active set for other slots (line[6]) in that round t). At each round t, the slots in $unalloc\_slts$ are allocated the agents based on decreasing UCB values (lines[25-27]).

4.5.2 Monotonicity of SupLinUCB-M

Claim 6. *In* SupLinUCB-M*, for a fixed context-arrivals $\{x_t\}_t$ and click realization ρ, in each round t, for each agent i, and for any two bid vectors* b *and b′, if $i \in S_{act}(b,t) \cap S_{act}(b^{'},t)$, then*

$$\mu_i^-(b,t)/b_i = \mu_i^-(b',t)/b_i' \text{ and } \mu_i^+(b,t)/b_i = \mu_i^+(b',t)/b_i'$$

Proof. In *SupLinUCB-M* (Algorithm 16), at round t if an agent i is selected in the initial allocation to K slots and $i \in S_{act}(\cdot)$ then only μ_i^- and μ_i^+ are updated in that round. Since $i \in S_{act}(b,t) \cap S_{act}(b^{'},t)$ and *SupLinUCB-M* does the initial allocation based on round-robin order only, the updates are independent of bids. Thus, claim follows from the fact that contexts and click realizations are fixed. □

[2] $argmax_j^c\{u_i\}$: The index of the c^{th} largest element in a set of numbers $\{u_i\}_{j \in [n]}$.

Claim 2-4 can be easily verified for *LinUCB-MB* algorithm following the same line of proofs as followed in the corresponding claims using by Claim 6.

Theorem 16. *Allocation rule induced by SupLinUCB-M (Algorithm 16) is ex-post monotone.*

Proof. For a fixed context-arrivals $\{x_t\}_t$, click realization ρ, bids of agents except i, i.e., b_{-i} and two possible bids $b_i^+ > b_i$, let τ and τ^+ be the last round for i s.t. i is in active set with bids b_i and b_i^+ respectively. From Claim 4, $\tau^+ \geq \tau$. Thus, i will receive more number of rounds with bid b_i^+ as compared with bid b_i. □

M-SupLinUCB-M

The mechanism *M-SupLinUCB-M* follows a similar structure as that of mechanism *M-SupLinUCB-M*. The only change is that the allocation rule $\mathcal{A}$ is given by *SupLinUCB-M* (Algorithm 16).

4.5.3 *M-SupLinUCB-M*: Game-Theoretic Analysis

Theorem 17. M-SupLinUCB-M *is ex-post incentive compatible (EPIC) and ex-post individually rational (EPIR) mechanism.*

Proof. The result follows from Theorem 10 and by ex-post monotonicity of $\mathcal{A}$ defined in Algorithm 16. □

4.5.4 M-SupLinUCB-M: Regret Analysis

Theorem 18. *SupLinUCB-M has regret* $O(n^2\sqrt{dKT\ln KT})$ *with probability at least* $1-\kappa$ *if it is run with* $\alpha = \sqrt{\frac{1}{2}\ln\frac{2nKT}{\kappa}}$.

Proof. As we have considered additive reward and semi-bandit feedback, the regret analysis follows the reasoning as in the proof for regret of mechanism *M-SupLinUCB-S*. The design of the mechanism *M-SupLinUCB-M* is such that the effectively the mechanism *M-SupLinUCB-M* runs mechanism *M-LinUCB-S* for $(K \cdot T)$ number of rounds, which can be trivially observed from the two *for* loops (lines[2,5]). Hence the regret proof follows the proof of Theorem 13, with the effective number of rounds being KT, where K is the number of slots and T is the total number of actual rounds. □

4.6 Experimental Analysis

4.6.1 Data Preparation

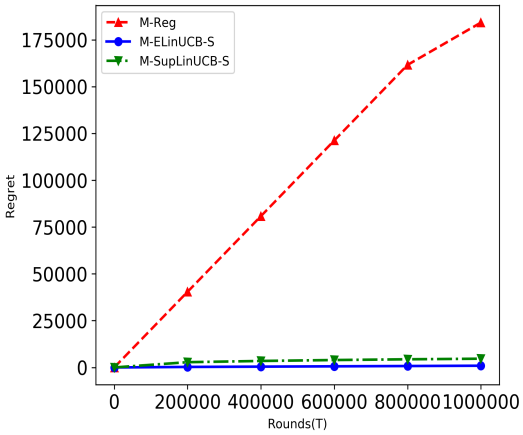
Our simulated data follow the structure and information availability found in a real-world system. Typically, a center has access to user features such as gender, age, geographic features, device model, and behavioral categories (which summarize the user's past preferences), which constitute the context. Note that each of the stated features can be discretized. Considering the above facts, we created the corpus of users χ: with $d = 4$, for each feature, we randomly select 4 possible different values from 0 to 100, and then by taking all possible combination of features, we generated random 256 (4^4) users. We normalize each $x \in \chi$ such that $x \in [0, 1]^d$, with $||x||_2 = 1$ and store these normalized contexts as a database χ_{db}.

We then select x_t uniformly at random from the context database χ_{db} for each round to generate a stochastic context. For each agent (advertiser), we generate $\theta_i \sim U([0, 1]^d)$ and then normalize it s.t. $||\theta_i||_2 = 1$. To simulate the clicks, at round t with the sampled x_t, we generate a click $r_{i,t}$ from Bernoulli distribution with parameter $\theta_i^T x_t$. We conduct experiments for 40 iterations, and for each iteration, we randomly generate a sequence of contexts from χ_{db} for $T = 10^6$ rounds. We generate a valuation of agent i for a click to be v_i sampled from the uniform distribution $[0, 1]$ and assume the agents bid truthfully; due to our mechanisms' truthfulness properties.

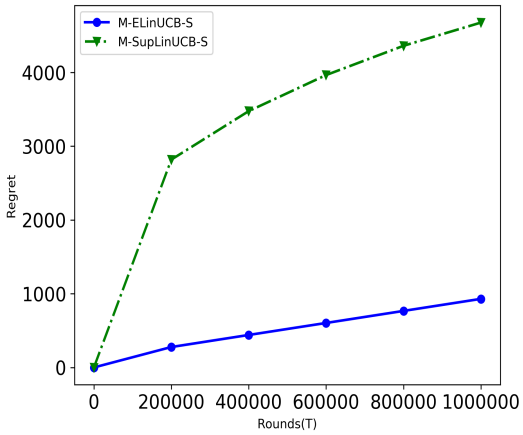
4.6.2 Results and Comparison

From our experimentation, we found learning parameter $\alpha = 1$ and batch size $bs = 100$ to be suitable for *M-ELinUCB-SB*. The metric of comparison between the mechanisms is regret, which is averaged over 40 iterations. Fig.4.2a compares the regret of *M-Reg*, *M-ELinUCB-SB*, and *M-SupLinUCB-S* for $n = 7$. When $n > 7$, *M-Reg* becomes infeasible as the number of exploration rounds λ exceeds the total number of rounds T, for $T = 10^6$. In terms of regret, it is evident that both the mechanisms *M-ELinUCB-SB* and *M-SupLinUCB-S* outperform *M-Reg* by a very large margin. Fig.4.2b highlights difference in experimental regret *M-ELinUCB-SB* and *M-SupLinUCB-S* (it is zoomed version from Fig. 4.2a). We can see that *M-ELinUCB-SB* experimentally performs approximately 5 times better than *M-SupLinUCB-S*; albeit the results are validated only on the randomly generated 256 contexts (χ_{db}).

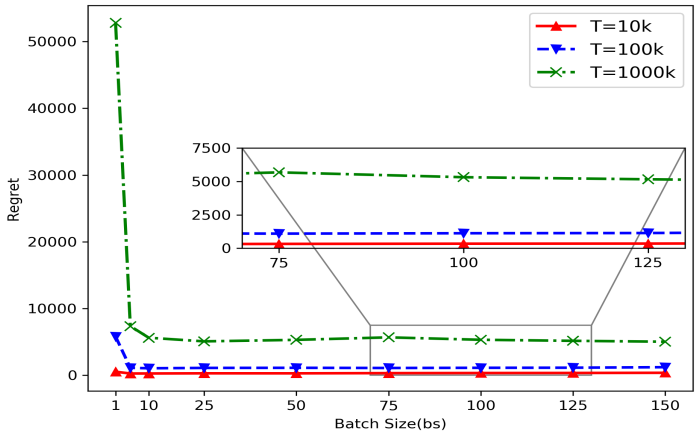
Though in theory, *M-ELinUCB-SB* has the worst regret, from simulations, the slope being very small, for reasonable values of T it outperforms *M-Reg*. Our experiments show that *M-ELinUCB-SB* and *M-SupLinUCB-S*, both have nearly negligible regret as compared to *M-Reg*. Fig.4.2c compares the regret incurred by *M-ELinUCB-SB* with varying batch size $bs \in \{1, 5, 10, 25, 50, 75, 100, 125, 150\}$ for $T \in \{10k, 100k, 1000k\}$. From the figure, observe the significant improvement in regret when we move from batch size $bs = 1$ to greater batch sizes. One may need to tune bs for different experimental setup.



(a) Regret vs Rounds (T)



(b) Regret vs Rounds (T)



(c) Regret vs Batch Size (bs)

Figure 4.2: Regret comparisons

4.7 Conclusion

We believe that ours is the first attempt to design a non-exploration-separated ConMAB mechanism. We focused on designing ConMAB mechanisms for sponsored search auction (SSA). For a single-slot, we first designed a LinUCB-based ex-post monotone allocation rule *ELinUCB-S*. We show that the introduction of batch size in *ELinUCB-S* significantly improves the regret while satisfying the ex-post monotone property. With this observation, we present *ELinUCB-SB*. Through simulations, it performs better for regret; however, theoretically, it may incur linear regret in carefully chosen contexts. To achieve sub-linear regret, we proposed another ex-post monotone allocation rule, *SupLinUCB-S*. We further extended these allocation rules to mechanisms, *M-ELinUCB-SB*, and *M-SupLinUCB-SB* satisfying EPIC and EPIR properties. We showed our mechanism performs significantly better than the existing mechanism *M-Reg* [56]. In summary, *M-SupLinUCB-S* is novel, truthful yet efficient non-exploration-separated ConMAB mechanism.

Although our mechanisms are randomized, they are game theoretically sound and scalable. Further, in terms of regret, *M-ELinUCB-SB* and *M-SupLinUCB-S* outperforms *M-Reg* in experiments. Theoretically, *M-SupLinUCB-S* matches the regret in non-strategic setting up to a factor of $O(n)$ which is the price of truthfulness. Along with SSA, this work can form a baseline for other applications such as crowdsourcing [28], smart grids [71], where similar settings arise to learn the stochastic parameters in the presence of strategic agents and side information (context) about the agents. It will be interesting to extend the work in the setting where the change in valuations and bids are permissible.

Chapter 5

Sleeping Combinatorial Bandits

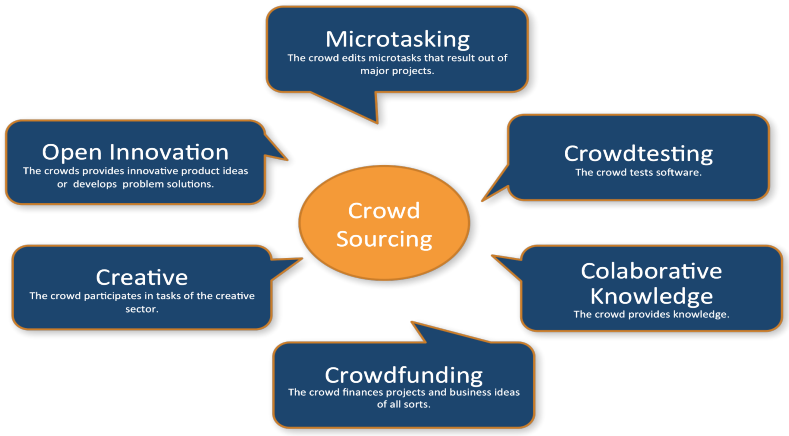


Figure 5.1: Types of crowdsourcing tasks

In this chapter, we study an interesting combination of sleeping and combinatorial stochastic bandits. In the mixed model studied here, at each discrete time instant, an arbitrary *availability set* is generated from a fixed set of *base* arms. An algorithm can select a subset of arms from the *availability set* (sleeping bandits) and receive the corresponding reward along with semi-bandit feedback (combinatorial bandits). We adapt the well-known CUCB algorithm in the sleeping combinatorial bandits setting and refer to it as CS-UCB. We prove — under mild smoothness conditions — that the CS-UCB algorithm achieves an $O(\log(T))$ instance-dependent regret guarantee. We further prove that (i) when the range of the rewards is bounded, the regret guarantee of CS-UCB algorithm is $O(\sqrt{T\log(T)})$ and (ii) the instance-independent regret is $O(\sqrt[3]{T^2\log(T)})$ in a general setting. Our results are quite general and hold under general environments — such as non-additive reward functions, volatile arm availability, a variable number of base-arms to be pulled — arising in practical applications. We validate the proven theoretical guarantees through experiments.

5.1 Introduction

The stochastic multi-armed bandit (MAB) problem is one of the fundamental online learning problems that captures the classic exploration vs. exploitation dilemma. A MAB algorithm, operating in an uncertain environment, is expected to optimally trade-off acquisition of new information with optimal use of information-at-hand to choose an action that maximizes the expected reward or, equivalently, minimizes the expected regret. In a classical stochastic MAB setup, an algorithm has to pull (aka select) a single arm (aka choice) at each time instant and receive a reward corresponding to a pulled arm. The reward from each arm is an independent sample from a fixed but unknown stochastic distribution. The goal is to minimize the *expected regret*; the difference between the expected cumulative reward of the best offline algorithm with known distributions and the expected cumulative reward of the algorithm.

In this chapter, we study a combination of two well studied extensions of classical stochastic MABs, namely *sleeping bandits* [75] and *combinatorial bandits* [53]. In the sleeping bandits setting, only a subset of base arms is available at each time instant. This variant, sometimes also known as volatile bandits [29] or mortal bandits [36], models many real-world scenarios such as crowdsourcing [83], online advertising [36], and network routing [29, 75] where an algorithm is restricted to select from only the available set of choices.

Another well studied generalization of the classical MAB setting is the combinatorial MAB (CMAB) problem [14, 35, 45, 52, 114]. Similar to sleeping bandits, this variant too provides an abstraction to many real-world decision problems. For instance, in an online advertising setup, the platform selects multiple ads to display at any point in time [52]; in crowdsourcing, the requester chooses multiple crowd workers at the same time [112] and in network routing algorithm has to choose a path instead of a single edge [76, 106]. Studying the two settings together presents interesting and non-trivial technical challenges.

We consider the semi-bandit feedback model and a general reward function (under mild smoothness constraints). In the semi-bandit feedback model, an algorithm observes the reward realizations corresponding to each of the selected arms and the overall reward for pulling the subset of arms. The smoothness properties on the reward functions studied in this chapter are similar to those in [41]. It is worth mentioning here that in the sleeping MAB setting, the conventional definition of regret is not appropriate as the best arm (or the best subset of arms in the combinatorial sleeping MAB case) may not be available at all time instants. Hence, we evaluate the performance of an algorithm in terms of its *sleeping regret* [75], defined as the difference between the expected reward obtained from best *available* arm and the arm pulled by the algorithm.

The chapter is organized as follows: In Section 6.3.1, we formally introduce the sleeping combinatorial bandits' problem and define Lipschitz smoothness and Bounded smoothness assumptions. The required notational setup is introduced, and CS-UCB algorithm is given in Section 5.3. In Section 5.4, we provide regret analysis of CS-UCB under Lipschitz smoothness and Bounded smoothness assumptions. In Section 5.5, we provide an in-depth verify of the theoretical results on simulated data with few

reward functions. The related literature is discussed in Section 5.6 and in Section 5.7, we conclude our chapter with a brief discussion on the results and future directions.

5.2 Model and Assumptions

In a classical stochastic multi-armed bandits (MAB) problem, at each discrete time step t, an algorithm pulls a single arm $i_t \in [k]$ and observes a random reward $X_{i_t,t}$. The random variables $(X_{i,t})_t$ are identical and independently distributed according to a distribution $\mathcal{D}_i(\mu_i)$. Here, μ_i is the mean of distribution $\mathcal{D}_i$. Note that the reward corresponding to arms $j \neq i_t$ is not observed. The reward distributions $(\mathcal{D}_i)_{i\in[k]}$ are unknown to the algorithm. Throughout this chapter we consider that the reward distributions have a bounded support. The algorithm's objective is to minimize expected regret defined as, $\mathcal{R}_{\text{ALG}}(T) = \mathbb{E}[\sum_{t=1}^{T}(X_{i^\star,t} - X_{i_t,t})]$. Here, $i^\star = \arg\max_i \mu_i$ denotes the best arm.

In this chapter we consider a sleeping combinatorial bandits problem with $[k] := \{1, 2, \cdots, k\}$ denoting the set of *base* arms and $\boldsymbol{\mu} \in [0,1]^k$, the vector of unknown mean qualities of the base arms. Similar to the classical stochastic MAB problem, each base arm i corresponds to an unknown distribution $\mathcal{D}_i$ with mean $\mu_i \in [0,1]$ over its quality. At each time instant t, a subset $A_t \subseteq [k]$ of the base arms become available. Throughout the chapter, we consider that A_t is an arbitrary non-empty subset. A decision maker (i.e. an algorithm) can pull any non-empty subset $S_t \subseteq A_t$ of arms and receive a reward $R_t := R(S_t, \boldsymbol{\mu})$. The reward depends upon the selected subset S_t and the mean qualities of the arms, $\boldsymbol{\mu}$. We define, $R_S := R(S, \boldsymbol{\mu})$ whenever the quality vector is clear from the context. Furthermore, the reward depends only on the qualities of pulled arms S_t [1]. We remark here that the classical stochastic bandits setting is a special case of our setting with $A_t = [k], |S_t| = 1$ and $R_t = X_{S_t,t}$ for all t.

For a given reward function R the problem reduces to finding a reward maximizing subset of arms. This problem, even when the qualities of the base arms are known, is known to be NP-hard in general [116]. However, many important settings, such as submodular reward functions, admit polynomial time approximation schemes that provide a decent approximation guarantee. To demarcate the computational problem of finding an optimal set of arms from effectively learning the quality distributions (and hence the learning an optimal set of arms to be pulled) we assume the existence of an (γ, β)-approximation oracle (denoted by (γ, β)-ORACLE), which, given an availability set A and a quality vector $\boldsymbol{\mu}$, outputs a set S such that $R_S(\boldsymbol{\mu}) \geq \gamma \cdot R_{S'}(\boldsymbol{\mu})$, for all $S' \in 2^A$ with the probability of at least β, with $\gamma, \beta \in (0,1]$. The computation oracle separates the learning task from the offline computation task and is extensively used in the literature [53, 41, 40].

For the semi-bandit feedback to work effectively, we assume some smoothness properties on the reward function. These smoothness properties ensure that when the learning parameters are estimated with a certain precision, one can approximate the true reward with high accuracy. Formally, the reward function $R_S(\boldsymbol{\mu})$, as a function of stochastic parameter $\boldsymbol{\mu}$, satisfies the following properties.

