Online Learning for Resource Allocation in Wireless Networks: Fairness, Communication Efficiency, and Data Privacy

Fengjiao Li

Dissertation submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computer Science and Application

Bo Ji, Chair

Ismini Lourentzou

Sharath Raghvendra

Jie Wu

Yu Wang

November 21, 2022

Blacksburg, Virginia

Keywords: resource allocation, online bandit learning, fairness, communication efficiency, data privacy

Online Learning for Resource Allocation in Wireless Networks: Fairness, Communication Efficiency, and Data Privacy

Fengjiao Li

(ABSTRACT)

As the Next-Generation (NextG, 5G and beyond) wireless network supports a wider range of services, optimization of resource allocation plays a crucial role in ensuring efficient use of the (limited) available network resources. Note that resource allocation may require knowledge of network parameters (e.g., channel state information and available power level) for packet schedule. However, wireless networks operate in an uncertain environment where, in many practical scenarios, these parameters are unknown before decisions are made. In the absence of network parameters, a network controller, who performs resource allocation, may have to make decisions (aimed at optimizing network performance and satisfying users' QoS requirements) while *learning*. To that end, this dissertation studies two novel online learning problems that are motivated by autonomous resource management in NextG.

Key contributions of the dissertation are two-fold. First, we study reward maximization under uncertainty with fairness constraints, which is motivated by wireless scheduling with Quality of Service constraints (e.g., minimum delivery ratio requirement) under uncertainty. We formulate a framework of combinatorial bandits with fairness constraints and develop a fair learning algorithm that successfully addresses the tradeoff between reward maximization and fairness constraints. This framework can also be applied to several other real-world applications, such as online advertising and crowdsourcing. Second, we consider global reward maximization under uncertainty with distributed biased feedback, which is motivated by the problem of cellular network configuration for optimizing network-level performance (e.g., average user-perceived Quality of Experience). We study both the linear-parameterized and non-parametric global reward functions, which are modeled as distributed linear bandits and

kernelized