[1] That is for any set $S \subseteq [k]$, $R_S(\boldsymbol{\mu}) = R_S(\boldsymbol{\mu}')$ if $\boldsymbol{\mu}_i = \boldsymbol{\mu}_i'$ for all $i \in S$.

Property 1. ***Monotonicity****: Let $\boldsymbol{\mu}, \boldsymbol{\mu}^{'} \in [0,1]^k$ be two vectors such that $\boldsymbol{\mu}_i^{'} \geq \boldsymbol{\mu}_i$ for all $i \in [k]$ then, for any $S \subseteq [k]$, $R_S(\boldsymbol{\mu}^{'}) \geq R_S(\boldsymbol{\mu})$.*

The monotonicity property implies that the reward from any subset increase if the mean qualities of an base arms increase.

Property 2. ***Lipschitz Continuity****: There exists real valued constant $C \geq 1$ such that for all $S \subseteq [k]$, we have $|R_S(\boldsymbol{\mu}) - R_S(\boldsymbol{\mu}^{'})| \leq C \max_{i \in S} |\boldsymbol{\mu}_i - \boldsymbol{\mu}_i^{'}|$.*

Property 3. ***Bounded Smoothness****: There exists a strictly increasing function f such that for any $S \subseteq [k]$, $|R_S(\boldsymbol{\mu}) - R_S(\boldsymbol{\mu}^{'})| \leq f(\Lambda)$ whenever $\max_{i \in S} |\boldsymbol{\mu}_i - \boldsymbol{\mu}_i^{'}| \leq \Lambda$.*

In our first setting we study the reward function $R(\cdot)$ satisfying monotonicity (Property 1) and the Lipschitz continuity property (Property 2) whereas in the setting setting we consider Property 1 and Property 3. With a slight abuse of terminology, we call the first setup as Lipschitz smoothness and the second setup (i.e., monotonicity and bounded smoothness) as the Bounded smoothness.

The reward assumptions and regret notion considered in the chapter encompass many specialized settings studied in literature as a particular case. For instance, additive rewards with a fixed number of arms to pull [77], submodular rewards with volatile bandits [38], average reward, and so on. However, we remark here that the problem's technical treatment requires newer proof techniques as the existing proof techniques from combinatorial bandits setup do not generalize trivially to the sleeping combinatorial bandits setting.

Main Results of the Chapter

- In the Lipschitz smoothness setting, we show that CS-UCB achieves $O(\log(T)/\Delta_{\min})$ instance-dependent regret guarantee (See Theorem 19). Here, $\Delta_{\min}$ is the difference between the reward from an optimal super-arm and a sub-optimal super-arm with maximum reward.
- Note that for smaller values of $\Delta_{\min}$, the regret guarantee of Theorem 19 regret guarantee is vacuous. In Theorem 20, we show that CS-UCB attains an instance-dependent regret guarantee of $O(\sqrt{\sigma k T \log(T)})$. Here, $\sigma = \Delta_{\max}/\Delta_{\min}$. Note that, in contrast with Theorem 19 this result depends only on the range of rewards of super-arms. In particular, if the best and worst super-arms do not have a large reward ratio, the result in Theorem 20 is tight. We refer to this setting as *weak instance-dependent*.
- Next, in Theorem 21, we obtain a $O(\sqrt[3]{kT^2 \log(T)})$ instance-independent regret guarantee without any dependence on σ in a Lipschitz smoothness setting.
- Finally, in a Bounded smoothness setting, in Theorem 22, we show that CS-UCB attains $O(\log(T))$ regret guarantee. Though a similar result exists for the non-sleeping case [41]; the regret analysis does not trivially generalize to the combinatorial sleeping MAB setting.

5.3 The Setting

In this work, we consider that only a subset $A_t \subseteq [k]$ of arms is available at time t. Note that, A_t is revealed only at time t. Further, let $S_t \subseteq A_t$ be the set of arms pulled by the algorithm at time t. The set S_t is also called as a *super-arm*. To evaluate the performance of an algorithm with limited availability of arms, we extend the notion of regret considered for classical CMAB problem appropriately and call it a *sleeping regret* given by $\mathcal{R}_{\text{ALG}}(T) := \max_{(A_t)_{t=1}^T} \mathbb{E}_{\text{ALG}}\big[\sum_{t=1}^T (R_{S_t^\star} - R_{S_t})\big]$. Here, $S_t^\star \in \arg\max_{S \subseteq A_t} R_S$. Note that when $A_t = [k]$ for all t, we recover the setting of [41]. Next, we define the regret in the presence of (γ, β)-oracle. Let, B_t be the event that an oracle returns an γ-approximate solution at time t i.e. $B_t = \{R_{S_t} \geq \gamma \cdot R_{S_t^\star}\}$. Note that $\mathbb{P}(B_t) \geq \beta$. The expected sleeping regret of ALG with oracle access is given by,

$$\mathcal{R}_{\text{ALG}}(T) = \max_{(A_t)_{t=1}^T} \mathbb{E}_{\text{ALG}}\Big[\sum_{t=1}^T (\gamma \cdot \beta \cdot R_{S_t^\star} - R_{S_t})\Big]. \tag{5.1}$$

Notational Setup

We begin with the additional notation required to prove the results. For each base arm $i \in [k]$, let $N_{i,t}$ denotes the number of times arm i is pulled till time t and $\hat{\mu}_{i,t}$ be the average reward obtained from arm i till (and excluding) time t. Let

$$\overline{\mu}_{i,t} := \hat{\mu}_{i,t} + \sqrt{3\log(t)/2N_{i,t}}. \tag{5.2}$$

Following a standard terminology, we call $\overline{\mu}_{i,t}$ as the UCB estimate of arm i at time t. Furthermore, let $\Delta_S := \gamma \cdot \text{OPT}_A - R_S$ be the regret incurred by pulling super-arm S. Here, $\text{OPT}_A := R_{S^\star} = \max_{S \subseteq A} R_S$ denotes the optimal reward when the set of available arms is A. A super-arm $S \subseteq A$ is *bad* (sub-optimal), if $\Delta_S > 0$. For a given $A \subseteq [k]$, we define the set of bad super-arms as $S_B(A) = \{S \subseteq A | \Delta_S > 0\}$. Further, for a given $A \subseteq [k]$, define

$$\Delta_{\min}(A) = \gamma \cdot \text{OPT}_A - \max_{S \in S_B(A)} R_S \quad \text{and,}$$
$$\Delta_{\max}(A) = \gamma \cdot \text{OPT}_A - \min_{S \in S_B(A)} R_S.$$

Note that, for any availability set A, we have $\Delta_{\max}(A) \geq \Delta_{\min}(A) > 0$. The strict inequality follows from the definition of $S_B(A)$. Next, define $\Delta_{\max} = \max_{A \subseteq [k]} \Delta_{\max}(A)$ and $\Delta_{\min} = \min_{A \subseteq [k]} \Delta_{\min}(A)$. Arm i is called *saturated* if it is pulled for sufficiently many number of time steps, i.e., $N_{i,t} \geq \ell_t$, where ℓ_t works as threshold exploration. Note that a saturated arm at any time instant may become unsaturated in future. Further, we call a set A_t *explored* if all the arms in A_t are saturated, i.e., $N_{i,t} \geq \ell_t$ for all $i \in A_t$. First observe that if A_t is either empty or a singleton set then CS-UCB incurs a zero regret. Hence, without loss of generality we assume that $|A_t| \geq 2$ for all $t \leq T$. We first make following useful observation.

Observation 1. *Let $\ell_t \in \mathbb{R}_+$ be a positive number, $S \subseteq [k]$ be any non-empty set of arms and such that $N_{i,t} \geq \ell_t$ for all $i \in S$ and $\varepsilon_t := \sqrt{\frac{3\log(t)}{2\ell_t}}$, then $\mathbb{P}\{\max_{i\in S} |\overline{\mu}_{i,t} - \mu_i| < 2\varepsilon_t\} \geq 1 - 2|S|/t^3$.*

Proof. Using Hoeffding's lemma [65] we have,

$$\mathbb{P}\{|\hat{\mu}_{i,t} - \mu_i| \geq \varepsilon_t\} \leq 2e^{-2N_{i,t}\varepsilon_t^2} = 2e^{-3N_{i,t}\log(t)/\ell_t} \leq 2/t^3. \tag{5.3}$$

Here, $\hat{\mu}_{i,t}$ is the empirical mean reward of arm i till time t. The last inequality follows from the fact that $N_{i,t} \geq \ell_t$. Further, from the definition of $\overline{\boldsymbol{\mu}}$, with probability at least $1 - \frac{2}{t^3}$,

$$\varepsilon_t \underset{(i)}{>} |\hat{\mu}_{i,t} - \mu_i| \underset{(ii)}{=} |\overline{\mu}_{i,t} - \mu_i - \varepsilon_t| \underset{(iii)}{\geq} |\overline{\mu}_t - \mu_i| - \varepsilon_t. \tag{5.4}$$

Here, (i) follows from Equation 5.3, (ii) is immediate from the definition of $\boldsymbol{\mu}$ and finally (iii) follows from the triangle inequality. Thus, we have $|\overline{\mu}_{i,t} - \mu_i| < 2\varepsilon_t$ with probability atleast $1 - 2|S|/t^3$. □

CS-UCB

Algorithm 17 CS-UCB

Initialization:
for $i \in [k]$ **do**
    $N_{i,0} = 0$, $\overline{\boldsymbol{\mu}}_{i,0} = 1$, $X_{i,0} = 0$
for $t = 1, 2, 3, \ldots$ **do**
    Observe set of available arms as A_t
    if $\exists j \in A_t$, such that, $N_{j,t} = 0$ **then**
        Select $S_t = A_t$
    else
        $S_t = \text{ORACLE}(A_t, \overline{\boldsymbol{\mu}}_t)$
    Observe: Semi-bandit feedback as $X_{j,t} \in \{0, 1\}, \forall j \in S_t$ and $R_{S_t}(\boldsymbol{\mu})$;
    Update:

- $N_{i,t} = \begin{cases} N_{i,t-1} & \text{if } \forall i \notin S_t \\ N_{i,t-1} + 1 & \text{if } \forall i \in S_t \end{cases}$
- $\overline{\mu}_{i,t} = \frac{X_{i,1:t}}{N_{i,t}} + \sqrt{\frac{3\log(t)}{2N_{i,t}}}$
- $X_{i,1:t} = \begin{cases} X_{i,1:t-1} & \text{if } \forall i \notin S_t \\ X_{i,1:t-1} + X_{i,t} & \text{if } \forall i \in S_t \end{cases}$

Note that the proposed CS-UCB algorithm is the same as CUCB [41] except that at each time, only a subset of the arms is available, and the regret notion considered is sleeping regret instead of conventional regret. Similar to CUCB, we assume that the algorithm has access to a (γ, β)-approximation oracle.

At each time t, CS-UCB receives the set of available arms A_t. If there is a base arm in A_t which is not pulled previously, an algorithm pulls all the available arms. For each time instances where all available arms are pulled atleast once, CS-UCB obtains $S_t = \textsc{Oracle}(\overline{\boldsymbol{\mu}}_t, A_t)$. Here, $\overline{\boldsymbol{\mu}}_t$ represent the vector of UCB estimates given by Equation 5.2. The algorithm then pulls a super-arm S_t and obtain rewards $R_{S_t}(\boldsymbol{\mu})$ and an individual base arm rewards (semi-bandit feedback) $X_{i,t}$ for each $i \in S_t$. Finally, CS-UCB update parameters

- $N_{i,t+1} = N_{i,t} + \mathbb{1}(i \in S_t)$
- $\overline{\mu}_{i,t+1} = \frac{N_{i,t} \cdot \hat{\mu}_{i,t} + \mathbb{1}(i \in S_t) \cdot X_{i,t}}{N_{i,t} + \mathbb{1}(i \in S_t)} + \sqrt{\frac{3\log(t)}{N_{i,t} + \mathbb{1}(i \in S_t)}}$.

Note that the regret (Equation 5.1) depends on the rewards from the base arm $X_{i,t}$ only through $R(.)$. Further, observe that when $A_t = [k]$ for all t, the sleeping regret is same as conventional regret guarantee and CS-UCB is same as CUCB; hence the regret guarantees of [41] will apply.

5.4 Regret Analysis of CS-UCB

Our first result shows that under the Lipschitz smoothness setting, CS-UCB incurs a logarithmic instance-dependent regret. However, note that the regret depends inversely on the $\Delta_{\min}$ value. That is, for arbitrarily smaller values of $\Delta_{\min}$, the regret bound is vacuous. In Theorem 20, we prove that the weak instance-dependent regret of the proposed algorithm is $O(\sqrt{\sigma k T \log(T)})$. Here, $\sigma = \Delta_{\max}/\Delta_{\min}$; i.e., this result depends only on the ratio of the maximum and minimum achievable rewards. Finally, in Theorem 21, we show that the instance-independent regret of the proposed algorithm is $O(\sqrt[3]{kT^2 \log(T)})$ in general. We begin with the following observation.

Observation 2. *For all time instances t such that $N_{i,t} > 0$ and the reward function satisfies monotonicity and Lipschitz continuity (Properties 1 and 2), $\Delta_{S_t} \leq C\big(1 + \sqrt{3\log(T)/2}\big)$.*

Proof. The monotonicity property and Lipschitz smoothness implies that,

$$|R_{S_t}(\overline{\boldsymbol{\mu}}_t) - R_{S_t}(\boldsymbol{\mu})| = R_{S_t}(\overline{\boldsymbol{\mu}}) - R_{S_t}(\boldsymbol{\mu}) \leq C \max_{i \in S_t} |\overline{\mu}_{i,t} - \mu_i|$$

(Monotonicity property and Lipschitz property)

However,

$$R_{S_t}(\overline{\boldsymbol{\mu}}_t) - R_{S_t}(\boldsymbol{\mu}) \geq \gamma \cdot R_{S_t^\star}(\overline{\boldsymbol{\mu}}_t) - R_{S_t}(\boldsymbol{\mu}_t) \geq \gamma \cdot R_{S_t^\star}(\boldsymbol{\mu}_t) - R_{S_t}(\boldsymbol{\mu}) = \Delta_{S_t}. \tag{5.5}$$

Further, from the definition of $\overline{\mu}_{i,t} = \hat{\mu}_{i,t} + \varepsilon_{i,t}$, where $\varepsilon_{i,t} = \sqrt{3\log(T)/2N_{i,t}}$ for arm i till time t, we have

$$|\overline{\mu}_{i,t} - \mu_i| \leq |\hat{\mu}_{i,t} - \mu_i| + \varepsilon_{i,t} \leq 1 + \sqrt{3\log(T)/2}. \tag{5.6}$$

From Eq. 3 and Eq. 4, observe that $\Delta_{S_t} \leq C \max_{i \in S_t} |\overline{\mu}_{i,t} - \mu_i| \leq C(1 + \sqrt{3\log(T)/2})$. □

We are ready to present out first result.

Theorem 19. *The expected sleeping regret incurred by* CS-UCB *when the reward function satisfies Lipschitz condition (Properties 1 and 2) is given by*

$$\mathcal{R}_{\text{CS-UCB}}(T) \leq 2\beta kC \left[\zeta(3)(1 + \sqrt{\frac{3\log(T)}{2}}) + 3\frac{\sigma C \log(T)}{\Delta_{\min}} \right]$$

Here, ζ is the Reimann zeta function and $\sigma = \Delta_{\max}/\Delta_{\min}$.

Proof Outline: Set $\ell_t := 6C^2\log(t)/\Delta_{\min}^2$ and $\varepsilon_t := \sqrt{3\log(t)/2\ell_t}$ and divide the time instants into sets T_e and T_u as described as follows. Let T_e be the set of time instances t such that A_t is explored, i.e., $T_e = \{t \leq T : N_{i,t} \geq \ell_t, \forall i \in A_t\}$ and $T_u = [T] \setminus T_e$. Further let, for $t \in T_u$, $A_{e,t}$ be the set of saturated arms that are available at time t, i.e., $A_{e,t} := \{i | N_{i,t} \geq \ell_t\}$ and $A_{u,t} := A_t \setminus A_{e,t}$. We have

$$\begin{aligned} T_u &= \{t : \exists j \in A_{u,t}\} \\ &= \underbrace{\{t : \exists j \in A_{u,t} \cap S_t\}}_{D} \cup \underbrace{\{t : \forall j \in A_{u,t}, j \notin S_t\}}_{E}. \end{aligned}$$

We bound the sleeping regret incurred in disjoint sets T_e, D and E separately. Recall that B_t is an event that the oracle returns γ-approximate solution i.e. $R_{S_t}(\overline{\boldsymbol{\mu}}_t) \geq \gamma \cdot R_{S^\star}\{\overline{\boldsymbol{\mu}}_t\}$. We begin with following supporting lemmas.

Lemma 6. *For all $t \in T_e$ we have $\mathbb{P}\{S_t \in S_B(A_t) | B_t\} \leq 2|A_t|t^{-3}$.*

Proof. Let $S_t^\star = \arg\max_{S \subseteq A_t} R_S$ and the event B_t has occurred. We prove the lemma using the following supporting claim.

Claim 7. *Let $t \in T_e$ and $S_t = \textsc{Oracle}(A_t, \overline{\boldsymbol{\mu}})$ and $S_t^{'} = \textsc{Oracle}(A_t, \boldsymbol{\mu})$. Then $\mathbb{P}\{R_{S_t}(\boldsymbol{\mu}) = R_{S_t^{'}}(\boldsymbol{\mu})\} \geq 1 - 2|A_t|/t^3$.*

To see the proof of the lemma observe that

$$R_{S_t}(\boldsymbol{\mu}) = R_{S_t^{'}}(\boldsymbol{\mu}) \geq \gamma \cdot R_{S_t^\star}(\boldsymbol{\mu}) = \gamma \cdot \textsc{Opt}_{\boldsymbol{\mu}}(A_t).$$

The first equality in the above equation is true with probability atleast $1 - 2|A_t|/t^3$ from Claim 7. The first inequality holds from the fact that the event B_t has occurred. Hence, we have $\mathbb{P}(S_t \notin S_B(A_t)|B_t) \geq 1 - 2|A_t|/t^3$. This completes the proof of the lemma. □

Proof of Claim 7. First note that, it is enough to show that $S_t^{'} = S_t$. However, these sets might not be unique and hence we assume $S_t^{'} \neq S_t$. Let $Q_t^{\star} \in \arg\max_{S \in A_t} R_S(\boldsymbol{\mu})$ and $S_t^{\star} \in \arg\max_{S \in A_t} R_S(\overline{\boldsymbol{\mu}})$. From the monotonicity property of R and the definition of $\overline{\boldsymbol{\mu}}$ it holds that

$$R_{S_t}(\overline{\boldsymbol{\mu}}_t) \geq R_{S_t^{'}}(\overline{\boldsymbol{\mu}}_t) \geq R_{S_t^{'}}(\boldsymbol{\mu}) \tag{5.7}$$

Here, the first inequality follows from the optimality of S_t with respect to $\overline{\boldsymbol{\mu}}_t$ and the second inequality follows from the monotonicity property. For contradiction, let us assume that $S_t \neq S_t^{'}$ and $R_{S_t^{'}}(\boldsymbol{\mu}) > R_{S_t}(\boldsymbol{\mu})$. Using this inequality with Equation 5.7 we get $R_{S_t}(\overline{\boldsymbol{\mu}}_t) > R_{S_t}(\boldsymbol{\mu})$. From Lipschitz property we have,

$$R_{S_t}(\overline{\boldsymbol{\mu}}_t) - R_{S_t}(\boldsymbol{\mu}) = |R_{S_t}(\overline{\boldsymbol{\mu}}_t) - R_{S_t}(\boldsymbol{\mu})| \leq C \max_{i \in S_t} |\overline{\mu}_{i,t} - \mu_i|. \tag{5.8}$$

Let $\ell_t = 6C^2 \log(t)/\Delta_{\min}^2$. As $t \in T_e$, we have $N_{i,t} \geq \ell_t$ for all $i \in A_t$. Hence, from Observation 1, with probability atleast $1 - \frac{2|A_t|}{t^3}$, we have, $\max_{i \in S_t} |\overline{\mu}_{i,t} - \mu_i| \leq \max_{i \in A_t} |\overline{\mu}_{i,t} - \mu_i| < 2\varepsilon_t$. This gives, $R_{S_t}(\overline{\boldsymbol{\mu}}_t) - R_{S_t}(\boldsymbol{\mu}) < 2C \cdot \varepsilon_t = \Delta_{\min}$. To see the last inequality recall from Observation 1 that $\varepsilon_t = \sqrt{\frac{3\log(t)}{2\ell_t}}$. Hence, we have $\Delta_{\min} > \gamma \cdot R_{S_t^{\star}}(\overline{\boldsymbol{\mu}}) - R_{S_t}(\boldsymbol{\mu}) \geq \gamma \cdot R_{S_t^{\star}}(\boldsymbol{\mu}) - R_{S_t}(\boldsymbol{\mu})$. This contradicts the definition of $\Delta_{\min}$. Thus, with probability atleast $1 - \frac{2|A_t|}{t^3}$ we have that $R_{S_t}(\overline{\boldsymbol{\mu}}_t) = R_{S_t}(\boldsymbol{\mu}_t)$. This completes the proof of the claim. □

Lemma 7. $|D| \leq k\ell_T$.

Proof. Recall that by definition, we have $|D| := |\{t : \exists j \text{ such that } j \in A_{u,t}, j \in S_t\}|$. Hence we have,

$$|D| = |\{t : \exists j \text{ such that } N_{j,T} < \ell_T\}| \leq \sum_{j=1}^{k} |\{t : N_{j,T} \leq \ell_T\}| \leq k\ell_T.$$

□

Lemma 8. *For all $t \in E$ we have,* $\mathbb{P}\{S_t \in S_B(A_t) | B_t\} \leq 2|S_t|/t^3$.

Proof. Consider $t \in E$ and recall from Lemma 6 that $Q_t^{\star} = \arg\max_{S \in A_t} R_S(\boldsymbol{\mu})$. We have,

$$R_{S_t}(\overline{\boldsymbol{\mu}}_t) \geq \gamma \cdot R_S(\overline{\boldsymbol{\mu}}_t^{\star}), \quad \forall S \subseteq A_t.$$

For all $j \in S_t$ we have $N_{j,t} \geq \ell_t$. Hence from Observation 1, with probability atleast $1 - \frac{2|S_t|}{t^3}$ we get,

$$\max_{j \in S_t} |\overline{\mu}_{j,t} - \mu_j| < 2\varepsilon_t. \tag{5.9}$$

This implies,

$$
\begin{aligned}
& |R_{S_t}(\overline{\boldsymbol{\mu}}_t) - R_{S_t}(\boldsymbol{\mu})| < \Delta_{\min} && \text{(Lipschitz property(Property 2))} \\
\implies & R_{S_t}(\overline{\boldsymbol{\mu}}_t) - R_{S_t}(\boldsymbol{\mu}) < \Delta_{\min} \leq \Delta_{\min}(A_t) && \\
& \gamma \cdot R_{S_t^\star}(\overline{\boldsymbol{\mu}}_t) - R_{S_t}(\boldsymbol{\mu}) < \Delta_{\min}(A_t). && \text{(As, } R_{S_t}(\overline{\boldsymbol{\mu}}_t) \geq \gamma \cdot R_{S_t^\star}(\overline{\boldsymbol{\mu}}_t)\text{)}
\end{aligned}
$$

From the definition of $\Delta_{\min}(A_t)$ and monotonicity property (Property 1) we have, a contradiction. Hence, $S_t \notin S_B(A_t)$, which implies that $\mathbb{P}\{S_t \in S_B(A_t)\} \leq \frac{2|S_t|}{t^3}$ for all $t \in E$. Hence, $\mathbb{P}\{S_t \in S_B(A_t)|B_t\} \leq 2|S_t|/t^3$. □

Lemma 6 establishes that when all the base arms in the availability set are sufficiently explored, then the set S_t returned by the oracle is an optimal set with high probability. This result follows from the fact that, as all the available arms are sufficiently pulled in the past, $\overline{\boldsymbol{\mu}}$ is sufficiently close to $\boldsymbol{\mu}$. Lemma 7 follows directly from the fact that each base arm remains unsaturated till atmost ℓ_T pulls.

Finally, in Lemma 8 we handle the case that the availability set contains both saturated and unsaturated base arms. Note that the previous two lemmas also hold for CMAB settings. However, in contrast with CMAB, in our setting, the availability sequence may be such that at each time instant, only a few explored arms are available, and this may lead to high regret. Lemma 8 dismisses this hypothesis. First, note that only those base arms that are *available* but *not-pulled* are responsible for the regret. Furthermore, suppose an arm is available, and it is not pulled for many time instances. In that case, its UCB estimate increases and hence increasing its chances of getting pulled in the future due to the monotonicity assumption. This means that an optimal subset of the availability set will be pulled after some time with high probability.

Putting everything together: For a given arbitrary availability sequence $(A_t)_{t=1}^T$, the regret of CS-UCB is given as

$$\begin{aligned}
&\mathcal{R}_{\text{CS-UCB}}(T) = \mathbb{E}\big[\sum_{t\in[T]} \gamma\cdot\beta\cdot R_{S_t^\star} - R_{S_t}\big] \\
\leq &\mathbb{E}\big[\sum_{t\in[T]} \gamma\cdot R_{S_t^\star} - R_{S_t} \mid B_t\big]\cdot\beta \\
\leq &\big[\sum_{t\in T_e\cup E} \mathbb{P}\{S_t\in S_B(A_t)|B_t\}\cdot\Delta_{S_t} \\
&+\sum_{t\in D}\mathbb{P}\{S_t\in S_B(A_t)|B_t\}\cdot\Delta_{\max}\big]\cdot\beta \\
\leq &\big[\sum_{t\in T_e\cup E} 2\frac{|A_t|}{t^3}\Delta_{S_t} + k\ell_T\Delta_{\max}\big]\cdot\beta && \text{(From Lemmas 6, 7 and 8)} \\
\leq &\big[2kC\big(1+\sqrt{\frac{3\log(T)}{2}}\big)\sum_{t=1}^{\infty}1/t^3 + \frac{6kC^2\log(T)}{\Delta_{\min}^2}\cdot\Delta_{\max}\big]\cdot\beta && \text{(from Observation 2)} \\
\leq &\Big[2kC\zeta(3)\Big(1+\sqrt{3\log(T)/2}\Big) + \frac{6C^2k\sigma\log(T)}{\Delta_{\min}}\Big].\beta
\end{aligned}$$

Since the last equation holds for any arbitrary sequence $(A_t)_{t=1}^T$, it also holds for an adversarially chosen availability sequence. This completes the proof of the theorem. □

Notice that the regret guarantee in Theorem 19 depends on the value of $\Delta_{\min}$. If this value is sufficiently low the regret guarantee is vacuous. In the next result, we show a weak instance-dependent regret guarantee where the regret is given in terms of the ratio $\Delta_{\max}/\Delta_{\min}$.

Theorem 20. *The weak instance-dependent sleeping regret of* CS-UCB *when the reward function satisfies Lipschitz condition (properties 1 and 2) is given by*

$$\mathcal{R}_{\text{CS-UCB}}(T) \leq 4C\sqrt{6k\sigma T\log(T)} + 2kC\zeta(3).$$

Here, $\zeta(.)$ is a Reimann zeta function and $\sigma = \Delta_{\max}/\Delta_{\min}$.

Proof. First, consider the case $\Delta_{\min} \geq C\sqrt{\frac{6k\sigma\log(T)}{T}}$. From Theorem 19 we have,

$$\begin{aligned}
\mathcal{R}_{\text{CS-UCB}}(T) &\leq \left[\frac{6C^2k\sigma\log(T)}{\Delta_{\min}} + 2kC\zeta(3)\Big(1+\sqrt{3\log(T)/2}\Big)\right] \\
&\leq C\sqrt{6k\sigma T\log(T)} + 2k\zeta(3)C(1+\sqrt{3\log(T)/2}) \quad (\text{as, } \Delta_{\min}\geq C\sqrt{6k\sigma\log(T)/T}) \\
&\leq 3C\sqrt{6k\sigma T\log(T)} + 2kC\zeta(3).
\end{aligned}$$

The last inequality follows for all $T \geq k$, from the fact that $\sqrt{6\log(T)}kC\zeta(3) \leq 2\sqrt{6\log(T)}kC \leq 2C\sqrt{6k\sigma T\log(T)}$.

Next, let $\Delta_{\min} < C\sqrt{6k\sigma\log(T)/T}$. Further, let $\eta \geq C\sqrt{6k\sigma\log(T)/T}$ be a constant. We decompose the regret Δ_{S_t} at any time t into two parts; i.e. $\Delta_{S_t} \geq \eta$ and $\Delta_{S_t} < \eta$, respectively. Thus, instance-independent sleeping regret of CS-UCB,

$$\mathcal{R}_{\text{CS-UCB}}(T) = \mathbb{E}\big[\sum_{t=1}^{T} \mathbb{1}(S_t \in S_B(A_t))\Delta_{S_t}\big] = \mathbb{E}\big[\sum_{t=1}^{T}[\mathbb{1}(S_t \in S_B(A_t), \Delta_{S_t} < \eta) + \mathbb{1}(S_t \in S_B(A_t), \Delta_{S_t} \geq \eta)]\Delta_{S_t}\big].$$

The first term is upper bounded by ηT. To bound the second term, consider a CSMAB instance such that $S_B^{'}(A) = S_B(A) \cap \{S \subseteq A | \Delta_{S_t} \geq \eta\}$. In this instance we have $\Delta_{\max}^{'} = \Delta_{\max}$ and $\Delta_{\min}^{'} = \eta$. Hence,

$$\begin{aligned}
\mathcal{R}_{\text{CS-UCB}}(T) &\leq \eta T + \left[\sum_{t=1}^{T} \mathbb{P}\{S_t \in S_B^{'}(A_t)\}\Delta_{S_t}\right] \\
&\leq \eta T + \frac{6C^2k\sigma\log T}{\Delta_{\min}^{'}} + 2kC\zeta(3)(1+\sqrt{3\log(T)/2}) && \text{(from Theorem 19)} \\
&\leq \eta T + \frac{6C^2k\sigma\log T}{\eta} + 2kC\zeta(3)(1+\sqrt{3\log(T)/2}). && \text{(As } \Delta_{S_t} \geq \eta\text{)}
\end{aligned}$$

Choose $\eta = C\left(\frac{6k\sigma\log T}{T}\right)^{1/2}$ to get the desired upper bound. □

It is easy to see that for large values of $\Delta_{\min}$ one can use the result of Theorem 19 to obtain the desired bound of Theorem 20. However, when $\Delta_{\min}$ is small i.e. $\Delta_{\min} < C\sqrt{6k\sigma\log(T)/T}$, the upper bound on regret is obtained by parametrized analysis with selecting parameter $\eta \in (\Delta_{\min}, \Delta_{\max}]$ appropriately to minimize the regret. Observe that the regret dependence of Theorem 20 on time horizon increase from $O(\log(T))$ to $O(\sqrt{T\log(T)})$ when we consider the weak instance-dependent regret guarantee. In the next result, we further relax the dependence on instance parameters (σ) to obtain a *strong* instance-independent regret guarantee of $O(\sqrt[3]{T^2\log(T)})$.

Theorem 21. *The instance-independent sleeping regret of* CS-UCB *when the reward function satisfies Lipschitz condition (Properties 1 and 2) is given by*

$$\mathcal{R}_{\text{CS-UCB}}(T) \leq C(1+\lambda)\cdot\sqrt[3]{6kT^2\log(T)} + 2k\lambda C\zeta(3)$$

where $\lambda = (1+\sqrt{3\log(T)/2})$.

Proof. Let the regret of selecting super-arm S_t at round t be, $\Delta_{S_t} := \gamma \cdot \text{OPT}_{A_t} - R_{S_t}(\boldsymbol{\mu})$, where $\text{OPT}_{A_t} := R_{S_t^\star} = \max_{S\subseteq A_t} R_S$. From Theorem 19 it is easy to see that the said instance-independent

upper bound holds if $\Delta_{\min} \geq (\frac{T}{\log(T)})^{-1/3}$. Hence, without loss of generality let $\Delta_{\min} < (\frac{T}{\log(T)})^{-1/3}$. Further, let $\eta \geq (\frac{T}{\log(T)})^{-1/3}$ be a constant. We decompose the regret at any time t into two parts, where the per round regret is at most η and larger than η. From Observation 2, observe that $\Delta_{S_t} \leq C\Big(1 + \sqrt{3\log(T)/2}\Big)$. Let $\lambda = \Big(1 + \sqrt{3\log(T)/2}\Big)$. Thus, instance-independent sleeping regret of CS-UCB,

$$
\begin{aligned}
\mathcal{R}_{\text{CS-UCB}}(T) &= \mathbb{E}\big[\sum_{t=1}^{T}(\gamma \cdot \beta \cdot R_{S_t^\star} - R_{S_t}))\big] \\
&= \left[\sum_{t=1}^{T}\mathbb{P}\{S_t \in S_B(A_t)|B_t\}\Delta_{S_t}\mathbb{1}\{\Delta_{S_t} < \eta\} + \sum_{t=1}^{T}\mathbb{P}\{S_t \in S_B(A_t)|B_t\}\Delta_{S_t}\mathbb{1}\{\Delta_{S_t} \geq \eta\}\right] \cdot \beta \\
&\leq \eta T + \sum_{t=1}^{T}\mathbb{P}\{S_t \in S_B(A_t) \cup \Delta_{S_t} \geq \eta | B_t\}\Delta_{S_t} \qquad (\beta \leq 1) \\
&= \eta T + \sum_{t \in T_e \cup E}\mathbb{P}\{S_t \in S_B(A_t) \cup \Delta_{S_t} \geq \eta | B_t\}\Delta_{S_t} + \sum_{t \in D}\mathbb{P}\{S_t \in S_B(A_t) \cup \Delta_{S_t} \geq \eta | B_t\}\Delta_{S_t} \\
&\leq \eta T + 2\zeta(3)kC\Big(1 + \sqrt{3\log(T)/2}\Big) + C\Big(1 + \sqrt{3\log(T)/2}\Big)\sum_{t \in D}\mathbb{P}\{\Delta_{S_t} \geq \eta\} \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(Observation 2)} \\
&\leq \eta T + 2k\lambda C\zeta(3) + C\lambda\sum_{t \in D}\mathbb{P}\{\Delta_{S_t} \geq \eta\} \\
&\leq \eta T + 2k\lambda C\zeta(3) + k\lambda C\frac{6C^2\log(T)}{\eta^2}.
\end{aligned}
$$

Choose $\eta = C\left(\frac{6k\log T}{T}\right)^{1/3}$ to get the following sleeping regret:

$$\mathcal{R}_{\text{CS-UCB}}(T) \leq C(1+\lambda) \cdot \sqrt[3]{6kT^2\log(T)} + 2k\lambda C\zeta(3).$$

□

First, using Theorem 19 we establish that the said instance-independent upper bound holds in this setting if $\Delta_{\min} \geq (\frac{T}{\log(T)})^{-1/3}$. Then, similar to Theorem 20, we split the regret at any time t into two parts, where the per time regret is at most η and larger than η with $\eta \geq (\frac{T}{\log(T)})^{-1/3}$. As stated previously, this result provides the instance-independent regret guarantee without any additional restrictions on minimum and maximum rewards.

Theorem 22. *The expected sleeping regret incurred by* CS-UCB *when the reward function satisfies bounded smoothness condition (Properties 1 and 3), is upper bounded by*

$$\mathcal{R}_{\text{CS-UCB}}(T) \leq \Big[\frac{6\log(T)}{(f^{-1}(\Delta_{\min}))^2} + 2\zeta(3)\Big]k \cdot \Delta_{\max}.$$

Proof. Following the similar 3 step proof of Theorem 19. We choose with $\ell_t = \frac{6\log(t)}{(f^{-1}(\Delta_{\min}))^2}$ and $\varepsilon_t = \sqrt{\frac{3\log(t)}{2\ell_t}}$ and divide the time instants into sets T_e and T_u as described in Section 5.4. Step 1 and 2 is proved as Lemma 9 and Lemma 10. Observe that Step 3 follows trivially as for in Theorem 19.

Lemma 9. *Let $t \in T_e$, when the reward function satisfies monotonicity and Lipschitz smoothness property, then $\mathbb{P}\{S_t \in S_B(A_t)|B_t\} \leq 2|A_t|t^{-3}$.*

Proof of the lemma. Let $\ell_t := \frac{6\log(t)}{(f^{-1}(\Delta_{\min}))^2}$ and $\varepsilon_t := \sqrt{\frac{3\log(t)}{2\ell_t}}$. We have

$$\mathbb{P}\{S_t \in S_B(A_t)\} = \mathbb{P}\{\exists i \in A_t : |\hat{\mu}_{i,t} - \mu_i| \geq \varepsilon_t, S_t \in S_B(A_t)|B_t\} + \mathbb{P}\{\forall i \in A_t : |\hat{\mu}_{i,t} - \mu_i| < \varepsilon_t, S_t \in S_B(A_t)|B_t\}. \tag{5.10}$$

We first prove an upper bound on the first term on the right side of the above expression. We have, for all the arms i in A_t,

$$\begin{aligned}
\mathbb{P}\{|\hat{\mu}_{i,t} - \mu_i| \geq \varepsilon_t\} &\leq 2e^{-2N_{i,t}\varepsilon_t^2} && \text{(from Hoeffding's inequality)} \\
&= 2e^{-N_{i,t}\frac{3\log(t)}{\ell_t}} \\
&\leq 2t^{-3}. && \text{(as } N_{i,t} \geq \ell_t\text{)}
\end{aligned}$$

Using union bound we get the following upper bound on the first term

$$\mathbb{P}\big\{\exists i \in A_t : |\hat{\mu}_{i,t} - \mu_i| \geq \varepsilon_t, S_t \in \mathcal{S}_B(A_t)\big\} \leq \mathbb{P}\{\exists i \in A_t : |\hat{\mu}_{i,t} - \mu_i| \geq \varepsilon_t\} \leq 2|A_t|t^{-3}. \tag{5.11}$$

Next, we bound the second term. From Equation 5.4 and the bounded smoothness property (Property 3), for any $S_t^{'} \subseteq A_t$, we have $|R_{S_t^{'}}(\overline{\boldsymbol{\mu}}_t) - R_{S_t^{'}}(\boldsymbol{\mu})| < f(2\varepsilon_t)$. In particular, for the selected super-arm S_t we have,

$$|R_{S_t}(\overline{\boldsymbol{\mu}}_t) - R_{S_t}(\boldsymbol{\mu})| < f(2\varepsilon_t). \tag{5.12}$$

This implies,

$$\begin{aligned}
R_{S_t}(\boldsymbol{\mu}) + \Delta_{\min} &= R_{S_t}(\boldsymbol{\mu}) + f(2\varepsilon_t) && \text{(As } f(2\varepsilon_t) = \Delta_{\min}\text{)} \\
&> R_{S_t}(\overline{\boldsymbol{\mu}}_t) && \text{(from Eq.5.12)} \\
&\geq \gamma \cdot R_{S_t^\star}(\overline{\boldsymbol{\mu}}_t) && \text{(As } S_t \text{ is optimal super-arm for } \overline{\boldsymbol{\mu}}\text{)} \\
&\geq \gamma \cdot R_{S_t^\star}(\boldsymbol{\mu}) = \gamma \cdot \text{OPT}_{A_t}. && \text{(from the monotonicity property)}
\end{aligned}$$

Hence, we have $\Delta_{\min} > \gamma \cdot \text{OPT}_{A_t} - R_{S_t}(\boldsymbol{\mu})$. This contradicts the definition of $\Delta_{\min}$ and hence we have that $\mathbb{P}\big\{\forall i \in A_t : |\hat{\mu}_{i,t} - \mu_i| < \varepsilon_t, S_t \in S_B(A_t)|B_t\big\} = \mathbb{P}\big\{\forall i \in A_t : |\hat{\mu}_{i,t} - \mu_i| < \varepsilon_t\big\} = 0$. This, Eq. 5.10 and Eq. 5.11 completes the proof of the lemma. □

Lemma 10. *For given t, if $\forall i \in S_t$, $N_{i,t} \geq \ell_t$ is true and reward function satisfies monotonicity and bounded smoothness property then*

$$\mathbb{P}\{S_t \in S_B(A_t)|B_t\} \leq \frac{2|S_t|}{t^3}.$$

Proof. Let $\ell_t := \frac{6\log(t)}{(f^{-1}(\Delta_{\min}))^2}$ and $\varepsilon_t := \sqrt{\frac{3\log(t)}{2\ell_t}}$. Consider $t \in E$, where $E = \{t \in T_u | \forall j \in S_t, N_{j,t} \geq \ell_t\}$, i.e., at each $t \in E$ the each arm in super-arm are saturated. We have,

$$R_{S_t}(\overline{\boldsymbol{\mu}}_t) \geq R_S(\overline{\boldsymbol{\mu}}_t), \quad \forall S \in 2^{A_t}. \tag{5.13}$$

Let $S_t^\star = \arg\max_{S \in A_t} R_S(\boldsymbol{\mu})$ be an optimal super-arm for given available arms A_t at time t. For all $j \in S_t$ we have $N_{j,t} > \ell_t$. Hence from Observation 1, with probability atleast $1 - \frac{2|S_t|}{t^3}$ we have,

$$\max_{j \in S_t} |\overline{\mu}_{j,t} - \mu_j| \leq 2\varepsilon_t. \tag{5.14}$$

This implies,

$$\begin{aligned}
&|R_{S_t}(\overline{\boldsymbol{\mu}}_t) - R_{S_t}(\mu)| < \Delta_{\min} && \text{(Property 3, } \ell_t \text{ and } N_{i,t} \geq \ell_t\text{)} \\
\implies\ & R_{\overline{\boldsymbol{\mu}}_t}(S_t) - R_{S_t}(\boldsymbol{\mu}) < \Delta_{\min} \leq \Delta_{\min}(A_t) \\
& R_{S_t^\star}(\overline{\boldsymbol{\mu}}_t) - R_{S_t}(\boldsymbol{\mu}) < \Delta_{\min}(A_t) && \text{(As, } R_{S_t}(\overline{\boldsymbol{\mu}}_t) \geq R_{S_t^\star}(\overline{\boldsymbol{\mu}}_t)\text{)} \\
& \implies R_{S_t}(\boldsymbol{\mu}) > \max_{S \in S_B(A_t)} R_S(\boldsymbol{\mu}). \\
& && \text{(by definition of } \Delta_{\min}(A_t) \text{ and monotonicity property)}
\end{aligned}$$

Hence, we have $\mathbb{P}\{S_t \notin S_B(A_t)|B_t\} \geq 1 - \frac{2|S_t|}{t^3}$, which implies that $\mathbb{P}\{S_t \in S_B(A_t)|B_t\} \leq \frac{2|S_t|}{t^3}$ for all $t \in E$. □

Putting everything together:

With this, the upper bound on sleeping regret of CS-UCB under Bounded smoothness setting is

$$\begin{aligned}
\mathcal{R}_{\text{CS-UCB}}(T) \leq & \beta \cdot \Delta_{\max}\Big[\sum_{t \in T_e} \mathbb{P}\{S_t \in S_B(A_t)|B_t\} + \sum_{t \in E} \mathbb{P}\{S_t \in S_B(A_t)|B_t\} + \sum_{t \in D} \mathbb{P}\{S_t \in S_B(A_t)|B_t\}\Big] \\
\leq & \beta \cdot \Big[\sum_{t=1}^{T} 2\frac{\Delta_{\max}}{t^3}|S_t| + \Delta_{\max}|D|\Big] && \text{(From Lemma 7, Lemma 9, and Lemma 10)} \\
\leq & \Big[2\zeta(3)k\Delta_{\max} + k\ell_T\Delta_{\max}\Big] \cdot \beta = \left[\frac{6C^2\log(T)}{(f^{-1}(\Delta_{\min}))^2} + 2\zeta(3)\right]\beta \cdot k \cdot \Delta_{\max}.
\end{aligned}$$

□

Note that the proof technique closely follow Theorem 19. We remark here that we recover the regret bound of [41] for non-sleeping combinatorial bandits case i.e., when $A_t = [k]$ for all t. Further, observe that when rewards are additive, i.e. $R_{S_t} = \sum_{i \in S_t} X_{i,t}$ one can achieve $\tilde{O}(\sqrt{T})$ regret bound [77]; however it is not clear if the $\tilde{O}(\sqrt{T})$ regret upper bound holds under bounded smoothness assumption. Finally, the instance-independent regret (Theorem 20 and Theorem 21) guarantee under bounded smoothness condition follows trivially by choosing $C = \sup_{x \in [0,1]} f(x)$.

5.5 Simulation Results

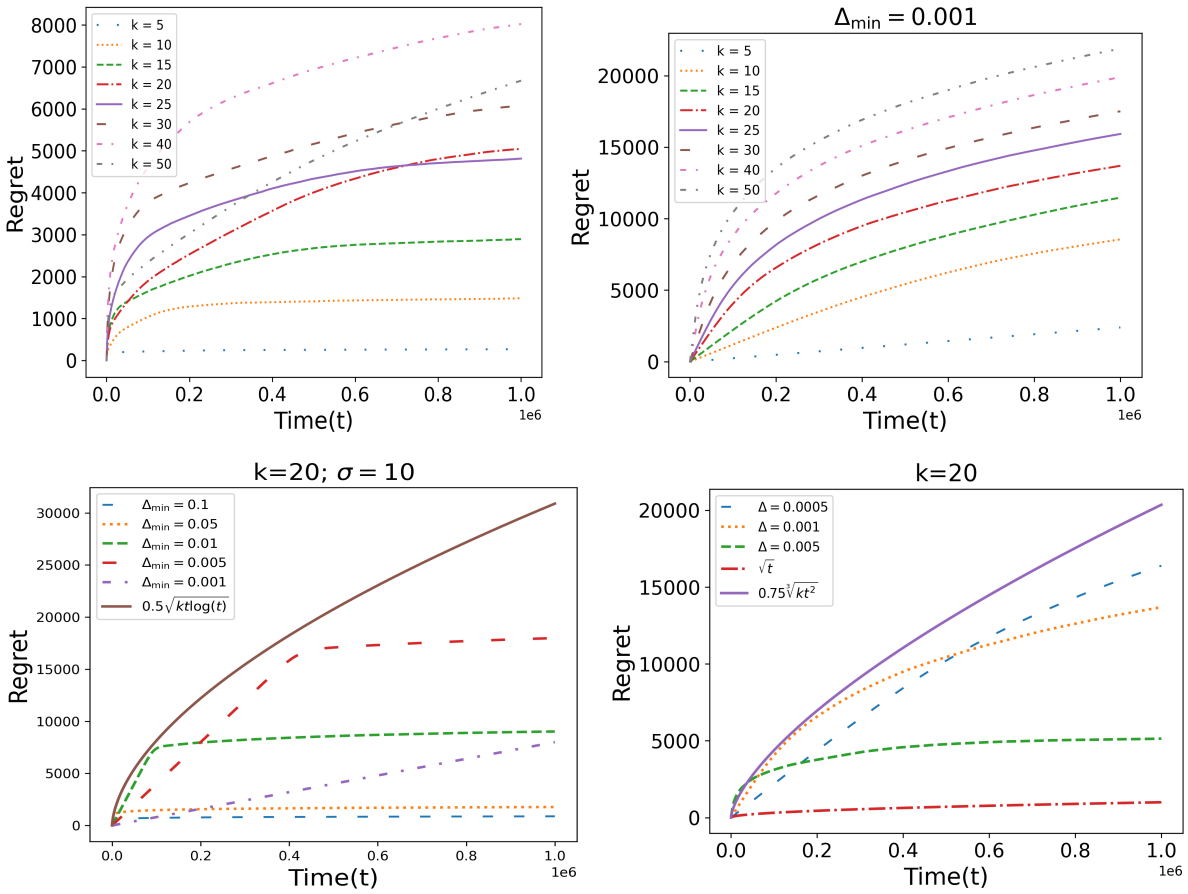


Figure 5.2: Regret Vs Time Plots For UtilReward: From L to R, (a) ExpOne: Instance-dependent regret with randomly generated qualities (Theorem 1) (b) ExpOne: Instance-dependent guarantee for $\Delta_{\min}$ = 0.001 (c) ExpTwo: Weak instance-dependent guarantee (Theorem 2) (d) ExpTwo: Instance-independent regret guarantee (Theorem 3).

In this section, we validate the chapter's theoretical results using different reward functions on simulated data. In particular, we perform experiments on two different combinatorial bandits settings studied in the literature [70, 76]. In the first setting, the average quality of base arm i takes the form $a_i \cdot \mu_i - b_i$;

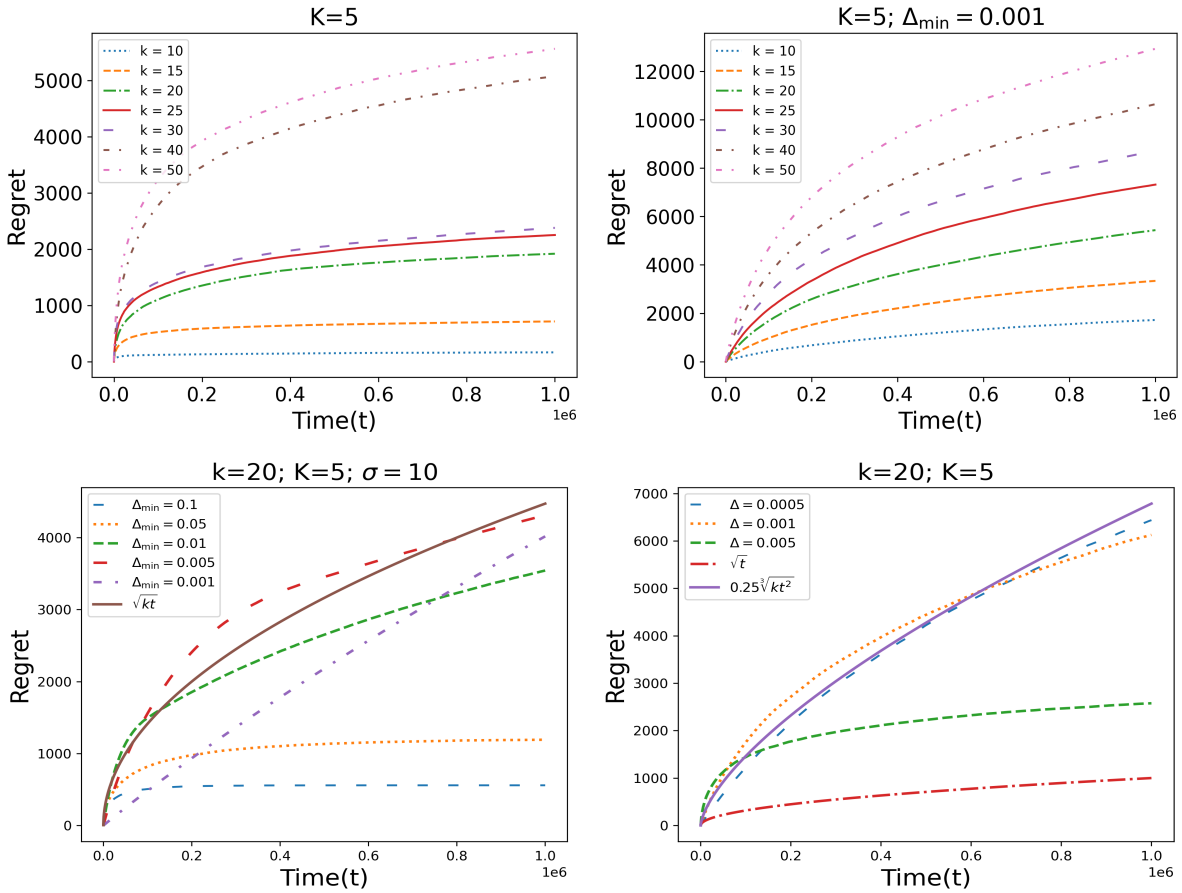


Figure 5.3: Regret Vs Time Plots For TopKReward: From L to R, (a) ExpOne: Instance-dependent regret with randomly generated qualities (Theorem 1) (b) ExpOne: Instance-dependent guarantee for $\Delta_{\min} = 0.001$ (c) ExpTwo: Weak instance-dependent guarantee (Theorem 2) (d) ExpTwo: Instance-independent regret guarantee (Theorem 3).

here μ_i is a mean of the Bernoulli random variable, and a_i and b_i are unknown constants. In this setting, the quality is also referred to as the utility from arm i, where $a_i \cdot X_i$ being a random reward with mean μ_i and b_i being the fixed cost corresponding to arm i. We call this reward setting [2] as UtilReward. The goal is to select all the available base arms with positive quality. In the second setting (which we call TopKReward), we consider the problem of pulling top K (in terms of quality) available arms and the reward function is additive [3]. In this setting, note that, if at some time t, $|A_t| \leq K$, all the available arms are pulled, and the regret at time instant t is zero. Further observe that, both the settings admit polynomial time exact oracles; i.e. $(1, 1)$-ORACLE.

[2] See [70] for detailed motivation and applications of this setting.

[3] More details and the regret analysis in *non-sleeping* case is given in [76].

Simulation Setup and Observations

We run two experiments for each of the reward settings mentioned above. In the first experiment, which we call ExpOne, the quality parameter μ_i of each of the base arms i is chosen independently from uniform distribution over interval $[0.3, 0.8]$. Quality feedback from the base arm $i \in S_t$ is an independent sample from a Bernoulli distribution with mean μ_i. The availability parameter corresponding to arm i is uniformly sampled from $[0.4, 0.9]$. Like the quality feedback, the availability of i is decided by a random draw from a Bernoulli distribution with a given availability parameter.

The second experiment, ExpTwo, is designed to validate the results of Theorem 20 and 21. The availability of base arms is generated using same approach as in the first experiment. However, the qualities of base arms is fixed to be close to each other. We validate the result of Theorem 20, by fixing the value of $\sigma := \Delta_{\max}/\Delta_{\min}$ and varying the values of $\Delta_{\min}$, and Theorem 21 by varying the values of $\Delta_{\min}$. Each of the experiments is executed over time horizon $T = 10^6$ and the average rewards from 50 independent runs.

We present the plots associated to UtilReward reward function in Fig. 5.2. The first two plots in Fig. 5.2 show that as $\Delta_{\min}$ value decreases, the expected regret guarantee of Theorem 19 becomes vacuous. The next two plots show that the the regret dependence on time horizon increases from $\sqrt{T}$ to $\sqrt[3]{T^2}$ for similar values of $\Delta_{\min}$ when we fix σ and change $\Delta_{\min}$ to arbitrary values of $\Delta_{\min}$ and $\Delta_{\max}$. Similar results were observed for different values of σ, k, Δ_{min} and reward function TopKReward (Fig. 5.3).

5.6 Related Work

The stochastic bandits problem has been extensively studied in the literature [10, 17, 79, 109]. We refer the reader to [81, 104] for a book exposition on multi-armed bandits and their applications. Most previous work in literature— with few exceptions such as [37, 75, 83] — assume that all the arms are available at all time instants. It is shown that the classical algorithms, adapted appropriately, are also optimal in a sleeping bandit setting [37, 75].

Combinatorial multi-armed bandits (CMAB) is another well studied variant of stochastic MAB problem which considers multi-pull setup [35, 39, 41, 45, 52, 53, 77, 83, 94, 114, 115]. [41] consider a general reward function with some smoothness condition and proposed CUCB, a UCB-style algorithm. In contrast, we consider arbitrary arm availability and general rewards and show that CUCB, when extended to sleeping bandits, achieves optimal regret guarantee. We also remark here that their analysis does not generalize to combinatorial sleeping bandits, and hence we need novel proof techniques to bound the sleeping regret in a CMAB setting. To the best of our knowledge, we are the first to address combinatorial sleeping MAB with a general reward structure and provide instance-dependent, and instance-independent regret upper bound.

The closest work to this work is [38]. Similar to their work, we consider semi-bandit feedback and combinatorial sleeping bandits framework. However, [38] considers contextual bandits setting, whereas we study a sleeping combinatorial MAB setting. The proposed algorithms (CC-MAB and CS-

MAB, respectively) differ crucially in how they carry out exploration. CC-MAB explores the subset of available arms if it contains at least a single unsaturated base arm ([38], Algorithm 2, Line 7). Hence, the exploitation is carried only if all the available base arms are saturated. In contrast, CS-UCB does not demarcate exploration and exploitation in this manner. So, even if some "obviously" bad super-arms are not explored, CS-UCB does not pull them. Also, there are following two important differences in the setting considered. Firstly, [38] consider that the reward function is submodular, whereas we consider general reward functions. Indeed, if the reward function satisfies submodularity, our results can be extended easily by considering $(1 - \frac{1}{e})$- approximation oracle. Secondly, they consider that the time horizon is apriori known to the algorithm, which may be an unrealistic assumption in many practical cases. Note that we provide an any-time regret guarantee, i.e., T is not given as an input to the algorithm. Also, they proved that CC-MAB achieves the regret of $O(\sqrt[3]{T^2 \log(T)})$ for the specific case of the submodular reward function, whereas we provide similar regret bound with more general reward functions. The recent work of [92] also studies contextual combinatorial bandits set up with sleeping arms and semi-bandit feedback. The authors consider a setting where the arms are differentiated based on the context.

5.7 Conclusion and Future Work

In this chapter, we considered combinatorial sleeping multi-armed bandits setting where a subset of arms is available at a given time instant. We analyzed the CS-UCB algorithm and analyzed its regret guarantee under two setups; Lipschitz smoothness and Bounded smoothness. We showed that under Lipschitz smoothness setting, CS-UCB achieves $O(\log(T)/\Delta_{\min})$ instance-dependent sleeping regret guarantee. Additionally, we prove that CS-UCB achieves $O(\sqrt{T \log(T)})$ weak instance-dependent regret under the assumption that the ratio of maximum and minimum achievable rewards is bounded. Also, we provide $O(\sqrt[3]{T^2 \log(T)})$ instance-independent regret in the most general case. We also show that CS-UCB in Bounded smoothness setting matches the conventional regret guarantee for the combinatorial MAB setting under the same set of assumptions (i.e., $O(\log(T))$). Finally, we validate the proven theoretical guarantees through experiments.

The instance-independent regret guarantee under Bounded smoothness setting remains an interesting open problem. Also, a finely tuned analysis with availability specific regret guarantees is an interesting future direction. This setup could be used together with other MAB settings, for instance, rotting bandits [82], where the arm pulling strategy may lead to the dropping of the arms.

Chapter 6

A Multi-Arm Bandit Approach To Subset Selection Under Constraints

We explore the class of problems where a central planner needs to select a subset of agents, each with different quality and cost. The planner wants to maximize its utility while ensuring that the average quality of the selected agents is above a certain threshold. When the agents' quality is known, we formulate our problem as an integer linear program (ILP) and propose a deterministic algorithm, namely DPSS that provides an exact solution to our ILP.

We then consider the setting when the qualities of the agents are unknown. We model this as a Multi-Arm Bandit (MAB) problem and propose DPSS-UCB to learn the qualities over multiple rounds. We show that after a certain number of rounds, τ, DPSS-UCB outputs a subset of agents that satisfy the average quality constraint with a high probability. Next, we provide bounds on τ and prove that after τ rounds, the algorithm incurs a regret of $O(\ln T)$, where T is the total number of rounds. We further illustrate the efficacy of DPSS-UCB through simulations. To overcome the computational limitations of DPSS, we propose a polynomial-time greedy algorithm, namely GSS, that provides an approximate solution to our ILP. We also compare the performance of DPSS and GSS through experiments.

6.1 Introduction

Almost all countries have cooperative societies that cater to developing sectors such as agriculture and handicrafts. We observed that some cooperatives, especially those that are consumer-oriented, such as Coop (Switzerland) or artisan cooperatives who operate their stores, lack a well-defined system to procure products from its many members (manufacturers, artisans, or farmers). Since the production is highly decentralized and usually not standardized, each producer has a different quality and cost of produce depending on various factors such as workmanship and the scale at which it operates. The central planner (say, the cooperative manager) has to carefully trade-off between each producer's qualities and cost to decide the quantity to procure from each producer so that it is most beneficial for the society as a whole.

This problem is not limited to cooperatives, but it is also faced in other familiar marketplaces. E-commerce platforms, like Amazon and Alibaba, have several sellers registered on their platform. For

each product, the platform needs to select a subset of sellers to display on its page while ensuring that it avoids low-quality sellers and does not display only the searched product's high-cost variants. Similarly, a supermarket chain may need to decide the number of apples to procure from the regional apple farmers, each with a different quality of produce, to maximize profits while ensuring that the quality standards are met.

We formulate this as a subset selection problem where a central planner needs to select a subset of these sellers/producers, whom we refer to as agents. In this work, we associate each agent with its quality and cost of production. The agent's quality refers to the average quality of the units produced by it; however, the quality of an individual unit of its product could be stochastic, especially in artistic and farm products. Thus, it becomes difficult to design an algorithm that guarantees constraint satisfaction on the realized qualities of the individual units procured. Towards this, we show that we achieve probably approximately correct (PAC) results by satisfying our constraint on the expected average quality of the units procured. Every unit procured from these agents generates revenue that is a function of its quality. The planner aims to maximize its utility (i.e., revenue - cost) while ensuring that the procured units' average quality is above a certain threshold to guarantee customer satisfaction and retention [4, 107]. When the agents' quality is known, we model our problem as an Integer Linear Program (ILP) and propose a novel algorithm, DPSS that provides an exact solution to our ILP.

Often, the quality of the agents is unknown to the planner beforehand. An E-commerce platform may not know its sellers' quality at the time of registration, and an artisan's quality of work may be hard to estimate until its products are procured and sold in the market. Thus, the planner needs to carefully learn the qualities by procuring units from the agents across multiple rounds while minimizing its utility loss. Towards this, we model our setting as a Multi-Arm Bandit (MAB) problem, where each agent represents an independent arm with an unknown parameter (here, quality). To model our subset selection problem, we consider the variant of the classical MAB setting where we may select more than one agent in a single round. This setting is popularly referred to as a Combinatorial MAB (CMAB) problem [45, 53, 77]. In studying CMAB, we consider the semi-bandit feedback model where the algorithm observes the quality realizations corresponding to each of the selected arms and the overall utility for selecting the subset of arms. The problem becomes more interesting when we also need to ensure our quality constraint in a CMAB problem. We position our work with respect to the existing literature in Section 6.2.

Typically, in a CMAB problem, the planner's goal is to minimize the *expected regret*, i.e., the difference between the expected cumulative utility of the best offline algorithm with known distributions of an agent's quality and the expected cumulative reward of the algorithm. However, the traditional definition of regret is not suitable in our setting as an optimal subset of agents (in terms of utility) may violate the quality constraint. Thus, we modify the regret definition to make it compatible with our setting. We propose a novel, UCB-inspired algorithm, DPSS-UCB, that addresses the subset selection problem when the agents' quality is unknown. We show that after a certain threshold number of rounds, τ, the algorithm satisfies the quality constraint with a high probability for every subsequent round, and under the revised regret definition, it incurs a regret of $O(\ln T)$, where T is the total number of rounds.

To address the computational challenges of DPSS which has a time complexity of $O(2^n)$, we propose a greedy-based algorithm, GSS that runs in polynomial time $O(n \ln n)$, where n is the number of agents. We show that while the approximation ratio of the utility achieved by GSS to that of DPSS can be arbitrarily small in the worst case, it achieves almost the same utility as DPSS in practice, which makes GSS a practical alternative to DPSS especially when n is large.

In summary, our contributions are:

- We propose a framework, SS-UCB, to model subset selection problem under constraints when the properties (here, qualities) of the agents are unknown to the central planner. In our setting, both the objective function and the constraint depends on the unknown parameter.
- We first formulate our problem as an ILP assuming the agents' quality to be known and propose a novel, deterministic algorithm, namely DPSS(Algorithm 18) to solve the ILP.
- Using DPSS, we design DPSS-UCB which addresses the setting where the agents' quality is unknown. We prove that after a certain number of rounds, $\tau = O(\ln T)$, DPSS-UCB satisfies quality constraint with high probability. We also prove that it achieves a regret of $O(\ln T)$ (Theorem 23).
- To address the computational limitation of DPSS, we propose an alternative greedy approach, GSS and GSS-UCB, that solves the known and the unknown settings, respectively. We show that while the greedy approach may not be optimal, it performs well in practice with a huge computational gain that allows our framework to scale to settings with a large number of agents.

The remaining of the chapter is organized as follows: In Section 6.2, we discuss the related works. In Section 6.3, we define our model and solve for the setting when the quality of the agents are known. In Section 6.4, we address the problem when the quality of the agents is unknown. In Section 6.5, we propose a greedy approach to our problem. In Section 6.6, we discuss our simulation-based analysis and conclude the chapter in Section 6.7.

6.2 Related Work

Subset selection is a well-studied class of problems that finds its applications in many fields, for example, in retail, vehicle routing, and network theory. Usually, these problems are modeled as knapsack problems where a central planner needs to select a subset of agents that maximizes its utility under budgetary constraints [117]. There are several variations to the knapsack, such as robustness [99], dynamic knapsacks [97], and knapsack with multiple constraints [103] studied in the literature. In this work, we consider a variant where the constraint is not additive, i.e., adding another agent to a subset does not always increase the average quality.

When online learning is involved, the stochastic multi-armed bandit (MAB) problem captures the exploration vs. exploitation trade-off effectively [28, 64, 74, 67, 70, 104, 110, 111]. The classical MAB problem involves learning the optimal agent from a set of agents with a fixed but unknown reward

distribution [17, 34, 104, 109]. Combinatorial MAB (CMAB) [39, 40, 41, 45, 52] is an extension to the classical MAB problem where multiple agents can be selected in any round. In [41, 42, 53], the authors have considered a CMAB setting where they assume the availability of a feasible set of subsets to select from. The key difference with our setting is that our constraint itself depends on the unknown parameter (quality) that we are learning through MAB. Thus, the feasible subsets that satisfy the constraint need to be learned, unlike the previous works. [41, 42, 53] also assumes the availability of an oracle that outputs an optimal subset given the estimates of the parameter as input, whereas we design such an oracle for our problem. Bandits with Knapsacks (BwK) is another interesting extension that introduces constraints in the standard bandit setting [9, 22, 23, 70] and finds its applications in dynamic pricing, crowdsourcing, etc. (see [9, 22]). Typically, in BwK, the objective is to learn the optimal agent(s) under a budgetary constraint (e.g., a limited number of selections) that depends solely on the agents' cost. However, we consider a setting where the selected subset needs to satisfy a quality constraint that depends on the learned quantities.

The closest work to ours is [70] where the authors present an assured accuracy bandit (AAB) framework where the objective is to minimize cost while ensuring a target accuracy level in each round. While they do consider a constraint setting similar to ours, the objective function in [70] depends only on the agents' cost and not on the agents' unknown qualities. Hence, it makes our setting different and more generalizable with respect to both AAB and CMAB, as in our setting, both the constraint and the utility function depend on the unknown parameter.

6.3 Subset Selection With Known Qualities of Agents

Here we assume that the agents' quality is known and consider the problem where a central planner C needs to procure multiple units of a particular product from a fixed set of agents. Each agent is associated with the quality and cost of production. C's objective is to procure the units from the agents such that the average quality of all the units procured meets a certain threshold. We assume that there is no upper limit to the number of units it can procure as long as the quality threshold is met.

In Section 6.3.1, we define the notations required to describe our model, formulate it as an integer linear program (ILP) in Section 6.3.3, and propose a solution to it in Section 6.3.4.

6.3.1 Model and Notations

1. There is a fixed set of agents $[n] = \{1, 2, \ldots, n\}$ available for selection for procurement by planner C.

2. Agent i has a cost of production, c_i, and capacity, k_i (maximum number of units it can produce).

3. The quality of the j^{th} unit of produce by agent i is denoted by Q_{ij}, which we model as a Bernoulli random variable.

4. For any agent i, the probability that Q_{ij} is 1 is defined by q_i, i.e., $E[Q_{ij}] = q_i$ for any unit j procured from agent i. q_i is also referred to as the quality of the agent in the rest of the chapter.

5. The utility for C to procure a single unit of produce from agent i is denoted by r_i, which is equal to its expected revenue[1] minus the cost of production, i.e., $r_i = Rq_i - c_i$, where R is the proportionality constant.

6. The quantity of products procured by C from the i^{th} agent is given by x_i.

7. The average quality of products procured by C is therefore equal to $\frac{\sum_{i\in[n]}\sum_{j=1}^{x_i} Q_{ij}}{\sum_{i\in[n]} x_i}$.

8. We define $q_{av} = \frac{\sum_{i\in[n]} x_i q_i}{\sum_{i\in[n]} x_i}$, which is the expected average quality of the units procured by C.

9. C needs to ensure that the average quality of all the units procured is above a certain threshold, $\alpha \in [0, 1]$.

10. The total utility of C is given by, $z = \sum_{i\in[n]} x_i r_i$.

Usually, an individual unit's quality Q_{ij} may not be quantifiable and can only be characterized by observing whether it was sold. Hence, we model it as a Bernoulli random variable.

6.3.2 Ensuring Quality Constraints

In our setting, average quality (Section 6.3.1, point 7) is dependent on Q_{ij}, which is stochastic in nature. It is more natural to work with expected terms in such a stochastic framework than on a sequence of realized values. Towards this, we show that by ensuring our quality constraint on expected average quality, q_{av}, instead, we can still achieve approximate constraint satisfaction with a high probability. Formally, we present the following lemma,

Lemma 11. *The probability that average quality is less than $\alpha - \epsilon$ given that $q_{av} \geq \alpha$, can be bounded as follows:*

$$\mathcal{P}\left(\frac{\sum_{i\in[n]}\sum_{j=1}^{x_i} Q_{ij}}{\sum_{i\in[n]} x_i} < \alpha - \epsilon \mid q_{av} \geq \alpha\right) \leq \exp\left(-2\epsilon^2 m\right)),$$

where $m = \sum_{i\in[n]} x_i$, and ϵ is a constant.

Proof. Let, $V = \frac{\sum_{i\in[n]}\sum_{j=1}^{x_i} Q_{ij}}{\sum_{i\in[n]} x_i}$

$$E[V] = \frac{\sum_{i\in[n]}\sum_{j=1}^{x_i} E[Q_{ij}]}{\sum_{i\in[n]} x_i} = \frac{\sum_{i\in[n]} q_i x_i}{\sum_{i\in[n]} x_i} = q_{av}$$

[1] We assume expected revenue to be proportional to the quality of the product. It is a reasonable assumption as if q_i is the probability of the product being sold and R is the price of the product, its expected revenue would be Rq_i

Therefore,

$$\mathcal{P}\big(V < \alpha - \epsilon \mid E[V] \geq \alpha\big) \leq \mathcal{P}\big(V < E[V] - \epsilon\big)$$
$$= \mathcal{P}\big(V - E[V] < -\epsilon\big) \leq exp(-2\epsilon^2 m)$$

The last line follows from Hoeffding's inequality [65]. □

From the above lemma, we show that by ensuring $q_{av} \geq \alpha$, we can achieve probably approximate correct (PAC) results on our constraint. Hence, for the rest of the chapter, we work with $q_{av} \geq \alpha$ as our quality constraint (QC).

6.3.3 Integer Linear Program (ILP)

When the qualities of the agents are known, the planner's subset selection problem can be formulated as an ILP where it needs to decide on the number of units, x_i, to procure from each agent i so as to maximize its utility (objective function) while ensuring the quality and capacity constraints. The optimization problem can be described as follows:

$$\begin{aligned} \max_{x_i} \quad & \sum_{i \in [n]} (Rq_i - c_i)x_i \\ \text{s.t. } & q_{av} = \frac{\sum_{i \in [n]} q_i x_i}{\sum_{i \in [n]} x_i} \\ & q_{av} \geq \alpha \\ & 0 \leq x_i \leq k_i \qquad \forall i \in [n] \\ & x_i \in \mathbb{Z} \qquad \forall i \in [n] \end{aligned} \tag{6.1}$$

6.3.4 Dynamic Programming Based Subset Selection (DPSS)

In order to solve the ILP, we propose a dynamic programming based algorithm called DPSS. For ease of exposition, we consider $k_i = 1$, i.e., each agent has a unit capacity of production. This is a reasonable assumption that does not change our algorithm's results. For an agent with $k_i > 1$, we can consider each unit as a separate agent, and the proofs and discussion henceforth follows.

Formally, the algorithm proceeds as follows:

1. Divide the agents into one of the four categories:
   (a) S_1: Agents with $q_i \geq \alpha$ and $r_i \geq 0$
   (b) S_2: Agents with $q_i < \alpha$ and $r_i \geq 0$
   (c) S_3: Agents with $q_i \geq \alpha$ and $r_i < 0$
   (d) S_4: Agents with $q_i < \alpha$ and $r_i < 0$

2. Let $\boldsymbol{x} = \{x_i\}_{i\in[n]}$ be the selection vector, where $x_i = 1$ if the i^{th} agent is selected and 0 otherwise.

3. Since an agent in S_1 has a positive utility and above threshold quality, $x_i = 1$, $\forall i \in S_1$. Let $d = \sum_{i\in S_1}(q_i - \alpha)$ be the excess quality accumulated.

4. Similarly, all units in S_4 have a negative utility and below threshold quality. Hence, $x_i = 0$, $\forall i \in S_4$.

5. Let G be the set of the remaining agents (in S_2 and S_3). For each agent $i \in G$, we define $d_i = q_i - \alpha$. Thus, we need to select the agents $i \in G$ that maximizes the utility, such that $\sum_{i\in G} x_i d_i \leq d$.

6. For agents in G, select according to the DP function defined in Algorithm 18 (Lines [8-16]). Here, d^{te} denotes the access quality accumulated before choosing the next agent and x^{te} refers to the selections made so far in the DP formulation.

Algorithm 18 DPSS

1: **Inputs:** $[n]$, α, R, costs $\boldsymbol{c} = \{c_i\}_{i\in[n]}$, qualities $\boldsymbol{q} = \{q_i\}_{i\in[n]}$
2: **Output:** Quantities procured $\boldsymbol{x} = (x_1, \ldots, x_n)$
3: **Initialization:** $\forall i \in [n]$, $r_i = Rq_i - c_i$, $z = 0$
4: Segregate S_1,S_2,S_3,S_4 as described in Section 4.1
5: $\forall i \in S_1$, $x_i = 1$; $z = z + r_i$; $d = \sum_{i\in S_1}(q_i - \alpha)$
6: $\forall i \in S_4$, $x_i = 0$
7: $G = S_2 \cup S_3$; $\forall i \in G, d_i = q_i - \alpha$
8: **function** DP($i, d^{te}, x^{te}, x^\star, z^{te}, z^\star$)
9: **if** $i == |G|$ and $d^{te} < 0$ **then return** $x^\star, z^\star$
10: **if** $i == |G|$ and $d^{te} \geq 0$ **then**
11: **if** $z^{te} > z^\star$ **then**
12: $z^\star = z^{te}$; $x^\star = x^{te}$
13: **return** $x^\star, z^\star$
14: $x^\star, z^\star = DP(i+1, d^{te}, [x^{te}, 0], x^\star, z^{te}, z^\star)$
15: $x^\star, z^\star = DP(i+1, d^{te} + d_i, [x^{te}, 1], x^\star, z^{te} + r_i, z^\star)$
16: **return** $x^\star, z^\star$
17: x^G, z^G = DP(0,d,[ ],[ ],0,0)
18: $\forall i \in G, x_i = x_i^G$
19: **return** x

6.4 Subset Selection with Unknown Qualities of Agents

In the previous section, we assumed that the agents' qualities, q_i, are known to *C*. We now consider a setting when q_i are *unknown* beforehand and can only be learned by selecting the agents. We model it as a CMAB problem with semi-bandit feedback and QC.

6.4.1 Additional Notations

We introduce the additional notations to model our problem. Similar to our previous setting, we assume that we are given a fixed set of agents, $[n]$, each with its own average quality of produce, q_i and cost of produce, c_i. Additionally, our algorithm proceeds in discrete rounds $t = 1, \ldots, T$. For a round t:

- Let $\boldsymbol{x}^t \in \{0,1\}^n$ be the selection vector at round t, where $x_i^t = 1$ if the agent i is selected in round t and $x_i^t = 0$ if not.
- The algorithm selects a subset of agents, $S^t \subseteq [n]$, referred to as a super-arm henceforth, where $S^t = \{i \in [n] | x_i^t = 1\}$. Let s^t be cardinality of selected super-arm, i.e., $s^t = |S^t|$.
- Let w_i^t denote the number of rounds an agent i has been selected until round t, i.e., $w_i^t = \sum_{y \leq t} x_i^y$.
- For each agent $i \in S^t$, the planner, C, observes its realized quality X_i^j, where $j = w_i^t$ and E[X_i^j] = q_i. For an agent $i \notin S_t$, we do not observe its realized quality (semi-bandit setting).
- The empirical mean estimate of q_i at round t, is denoted by $\hat{q}_i^t = \frac{1}{w_i^t} \sum_{j=1}^{w_i^t} X_i^j$. The upper confidence bound (UCB) estimate is denoted by $(\hat{q}_i^t)^+ = \hat{q}_i^t + \sqrt{\frac{3 \ln t}{2 w_i^t}}$.
- Utility to C at round t is given by: $r_{\boldsymbol{q}}(S^t) = \sum_{i \in S^t} R q_i - c_i$, where $\boldsymbol{q} = \{q_1, q_2, \ldots, q_n\}$ is the quality vector.
- The expected average quality of selected super-arm at round t is given by: $q_{av}^t = \frac{1}{s^t} \sum_{i \in S^t} q_i$.

Following from Lemma 11, we continue to work with expected average quality instead of realized average quality.

6.4.2 SS-UCB

In this section, we propose an abstract framework, SS-UCB, for subset selection problem with quality constraint. SS-UCB assumes that there exist an offline subset selection algorithm, SSA, (e.g., DPSS), which takes a vector of qualities, $\boldsymbol{q'}$, and costs, $\boldsymbol{c'}$, along with the target quality threshold, α', and proportionality constant, R, as an input and returns a super-arm which satisfies the quality constraint (QC) with respect to $\boldsymbol{q'}$ and α'.

SS-UCB runs in two phases: (i) Exploration: where all the agents are explored for certain threshold number of rounds, τ ; (ii) Explore-exploit: We invoke SSA (line 10, Algorithm 19) with $\{(\hat{q}_i^t)^+\}_{i \in N}$, $\{c_i\}_{i \in N}$, $\alpha + \epsilon_2$ and R as the input parameters and select accordingly. We invoke SSA with a slightly higher target threshold, $\alpha + \epsilon_2$ so that our algorithm is more conservative while selecting the super-arm in order to ensure QC with a high probability (discussed in Section 6.4.3). As we shall see in Section 6.4.3, the higher the value of ϵ_2, the sooner the SSA satisfies QC with a high probability, but it comes

with the cost of loss in utility. Thus, the value of ϵ_2 must be appropriately selected based on the planner's preferences.

We refer to the algorithm as DPSS-UCB when we use DPSS (Algorithm 18) as SSA in the SS-UCB framework. We show that DPSS-UCB outputs the super-arm that satisfies the QC with high probability (w.h.p) after a certain threshold number of rounds, τ, and incurs a regret of $O(\ln T)$.

Algorithm 19 SS-UCB

Inputs: $[n]$, α, ϵ_2, R, costs $\boldsymbol{c} = \{c_i\}_{i\in[n]}$
For each agent i, maintain: w_i^t, q_i^t, $(\hat{q}_i^t)^+$
$\tau \leftarrow \frac{3\ln T}{2\epsilon_2^2}$; $t = 0$
while $t \leq \tau$ (**Explore Phase**) **do**
    Play a super-arm $S^t = [n]$
    Observe qualities $X_i^j, \forall i \in S^t$ and update w_i^t, $\hat{q}_i^t$
    $t \leftarrow t + 1$
while $t \leq T$ (**Explore-Exploit Phase**) **do**
    For each agent i, set $(\hat{q}_i^t)^+ = \hat{q}_i^t + \sqrt{\frac{3\ln t}{2w_i^t}}$
    S^t = SSA ($\{(\hat{q}_i^t)^+\}_{i\in N}, c, \alpha + \epsilon_2$,R)
    Observe qualities $X_i^j, \forall i \in S^t$ and update w_i^t, $\hat{q}_i^t$
    $t \leftarrow t + 1$

6.4.3 Ensuring Quality Constraints

We provide Probably Approximate Correct (PAC) [49, 62] bounds on DPSS-UCB satisfying QC after τ rounds:

Theorem 23. *For $\tau = \frac{3\ln T}{2\epsilon_2^2}$, if each agent is explored τ number of rounds, then if we invoke DPSS with target threshold $\alpha + \epsilon_2$ and $\{(\hat{q}_i^t)^+\}_{i\in N}$ as the input, the QC is approximately met with high probability.*

$$\mathcal{P}\left(q_{av}^t < \alpha - \epsilon_1 \mid \frac{1}{s^t}\sum_{i\in S^t}(\hat{q}_i^t)^+ \geq \alpha + \epsilon_2, t > \tau\right) \leq exp(-\epsilon_1^2 t).$$

where ϵ_1 is the tolerance parameter and refers to the planner's ability to tolerate a slighty lower average quality than required.

Henceforth, a super-arm will be called *correct* if it satisfies the QC approximately as described above.

Proof. The proof is divided into two parts. Firstly, we show that for each $t > \tau$ round, the average value of $(\hat{q}_i^t)^+$ and that of $\hat{q}_i^t$ of the agents i in selected super-arm S^t is less than ϵ_2. Secondly, we show that if the average of $\hat{q}_i^t$ is guaranteed to be above the threshold, then the average of q_i over the selected agents would not be less than $\alpha - \epsilon_1$ with a high probability.

Lemma 12. *The difference between the average of $(\hat{q}_i^t)^+$ and the average of $\hat{q}_i^t$ over the agents i in S^t is less than ϵ_2, $\forall t > \tau$.*

Proof. We have,

$$\frac{1}{s^t}\sum_{i\in S^t}\left((\hat{q}_i^t)^+ - \hat{q}_i^t\right) = \frac{1}{s^t}\sum_{i\in S^t}\frac{\sqrt{3\ln t}}{\sqrt{2w_i^t}} \leq \frac{\sqrt{3\ln t}}{\sqrt{2w_{min}^t}}.$$

where $w_{min}^t = \min_i w_i^t$. Since, for $t < \tau$, we are exploring all the agents, thus, $w_i^\tau = \tau$. Now, since $w_i^t \geq w_i^\tau, \forall t > \tau$, thus, we claim that $w_{min}^t \geq \tau$ for $t > \tau$. Hence,

$$\frac{\sqrt{3\ln t}}{\sqrt{2w_{min}^t}} \leq \frac{\sqrt{3\ln T}}{\sqrt{2\tau}}.$$

For $\tau = \frac{3\ln T}{2\epsilon_2^2}$, we have,

$$\frac{1}{s^t}\sum_{i\in S^t}\left((\hat{q}_i^t)^+ - \hat{q}_i^t\right) \leq \epsilon_2.$$

□

Lemma 13. $\forall t > \tau$

$$\mathcal{P}\left(\, q_{av}^t < \alpha - \epsilon_1 \mid \frac{1}{s^t}\Big(\sum_{i\in S^t}\left(\hat{q}_i^t\right) \geq \alpha\Big)\,\right) \leq exp(-\epsilon_1^2 t).$$

Proof. Let $Y^t = \frac{1}{s^t}\sum_{i\in S^t}\hat{q}_i^t$. Since E[$\hat{q}_i^t$] = E[$X_i^j$] = q_i, E[Y^t] = q_{av}^t. Hence, we have,

$$\begin{aligned}\mathcal{P}(E[Y^t] < \alpha - \epsilon_1) \mid Y^t \geq \alpha) &\leq \mathcal{P}\big(Y^t \geq E[Y^t] + \epsilon_1\big) \\ &\leq exp(-\epsilon_1^2 w^t).\end{aligned}$$

where $w^t = \sum_{i\in S^t} w_i^t$, i.e., total number of agents selected till round t. Since we pull atleast one agent in each round, we can say that, $w^t \geq t$. Thus, $\forall t > \tau$

$$\mathcal{P}\left(\, q_{av}^t < \alpha - \epsilon_1 \mid \frac{1}{s^t}\left(\sum_{i\in S^t}\left(\hat{q}_i^t\right) \geq \alpha\right)\,\right) \leq exp(-\epsilon_1^2 t).$$

□

From Lemma 12 and Lemma 13, the proof follows. □

6.4.4 Regret Analysis of DPSS-UCB

In this section, we propose the regret definition for our problem setting that encapsulates the QC. We then upper bound the regret incurred by DPSS-UCB to be of the order $O(\ln T)$.

We define regret incurred by an algorithm A on round t as follows:

$$Reg^t(A) = \begin{cases} (r_{\boldsymbol{q}}(S^\star) - r_{\boldsymbol{q}}(S^t)) & \text{if } S^t \text{ satisfies QC} \\ L & \text{otherwise.} \end{cases}$$

where $S^\star = argmax_{S \in S_f} r_{\boldsymbol{q}}(S)$ and $L = \max_{S \in S_f}(r_{\boldsymbol{q}}(S^\star) - r_{\boldsymbol{q}}(S))$ is some constant. Here, S_f are the feasible subsets which satisfies QC; $S_f = \{S | S \subseteq N and \frac{\sum_{i \in s} x_i q_i}{\sum_{i \in s} x_i} \geq q_{av}\}$.
Hence, the cumulative regret in T rounds incurred by the algorithm is:

$$Reg(A) = \sum_{t=1}^{T} Reg^t(A). \tag{6.2}$$

We now analyse the regret when the algorithm, A, is DPSS-UCB.

$$\begin{aligned} Reg(A) &= \sum_{t=1}^{\tau} Reg^t(A) + \sum_{t=\tau+1}^{T} Reg^t(A) \\ &\leq L \cdot \tau + \sum_{t=\tau+1}^{T} Reg^t(A) \\ &\leq \frac{L \cdot 3 \ln T}{2\epsilon_2^2} + \sum_{t=\tau+1}^{T} Reg^t(A). \end{aligned}$$

Since our algorithm ensures that S^t satisfies the approximate QC for $t > \tau$ with a probability greater than $1 - \sigma$, where $\sigma = exp(-\epsilon_1^2 t)$, we have,

$$\mathbb{E}[Reg(A)] \leq \frac{L \cdot 3 \log T}{2\epsilon_2^2} + \left(\underbrace{\sum_{t \geq \tau} \left[(1-\sigma)(r_{\boldsymbol{q}}(S^\star) - r_{\boldsymbol{q}}(S^t)) \right]}_{Reg_u(T)} + \sigma L \right). \tag{6.3}$$

where $S^t \in S_f$.

Now,

$$\begin{aligned} \sum_{t \geq \tau} \sigma L = \sum_{t \geq \tau} L e^{(-\epsilon_1^2 t)} &\leq \frac{L e^{(-\epsilon_1^2 \tau)}}{1 - e^{(-\epsilon_1^2)}} \\ &\sim O\left(\frac{1}{T^a}\right), \text{where } a = \frac{3\epsilon_1^2}{2\epsilon_2^2}. \end{aligned}$$

Now we bound the cumulative regret incurred after $t > \tau$ rounds when QC is satisfied, i.e., $Reg_u(T)$. Here we adapt the regret proof given by [41]. We highlight our setting's similarities and differences and use it to bound $Reg_u(T)$.

Bounding $Reg_u(T)$:
[41] have proposed CUCB algorithm to tackle CMAB problem which they prove to have an upper bound regret of $O(\ln T)$. Following is the CMAB problem setting considered in [41]:

- There exists a constrained set of super-arms $\chi \subseteq 2^{[n]}$ available for selection.
- There exists an offline (η, ν)-approximation oracle, $(\eta, \nu \leq 1)$ s.t. for a given quality vector $\boldsymbol{q}'$ as input, it outputs a super-arm, S, such $\mathcal{P}(r_{\boldsymbol{q}'}(S) \geq \eta \cdot opt_{\boldsymbol{q}'}) \geq \nu$, where $opt_{\boldsymbol{q}'}$ is the optimal reward for quality vector $\boldsymbol{q}'$ as input.
- Their regret bounds hold for any reward function that follows the properties of monotonicity and bounded smoothness (defined below).
- Similar to our setting, they assume a semi-bandit feedback mechanism.

Now, we state the reasons to adopt the regret analysis provided by [41] to bound $Reg_u(T)$

1. We have shown that after τ rounds, we get the constrained set of super-arms, χ, i.e., the set of super-arms that satisfies QC), which forms a well defined constrained set, to select from in future rounds ($t > \tau$).
2. We remark here that the utility function considered in our problem setting follows both the required properties, namely,
   (i) *Monotonicity:* The expected reward of playing any super-arm $S \in \chi$ is monotonically non-decreasing with respect to the quality vector, i.e., let $\boldsymbol{q}$ and $\overline{\boldsymbol{q}}$ be two quality vectors such that $\forall i \in [n]$, $q_i \leq \overline{q}_i$, we have $r_{\boldsymbol{q}}(S) \leq r_{\overline{\boldsymbol{q}}}(S)$ for all $S \in \chi$. Since our reward function is linear, it is trivial to note that it is monotone on qualities.
   (ii) *Bounded Smoothness:* There exists a strictly increasing (and thus invertible) function $f(.)$, called bounded smoothness function, such that for any two quality vectors $\boldsymbol{q}$ and $\overline{\boldsymbol{q}}$, we have $r_{\boldsymbol{q}}(S) - r_{\overline{\boldsymbol{q}}}(S) \leq$ f(Λ) if $\max_{i \in S}\ q_i$ - $\overline{q}_i \leq \Lambda$. As our reward function is linear in qualities, $f(\Lambda) = nR \times \Lambda$ is the bounded smoothness function for our setting, where n is the number of agents.
3. *Oracle*: Analogous to the oracle assumption in [41], we have assumed the existence of an algorithm SSA (Section 6.4.2). For DPSS-UCB, we use DPSS (Algorithm 18) as our SSA . As DPSS provides exact solution, it acts as an (η, ν)- approximate oracle for DPSS-UCB with $\eta = 1 = \nu$.

However, to ensure χ consists of all the correct super-arms, we need one additional property that should be satisfied, namely ϵ-seperatedness property.

Definition 14. *We say $\boldsymbol{q} = (q_1, q_2, \ldots, q_n)$ satisfies ϵ-seperatedness if $\forall S \subseteq [n]$, $U(S) = \frac{1}{s}\sum_{i \in S} q_i$ s.t. $U(S) \notin (\alpha - \epsilon, \alpha)$*

This suggests that there is no super-arm $S \in \chi$, such that $\alpha - \epsilon \leq \frac{1}{|S|}\sum_{i \in S} q_i^t \leq \alpha$. It is important for DPSS-UCB to satisfy ϵ_1-seperatedness because if there exists such a super-arm, for which the average quality is between ($\alpha - \epsilon_1$, α), DPSS-UCB will include it in χ due to tolerance parameter ϵ_1 while it would violate the QC.

Theorem 24. *If qualities of the agents satisfy ϵ_1-seperatedness, then $Reg_u(T)$ is bounded by $O(\ln T)$.*

Proof. Following from the proof in [41], we define some parameters. A super-arm, S is bad if $r_{\boldsymbol{q}}(S) < opt_{\boldsymbol{q}}$. Define S_B as the set of bad super-arms. For a given underlying agent $i \in [n]$, define:

$$\Delta^i_{\min} = \text{opt}_{\boldsymbol{q}} - \max\{r_{\boldsymbol{q}}(S) | S \in S_B, i \in S\}$$
$$\Delta^i_{\max} = \text{opt}_{\boldsymbol{q}} - \min\{r_{\boldsymbol{q}}(S) | S \in S_B, i \in S\}.$$

Using the same proof as in [41], we can show that, V_T, the expected number of times we play a sub-optimal agent till round T, is upper bounded as:

$$\begin{aligned} V_T &\leq n(l_T) + \sum_{t=\tau}^{T} \frac{2n}{t^2} \leq n(l_T) + \sum_{t=1}^{T} \frac{2n}{t^2} \\ &\leq \frac{6n \cdot \ln T}{(f^{-1}(\Delta_{\min}))^2} + \left(\frac{\pi^2}{3}\right) \cdot n. \end{aligned}$$

where $l_T = \frac{6 \ln T}{(f^{-1}(\Delta_{\min}))^2}$. Hence, we can bound the regret as:-

$$\begin{aligned} Reg_u(T) &\leq V_T \cdot \Delta_{\max} \leq \left(\frac{6 \cdot \ln T}{(f^{-1}(\Delta_{\min}))^2} + \frac{\pi^2}{3}\right) n \cdot \Delta_{\max} \\ &= \left(\frac{6 \cdot \ln T}{(\frac{\Delta_{\min}}{R})^2} + \frac{\pi^2}{3}\right) n \cdot \Delta_{\max}. \end{aligned}$$

□

Substituting the results of Theorem 24 in Equation 6.3, we prove that DPSS-UCB incurs a regret of $O(\ln T)$.

6.5 Greedy Approach

In the previous sections, we propose a framework and dynamic programming based algorithm to solve our subset selection problem for both when the agents' quality is known and not. Since DPSS explores all the possible combinations of the selected vector and the utility associated with it, the complexity of DPSS is of $O(2^n)$, which makes it difficult to scale when n is large.

To overcome this limitation, we propose a greedy-based approach to our problem. When the quality of agents are known, we propose GSS that runs in polynomial time, $O(n \log n)$, and provides an approximate solution to our ILP. Then, we use GSS as our SSA in the SS-UCB framework and propose GSS-UCB as an alternate algorithm to DPSS-UCB in the setting where the agents' qualities are unknown.

6.5.1 Greedy Subset Selection (GSS)

Greedy algorithms have been proven to provide approximate solutions to ILP problems such as 0-1 knapsack. They solve linearly relaxed variants of an ILP, such as fractional knapsack, and remove any fractional unit from its solution. We propose a similar algorithm for our subset selection problem by allowing $x_i \in [0, 1]$. However, we cannot simply remove fractional units from our solution, leading to QC violation. Consider the following example:

Given $n = 2$ agents with qualities, $\boldsymbol{q} = [0.6, 0.9]$, $\boldsymbol{c} = [10, 100]$ and $\alpha = 0.7$. Allowing fractional units to be taken, the optimal solution would be to take $x_1 = 1, x_2 = 0.5$ units of the two agents. Removing fractional units would lead to selecting only the first agent, which violates the QC. Towards this, we include an additional step (Line [22], Algorithm 20) in our algorithm that ensures that QC is not violated. Formally, the algorithm proceeds as follows:

1. Divide the agents into the four categories, namely, S_1, S_2, S_3, S_4, as described in Section 6.3.4.

2. Select all agents in S_1. Let $d = \sum_{i \in S_1}(q_i - \alpha)$ be the excess quality accumulated and as before, drop all agents in S_4.

3. For agents in S_2, sort them in the decreasing order of revenue gained per unit loss in quality ($\frac{r_i}{\alpha - q_i}$). Similarly, for agents in S_3, sort them in the increasing order of revenue lost per unit gain in quality ($\frac{r_i}{\alpha - q_i}$).

4. Select units (could be fractional) from agents from S_2 until the total loss of quality is no more than d. Essentially, we use the agents in S_2 to increase revenue while ensuring average quality is above the threshold.

5. For agents in S_2 with remaining fractional units, we pair them up with an equivalent fractional unit of an agent in S_3 that balances the loss in average quality.

6. When the revenue gained per unit loss in quality from the first non-exhausted agent in S_2 is less than the revenue lost per unit gain of quality from the first non-exhausted agent in S_3, terminate the algorithm. An agent is exhausted if the unit produce is completely selected.

7. For any agent in S_3 with a fractional unit, take the complete unit instead. For all other agents, remove any fractional units selected.

6.5.2 Approximation Ratio

While GSS is computationally more efficient than DPSS, it is essential to note that it may not always return the optimal subset of agents. We show for the following example, that GSS does not have a constant approximation w.r.t. the optimal solution:

Algorithm 20 GSS

Inputs: N, α, R, costs $\boldsymbol{c} = [c_i]$, qualities $\boldsymbol{q} = [q_i]$
Output: Quantities procured $\boldsymbol{x} = (x_1, \ldots, x_n)$
Initialization: $\forall i \in N$, $r_i = Rq_i - c_i$
Segregate S_1,S_2,S_3,S_4 as described in Section 6.3.4
$\forall i \in S_1$, $x_i = 1$; $d = \sum_{i \in S_1}(q_i - \alpha)$
$\forall i \in S_4$, $x_i = 0$
$L_2 = sort(S_2)$ on decreasing order of $\frac{r_i}{\alpha - q_i}$
$L_3 = sort(S_3)$ on increasing order of $\frac{r_i}{\alpha - q_i}$
$p = 0, q = 0$
while $d > 0$ and $p < |S_2|$ **do**
    $i = L_2[p]$;
    if $\alpha - q_i \leq d$ **then** $x_i = 1$, $d = d - \alpha - q_i$, $p+ = 1$
    else $x_i = \frac{d}{\alpha - q_i}$, $d = 0$
while $p < |S_2|$ and $q < |S_3|$ **do**
    $i = L_2[p]$, $j = L_3[q]$
    $a = \frac{r_i}{\alpha - q_i}$, $b = \frac{r_j}{\alpha - q_j}$
    if $a \leq b$ **then break**;
    $w_1 = \min((1 - x_i)(\alpha - q_i), (1 - x_j)(q_j - \alpha))$
    $x_i+ = \frac{w_1}{\alpha - q_i}$, $x_j+ = \frac{w_1}{q_j - \alpha}$
    if $x_i == 0$ **then** $p++$;
    if $x_j == 0$ **then** $q++$;
if $0 < x_j < 1$ **then** $x_j = 1$
return $\lfloor \boldsymbol{x} \rfloor$

Consider $n = 3$ agents with qualities, $\boldsymbol{q} = [1.00, 0.98, 0.97]$ and $\boldsymbol{c} = [R - \epsilon, \frac{78R}{100}, \frac{47R}{100}]$. Hence, $\boldsymbol{r} = [\epsilon, \frac{R}{5}, \frac{R}{2}]$, where R is some constant as discussed before such that $r_i = Rq_i - c_i$. If $\alpha = 0.99$, the value of $\frac{r_i}{\alpha - q_i}$ for the third agent is higher than that of the second, but only a fractional unit can pair with the first agent. Hence, according to GSS, we only select the first agent giving us a utility of ϵ, whereas the optimal utility is equal to $\epsilon + \frac{R}{5}$ corresponding to choosing the first and the third agent. Thus, the approximation ratio is $\frac{\epsilon}{\epsilon + \frac{R}{5}}$. Since ϵ can take an arbitrary small value, the approximation ratio between the utility achieved by GSS and DPSS can be arbitrary small.

However, through experiments, we show that in practice, GSS gives close to optimal solutions at a huge computational benefit that allows us to scale our framework for a large number of agents, such as in an E-commerce setting.

6.5.3 GSS-UCB

When we use GSS as the SSA in our SS-UCB framework, we refer to the algorithm as GSS-UCB. While the regret analysis may not necessarily hold, as GSS does not have a constant approximation, we still show that in practice, it works as good as DPSS-UCB in both (i) achieving constraint satisfaction after τ rounds and (ii) the regret incurred thereafter. We show this via experiments, as discussed in Section 6.6.

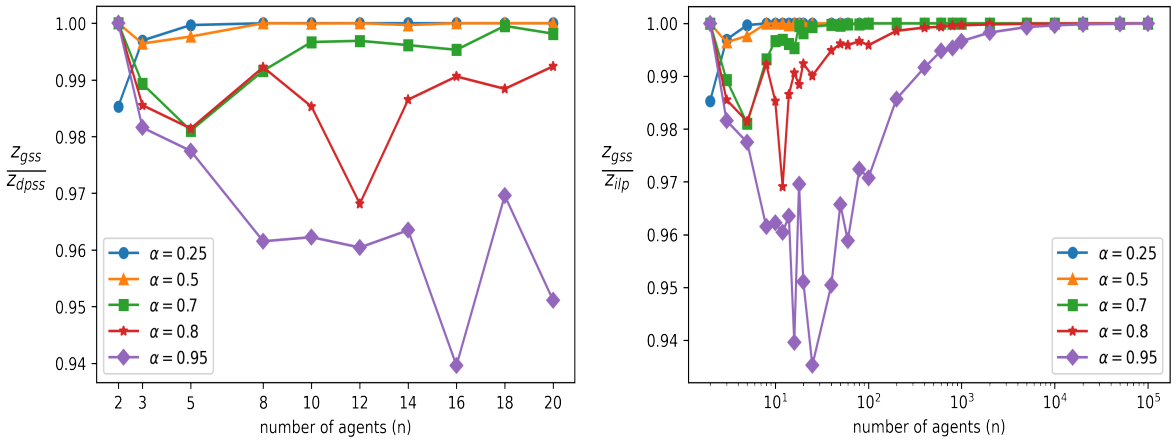
6.6 Experimental Analysis

6.6.1 Subset Selection With Known Qualities

In this section, we compare the performance of GSS with DPSS in the setting where quality of the agents is known. In Figure 6.1a, we compare the ratio of the utility achieved by GSS (z_{gss}) to the utility achieved by DPSS (z_{dpss}) while ensuring the QC is met. In Figure 6.2, we present a box plot of the distribution of the ratios of these utilities over 1000 iterations for $\alpha = 0.7$. To compare the performance of GSS for much larger values of n, we compare it against the utility achieved by an ILP solver (z_{ilp}), namely, the COIN-OR Branch and Cut Solver (CBC) [50] since the computational limitations of DPSS made it infeasible to run experiments for large values of n. The results for the same are presented in Figure 6.1b. Lastly, in Table 6.1, we compare the ratio of the time taken by GSS (t_{gss}) with respect to DPSS (t_{dpss}) and the ILP solver (t_{ilp}) for different values of n with α being set to 0.7.

6.6.1.1 Setup

For different values of n, the number of agents and α, the quality threshold, we generate agents with q_i and c_i both $\sim U[0, 1]$. For Figure 6.1a and 6.1b, we average our results over 1000 iterations for each (n, α) pair, while in Figure 6.2, we plot the distribution of the ratios obtained in each of the 1000 iterations for different values of n with α set to 0.7. We use $R = 1$ for all our experiments.



(a) w.r.t DPSS

(b) w.r.t. ILP

Figure 6.1: Performance of GSS on different values of α

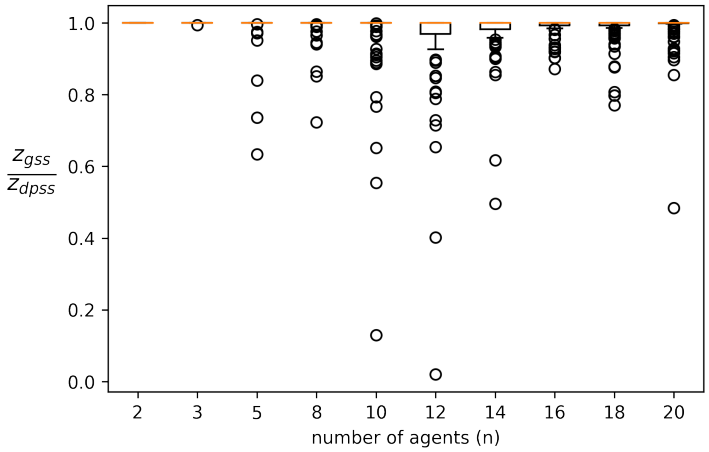


Figure 6.2: GSS vs DPSS ratio distribution

6.6.1.2 Results and Discussion

As can be seen from Figures 6.1a and 6.1b, the average ratio of both ($\frac{z_{gss}}{z_{dpss}}$) and ($\frac{z_{gss}}{z_{ilp}}$) lies approximately between [0.94,1.0], with a median of 1.0 for almost all values of n and only a few outliers and a few rare instances when the ratio drops below 0.2 as evident from Figure 6.2; this indicates that GSS performs almost as well as DPSS in practice with an exponentially improving computational performance in terms of time complexity with respect to DPSS and an almost 50x improvement over the ILP solver as well. This establishes the efficacy of GSS for practical use at scale.

n	$t_{dpss} : t_{gss}$	$t_{ilp} : t_{gss}$
2	5.5	70
5	15.7	64
8	32.6	63.7
10	54.3	58.6
12	106.3	67.6
14	284.4	65.3
16	897.1	60.2
18	3109.7	63.1
20	11360.6	68.1

n	$t_{ilp} : t_{gss}$
25	66.7
50	58.3
100	52.7
400	43.1
1000	31.8
5000	31.6
10000	34.5
50000	45
100000	56.8

Table 6.1: Computational Performance of GSS w.r.t. to DPSS and ILP

6.6.2 Subset Selection With Unknown Qualities

In this section, we present experimental results of DPSS-UCB and GSS-UCB towards the following:

1. Constraint Satisfaction: As discussed in section 6.4.3, DPSS-UCB satisfies the QC approximately with high probability after $\tau = \frac{3 \ln T}{2\epsilon_2^2}$ rounds. Here, $\alpha + \epsilon_2$ is the target constraint of the agent

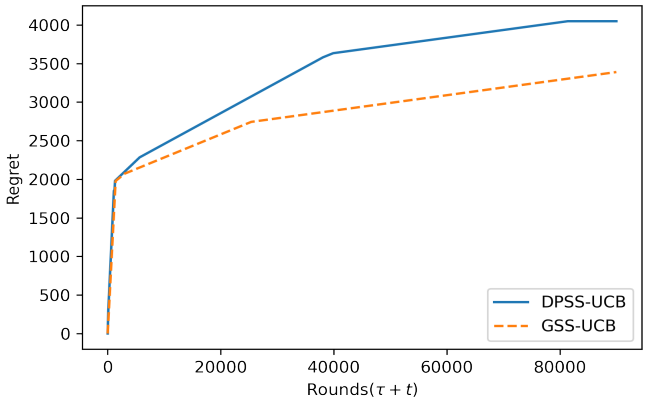


Figure 6.3: Regret incurred for $t > \tau$

when α is the required average quality threshold. Towards this, we plot the average number of iterations where DPSS-UCB and GSS-UCB returns a subset that satisfies QC at each round in our experiment for different values of ϵ_2.

2. Regret incurred for $t > \tau$: We show that the regret incurred by our algorithm for $t > \tau$, follows a curve upper bounded by $O(\ln T)$. Towards this we plot the cumulative regret vs. round t, where $\tau < t \leq T$.

6.6.2.1 Setup

To carry out these experiments, we generated $n = 10$ agents with both q_i, $c_i \sim U[0, 1]$. We chose $\alpha = 0.7$ as for a higher value of α the number of super-arms satisfying QC is very low and hardly much to learn whereas for a low value, the number of super-arms that satisfy QC is very high but practically of not much interest. In Figure 6.4, we perform the experiment over a varied range of values of ϵ_2, whereas in Figure 6.3, we set $\epsilon_2 = 0.01$. We average our results for 1000 iterations of each experiment. For example, in Figure 6.4, a value of 0.4 at some round t, would denote that in 40% of the iterations, the QC was satisfied at round t. For both the experiments, $R = 1$ and $T = 100000$.

6.6.2.2 Discussion

Higher the value of ϵ_2, higher is the target constraint and thus more conservative is our algorithm in selecting the subset of agents. Therefore, we achieve correctness quickly, which is evident from Figure 6.4. In all three cases, the algorithm achieves correctness in close to 100% of the iterations, after $\frac{3 \ln T}{2\epsilon_2^2}$ rounds (indicated by the vertical dotted line), which justifies our value of τ. Similarly, the regret incurred by DPSS-UCB for $t > \tau$ follows a curve upper bounded by $O(\ln T)$. The regret incurred by GPSS-UCB is slightly lower than DPSS-UCB which further establishes the efficacy of our greedy approach.

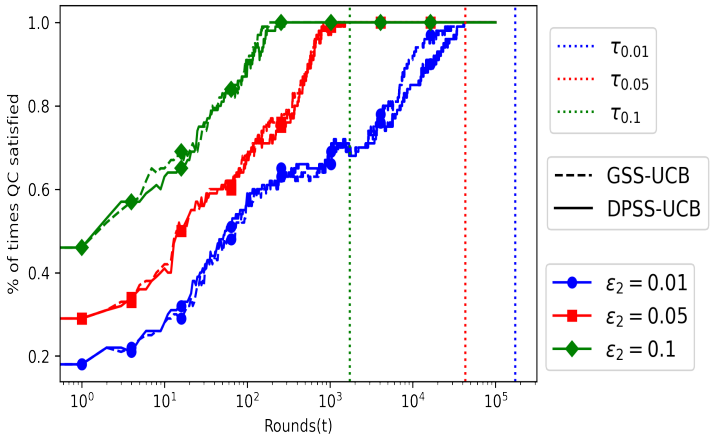


Figure 6.4: Constraint Satisfaction at each round

6.7 Conclusion and Future Work

In this chapter, we addressed the class of problems where a central planner had to select a subset of agents that maximized its utility while ensuring a quality constraint. We first considered the setting where the agents' quality is known and proposed DPSS that provided an exact solution to our problem. When the qualities were unknown, we modeled our problem as a CMAB problem with semi-bandit feedback. We proposed SS-UCB as a framework to address this problem. Both the constraint and the objective function depend on the unknown parameter, a setting not considered previously in the literature. Using DPSS as our SSA in SS-UCB, we proposed DPSS-UCB that incurred a $O(\ln T)$ regret and achieved constraint satisfaction with high probability after $\tau = O(\ln T)$ rounds. To address the computational limitations of DPSS, we proposed GSS for our problem that allowed us to scale our framework to a large number of agents. Through simulations, we showed the efficacy of GSS.

The SS-UCB framework proposed in this chapter can be used to design and compare other approaches to this class of problems that find its applications in many fields. We believe the result can be extended to solve for other interesting variants of the problem such as: (i) where the pool of agents to choose from is dynamic with new agents entering the setting, (ii) where an agent selected in a particular round is not available for the next few rounds (sleeping bandits) possibly due to lead time in procuring the units, a setting which is very common in operations research literature. Our work can also be extended to include strategic agents where the planner needs to design a mechanism to elicit the agents' production cost truthfully.

Chapter 7

Conclusion and Future Work

In this thesis, we dive into the rich and vibrant literature of MAB and address three specific problems inspired by real-world applications.

In the first problem, we address the problem in the intersection of Reinforcement Learning (MAB) and Game Theory (Mechanism design). We believe we are the first to design a non-exploration-separated ConMAB mechanism that is truthful and efficient in terms of regret. Though we have represented our work from the perspective of sponsored search auction, the solution can be applied to similar settings arising in other domains. We showed the theoretical bound on the performance of the mechanism and proved its truthful property. Through simulation, we show significant improvement in the performance of the proposed mechanism than the present state-of-the-art. This work can be lead to many extensions for example where the agents can change there valuations as well as bids, varying set of agents available for selection, non-stationary rewards etc.

In the second problem, we studied an interesting combination of sleeping and combinatorial bandits. We considered a more general reward setting conditioned that the reward function satisfies few mild smoothness properties. The generalization allows a large class of nonlinear reward functions. We considered two sets of assumptions, Lipschitz smoothness and Bounded smoothness. We provide theoretical guarantees under both setups. We validate the proven theoretical guarantees through experiments. In this work we have addressed some of the theoretical challenges arising in sleeping combinatorial setting, but still there are some open problems, for example the instance-independent regret guarantee under Bounded smoothness setting. This setup can be extended by considering other MAB setting, for instance, rotting bandits, where the arm pulling strategy may lead to dropping of the arms.

In the third problem, we addressed the class of problems where a central planner had to select a subset of agents that maximized its utility while ensuring a quality constraint. We propose algorithms to address the problem and show its correctness in terms of algorithm satisfying the constraints. We also provide a bound on its performance and show its efficacy through experiments. We believe many extensions are possible of our work like: at each round new agents enter in the setup, an agent selected in particular round is not available in next few rounds, the agents are strategic and may misreport there cost to maximize their utility etc.

Related Publications

- **Kumar Abhishek**, Shweta Jain and Sujit Gujar. Designing Truthful Contextual Multi-Armed Bandits based Sponsored Search Auctions. In Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2020. In review in Journal of Artificial Intelligence Research (JAIR).

- Ayush Deva, **Kumar Abhishek**, and Sujit Gujar. A Multi-Arm Bandit Approach To Subset Selection Under Constraints. In Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2021. Also accepted for presentation as a long talk at the 12^{th} Adaptive and Learning Agents (ALA) Workshop at AAMAS, 2021.

- **Kumar Abhishek**, Ganesh Ghalme, Sujit Gujar and Yadati Narahari. Sleeping Combinatorial Bandits. In International Joint Conference on Artificial Intelligence, IJCAI 2021. In review.

Bibliography

[1] Y. Abbasi-Yadkori, D. Pál, and C. Szepesvári. Improved algorithms for linear stochastic bandits. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, pages 2312–2320, USA, 2011. Curran Associates Inc.

[2] K. Abhishek, G. Ghalme, S. Gujar, and Y. Narahari. Sleeping combinatorial bandits. *arXiv preprint arXiv:2106.01624*, 2021.

[3] K. Abhishek, S. Jain, and S. Gujar. Designing truthful contextual multi-armed bandits based sponsored search auctions. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '20, page 1732–1734. International Foundation for Autonomous Agents and Multiagent Systems, 2020.

[4] K. Achilleas and S. Anastasios. Marketing aspects of quality assurance systems: The organic food sector case. *British Food Journal*, 110(8):829–839, 2008.

[5] D. Agarwal. Computational advertising: the linkedin way. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 1585–1586, 2013.

[6] D. Agarwal, B. Long, J. Traupman, D. Xin, and L. Zhang. Laser: A scalable response prediction platform for online advertising. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 173–182, 2014.

[7] G. Aggarwal, A. Goel, and R. Motwani. Truthful auctions for pricing search keywords. In *Proceedings of the 7th ACM Conference on Electronic Commerce*, EC '06, pages 1–7, New York, NY, USA, 2006. ACM.

[8] S. Agrawal, V. Avadhanula, V. Goyal, and A. Zeevi. Mnl-bandit: A dynamic learning approach to assortment selection. *Operations Research*, 67(5):1453–1485, 2019.

[9] S. Agrawal and N. R. Devanur. Bandits with concave rewards and convex knapsacks. In *Proceedings of the fifteenth ACM conference on Economics and computation*, pages 989–1006, 2014.

[10] S. Agrawal and N. Goyal. Analysis of thompson sampling for the multi-armed bandit problem. In *Conference on Learning Theory*, pages 39–1, 2012.

[11] F. Amat, A. Chandrashekar, T. Jebara, and J. Basilico. Artwork personalization at netflix. In *Proceedings of the 12th ACM conference on recommender systems*, pages 487–488, 2018.

[12] A. Anandkumar, N. Michael, A. K. Tang, and A. Swami. Distributed algorithms for learning and cognitive medium access with logarithmic regret. *IEEE Journal on Selected Areas in Communications*, 29(4):731–745, 2011.

[13] A. Archer and E. Tardos. Truthful mechanisms for one-parameter agents. In *Proceedings of the 42Nd IEEE Symposium on Foundations of Computer Science*, FOCS '01, pages 482–, Washington, DC, USA, 2001. IEEE Computer Society.

[14] J.-Y. Audibert and S. Bubeck. Minimax policies for adversarial and stochastic bandits. In *Proceedings of the 22nd Annual Conference on Learning Theory (COLT)*, January 2009.

[15] P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *J. Mach. Learn. Res.*, 3:397–422, Mar. 2003.

[16] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

[17] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, pages 235–256, 2002.

[18] E. Axell, G. Leus, E. G. Larsson, and H. V. Poor. Spectrum sensing for cognitive radio: State-of-the-art and recent advances. *IEEE signal processing magazine*, 29(3):101–116, 2012.

[19] M. Babaioff, R. D. Kleinberg, and A. Slivkins. Truthful mechanisms with implicit payment computation. In *Proceedings of the 11th ACM Conference on Electronic Commerce*, EC '10, page 43–52, New York, NY, USA, 2010. Association for Computing Machinery.

[20] M. Babaioff, R. D. Kleinberg, and A. Slivkins. Truthful mechanisms with implicit payment computation. *J. ACM*, 62(2):10:1–10:37, May 2015.

[21] M. Babaioff, Y. Sharma, and A. Slivkins. Characterizing truthful multi-armed bandit mechanisms: Extended abstract. In *Proceedings of the 10th ACM Conference on Electronic Commerce*, EC '09, pages 79–88, New York, NY, USA, 2009. ACM.

[22] A. Badanidiyuru, R. Kleinberg, and A. Slivkins. Bandits with knapsacks. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 207–216. IEEE, 2013.

[23] A. Badanidiyuru, J. Langford, and A. Slivkins. Resourceful contextual bandits. In *Conference on Learning Theory*, pages 1109–1134, 2014.

[24] D. Bergemann and J. Välimäki. The dynamic pivot mechanism. *Econometrica*, 78(2):771–789, 2010.

[25] S. Bhat, S. Jain, S. Gujar, and Y. Narahari. An optimal bidimensional multi-armed bandit auction for multi-unit procurement. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '15, page 1789–1790, Richland, SC, 2015. International Foundation for Autonomous Agents and Multiagent Systems.

[26] S. Bhat, S. Nath, S. Gujar, O. Zoeter, Y. Narahari, and C. Dance. A mechanism to optimally balance cost and quality of labeling tasks outsourced to strategic agents. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 917–924, 2014.

[27] A. Biswas, S. Jain, D. Mandal, and Y. Narahari. A truthful budget feasible multi-armed bandit mechanism for crowdsourcing time critical tasks. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 1101–1109, 2015.

[28] A. Biswas, S. Jain, D. Mandal, and Y. Narahari. A truthful budget feasible multi-armed bandit mechanism for crowdsourcing time critical tasks. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '15, pages 1101–1109, Richland, SC, 2015. International Foundation for Autonomous Agents and Multiagent Systems.

[29] Z. Bnaya, R. Puzis, R. Stern, and A. Felner. Volatile multi-armed bandits for guaranteed targeted social crawling. *AAAI (Late-Breaking Developments)*, 2(2.3):16–21, 2013.

[30] D. Bouneffouf, R. Laroche, T. Urvoy, R. Féraud, and R. Allesiardo. Contextual bandit for active learning: Active thompson sampling. In *International Conference on Neural Information Processing*, pages 405–412. Springer, 2014.

[31] D. Bouneffouf and I. Rish. A survey on practical applications of multi-armed and contextual bandits. *arXiv preprint arXiv:1904.10040*, 2019.

[32] D. Bouneffouf, I. Rish, G. A. Cecchi, and R. Féraud. Context attentive bandits: contextual bandit with restricted context. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 1468–1475, 2017.

[33] G. Bresler, G. H. Chen, and D. Shah. A latent source model for online collaborative filtering. In *Advances in Neural Information Processing Systems*, pages 3347–3355, 2014.

[34] S. Bubeck, N. Cesa-Bianchi, et al. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1):1–122, 2012.

[35] N. Cesa-Bianchi and G. Lugosi. Combinatorial bandits. *Journal of Computer and System Sciences*, pages 1404–1422, 2012.

[36] D. Chakrabarti, R. Kumar, F. Radlinski, and E. Upfal. Mortal multi-armed bandits. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 273–280. 2009.

[37] A. Chatterjee, G. Ghalme, S. Jain, R. Vaish, and Y. Narahari. Analysis of thompson sampling for stochastic sleeping bandits. In *Association for Uncertainty in Artificial Intelligence, UAI*, 2017.

[38] L. Chen, J. Xu, and Z. Lu. Contextual combinatorial multi-armed bandits with volatile arms and submodular reward. In *Advances in Neural Information Processing Systems*, pages 3247–3256, 2018.

[39] S. Chen, T. Lin, I. King, M. R. Lyu, and W. Chen. Combinatorial pure exploration of multi-armed bandits. In *Advances in Neural Information Processing Systems 27*, pages 379–387. 2014.

[40] W. Chen, W. Hu, F. Li, J. Li, Y. Liu, and P. Lu. Combinatorial multi-armed bandit with general reward functions. In *Advances in Neural Information Processing Systems*, pages 1659–1667, 2016.

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

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

[43] S.-C. Chow and M. Chang. Adaptive design methods in clinical trials–a review. *Orphanet journal of rare diseases*, 3(1):11, 2008.

[44] W. Chu, L. Li, L. Reyzin, and R. E. Schapire. Contextual bandits with linear payoff functions. *Journal of Machine Learning Research - Proceedings Track*, 15:208–214, 01 2011.

[45] R. Combes, M. S. Talebi Mazraeh Shahi, A. Proutiere, and m. lelarge. Combinatorial bandits revisited. In *Advances in Neural Information Processing Systems 28*, pages 2116–2124. 2015.

[46] A. Deva, K. Abhishek, and S. Gujar. *A Multi-Arm Bandit Approach To Subset Selection Under Constraints*, page 1492–1494. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2021.

[47] N. R. Devanur and S. M. Kakade. The price of truthfulness for pay-per-click auctions. In *Proceedings of the 10th ACM Conference on Electronic Commerce*, EC '09, pages 99–106, New York, NY, USA, 2009. ACM.

[48] M. Dong, T. Meng, D. Zarchy, E. Arslan, Y. Gilad, B. Godfrey, and M. Schapira. {PCC} vivace: Online-learning congestion control. In *15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18)*, pages 343–356, 2018.

[49] E. Even-Dar, S. Mannor, and Y. Mansour. Pac bounds for multi-armed bandit and markov decision processes. In *International Conference on Computational Learning Theory*, pages 255–270. Springer, 2002.

[50] J. J. Forrest, S. Vigerske, H. G. Santos, T. Ralphs, L. Hafer, B. Kristjansson, jpfasano, EdwinStraver, M. Lubin, rlougee, jpgoncal1, h-i gassmann, and M. Saltzman. coin-or/cbc: Version 2.10.5, Mar. 2020.

[51] M. Gagliolo and J. Schmidhuber. Algorithm selection as a bandit problem with unbounded losses. In *International Conference on Learning and Intelligent Optimization*, pages 82–96. Springer, 2010.

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

[53] Y. Gai, B. Krishnamachari, and R. Jain. Combinatorial network optimization with unknown variables: Multi-armed bandits with linear rewards and individual observations. *IEEE/ACM Transactions on Networking*, pages 1466–1478, 2012.

[54] D. Garg, Y. Narahari, and S. Gujar. Foundations of mechanism design: A tutorial part 1-key concepts and classical results. *Sadhana*, 33(2):83–130, 2008.

[55] D. Garg, Y. Narahari, and S. Gujar. Foundations of mechanism design: A tutorial part 2-advanced concepts and results. *Sadhana*, 33(2):131, 2008.

[56] N. Gatti, A. Lazaric, and F. Trovò. A truthful learning mechanism for contextual multi-slot sponsored search auctions with externalities. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, EC '12, pages 605–622, New York, NY, USA, 2012. ACM.

[57] G. Ghalme, S. Dhamal, S. Jain, S. Gujar, and Y. Narahari. Ballooning multi-armed bandits. *Artificial Intelligence*, 296:103485, 2021.

[58] G. Ghalme, S. Jain, S. Gujar, and Y. Narahari. Thompson sampling based mechanisms for stochastic multi-armed bandit problems. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '17, pages 87–95, Richland, SC, 2017. International Foundation for Autonomous Agents and Multiagent Systems.

[59] J. C. Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society: Series B (Methodological)*, 41(2):148–164, 1979.

[60] A. Gopalan, S. Mannor, and Y. Mansour. Thompson sampling for complex online problems. In *International Conference on Machine Learning*, pages 100–108, 2014.

[61] T. Graepel, J. Q. Candela, T. Borchert, and R. Herbrich. Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pages 13–20, 2010.

[62] D. Haussler. *Probably approximately correct learning*. University of California, Santa Cruz, Computer Research Laboratory, 1990.

[63] D. N. Hill, H. Nassif, Y. Liu, A. Iyer, and S. Vishwanathan. An efficient bandit algorithm for realtime multivariate optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1813–1821, 2017.

[64] C.-J. Ho, S. Jabbari, and J. W. Vaughan. Adaptive task assignment for crowdsourced classification. In *International Conference on Machine Learning*, pages 534–542, 2013.

[65] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[66] IAB. Iab internet advertising revenue report. 2018 first half-year results., 2018. Available at `https://www.iab.com/`.

[67] S. Jain, S. Bhat, G. Ghalme, D. Padmanabhan, and Y. Narahari. Mechanisms with learning for stochastic multi-armed bandit problems. *Indian Journal of Pure and Applied Mathematics*, 47(2):229–272, Jun 2016.

[68] S. Jain, G. Ghalme, S. Bhat, S. Gujar, and Y. Narahari. A deterministic mab mechanism for crowdsourcing with logarithmic regret and immediate payments. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pages 86–94, 2016.

[69] S. Jain and S. Gujar. A multiarmed bandit based incentive mechanism for a subset selection of customers for demand response in smart grids. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 2046–2053. AAAI Press, 2020.

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

[71] S. Jain, B. Narayanaswamy, and Y. Narahari. A multiarmed bandit incentive mechanism for crowdsourcing demand response in smart grids. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, pages 721–727. AAAI Press, 2014.

[72] J. Jiang, R. Das, G. Ananthanarayanan, P. A. Chou, V. Padmanabhan, V. Sekar, E. Dominique, M. Goliszewski, D. Kukoleca, R. Vafin, et al. Via: Improving internet telephony call quality using predictive relay selection. In *Proceedings of the 2016 ACM SIGCOMM Conference*, pages 286–299, 2016.

[73] K. Kandasamy, A. Krishnamurthy, J. Schneider, and B. Póczos. Parallelised bayesian optimisation via thompson sampling. In *International Conference on Artificial Intelligence and Statistics*, pages 133–142, 2018.

[74] D. R. Karger, S. Oh, and D. Shah. Iterative learning for reliable crowdsourcing systems. In *Advances in neural information processing systems*, pages 1953–1961, 2011.

[75] R. Kleinberg, A. Niculescu-Mizil, and Y. Sharma. Regret bounds for sleeping experts and bandits. *Machine learning*, 80(2-3):245–272, 2010.

[76] B. Kveton, Z. Wen, A. Ashkan, and C. Szepesvari. Combinatorial cascading bandits. In *Advances in Neural Information Processing Systems*, pages 1450–1458, 2015.

[77] B. Kveton, Z. Wen, A. Ashkan, and C. Szepesvari. Tight regret bounds for stochastic combinatorial semi-bandits. In *Artificial Intelligence and Statistics*, pages 535–543, 2015.

[78] L. Lai, H. Jiang, and H. V. Poor. Medium access in cognitive radio networks: A competitive multi-armed bandit framework. In *2008 42nd Asilomar Conference on Signals, Systems and Computers*, pages 98–102. IEEE, 2008.

[79] T. L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

[80] J. Langford and T. Zhang. The epoch-greedy algorithm for contextual multi-armed bandits. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, pages 817–824, USA, 2007. Curran Associates Inc.

[81] T. Lattimore and C. Szepesvári. Bandit algorithms. *preprint*, page 28, 2018.

[82] N. Levine, K. Crammer, and S. Mannor. Rotting bandits. In *Advances in neural information processing systems*, pages 3074–3083, 2017.

[83] F. Li, J. Liu, and B. Ji. Combinatorial sleeping bandits with fairness constraints. *IEEE Transactions on Network Science and Engineering*, 2019.

[84] L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 661–670, New York, NY, USA, 2010. ACM.

[85] L. Li, W. Chu, J. Langford, and X. Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 297–306, 2011.

[86] S. Li, A. Karatzoglou, and C. Gentile. Collaborative filtering bandits. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 539–548, 2016.

[87] P. Manisha and S. Gujar. Thompson sampling based multi-armed-bandit mechanism using neural networks. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2111–2113, 2019.

[88] W. Mao, Z. Zheng, F. Wu, and G. Chen. Online pricing for revenue maximization with unknown time discounting valuations. In *IJCAI*, pages 440–446, 2018.

[89] R. Mehrotra, N. Xue, and M. Lalmas. Bandit based optimization of multiple objectives on a music streaming platform. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and; Data Mining*, KDD '20, page 3224–3233, New York, NY, USA, 2020. Association for Computing Machinery.

[90] R. B. Myerson. Optimal auction design. *Mathematics of operations research*, 6(1):58–73, 1981.

[91] Y. Narahari. *Game theory and mechanism design*, volume 4. World Scientific, 2014.

[92] A. Nika, S. Elahi, and C. Tekin. Contextual combinatorial volatile multi-armed bandit with adaptive discretization. pages 1486–1496, 2020.

[93] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, New York, NY, USA, 2007.

[94] S. Ontanón. The combinatorial multi-armed bandit problem and its application to real-time strategy games. In *Ninth Artificial Intelligence and Interactive Digital Entertainment Conference*, pages 58–64, 2013.

[95] S. Pandey, D. Agarwal, D. Chakrabarti, and V. Josifovski. Bandits for taxonomies: A model-based approach. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 216–227. SIAM, 2007.

[96] S. Pandey, D. Chakrabarti, and D. Agarwal. Multi-armed bandit problems with dependent arms. In *Proceedings of the 24th international conference on Machine learning*, pages 721–728, 2007.

[97] J. D. Papastavrou, S. Rajagopalan, and A. J. Kleywegt. The dynamic and stochastic knapsack problem with deadlines. *Management Science*, 42(12):1706–1718, 1996.

[98] F. Radlinski, R. Kleinberg, and T. Joachims. Learning diverse rankings with multi-armed bandits. In *Proceedings of the 25th international conference on Machine learning*, pages 784–791, 2008.

[99] R. P. Rooderkerk and H. J. van Heerde. Robust optimization of the 0–1 knapsack problem: Balancing risk and return in assortment optimization. *European Journal of Operational Research*, 250(3):842–854, 2016.

[100] A. D. Sarma, S. Gujar, and Y. Narahari. Truthful multi-armed bandit mechanisms for multi-slot sponsored search auctions. *Current Science*, pages 1064–1077, 2012.

[101] S. L. Scott. A modern bayesian look at the multi-armed bandit. *Applied Stochastic Models in Business and Industry*, 26(6):639–658, 2010.

[102] S. L. Scott. Multi-armed bandit experiments in the online service economy. *Applied Stochastic Models in Business and Industry*, 31(1):37–45, 2015.

[103] P. Sinha and A. A. Zoltners. The multiple-choice knapsack problem. *Operations Research*, 27(3):503–515, 1979.

[104] A. Slivkins. Introduction to multi-armed bandits. *Foundations and Trends® in Machine Learning*, 2019.

[105] J. Sublime and S. Lefebvre. Collaborative clustering through constrained networks using bandit optimization. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2018.

[106] M. S. Talebi, Z. Zou, R. Combes, A. Proutiere, and M. Johansson. Stochastic online shortest path routing: The value of feedback. *IEEE Transactions on Automatic Control*, 63(4):915–930, 2017.

[107] M. Terziovski, D. Samson, and D. Dow. The business value of quality management systems certification. evidence from australia and new zealand. *Journal of operations management*, 15(1):1–18, 1997.

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

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

[110] L. Tran-Thanh, S. Stein, A. Rogers, and N. R. Jennings. Efficient crowdsourcing of unknown experts using bounded multi-armed bandits. *Artificial Intelligence*, 214:89–111, 2014.

[111] L. Tran-Thanh, M. Venanzi, A. Rogers, and N. R. Jennings. Efficient budget allocation with accuracy guarantees for crowdsourcing classification tasks. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 901–908. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[112] U. ul Hassan and E. Curry. Efficient task assignment for spatial crowdsourcing: A combinatorial fractional optimization approach with semi-bandit learning. *Expert Systems with Applications*, 58:36–56, 2016.

[113] J. Wang, P. Zhao, S. C. Hoi, and R. Jin. Online feature selection and its applications. *IEEE Transactions on Knowledge and Data Engineering*, 26(3):698–710, 2013.

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

[115] Z. Wen, B. Kveton, and A. Ashkan. Efficient learning in large-scale combinatorial semi-bandits. In *International Conference on Machine Learning*, pages 1113–1122, 2015.

[116] L. A. Wolsey and G. L. Nemhauser. *Integer and combinatorial optimization*, volume 55. John Wiley & Sons, 1999.

[117] G. Zaimai. Optimality conditions and duality for constrained measurable subset selection problems with minmax objective functions. *Optimization*, 20(4):377–395, 1989.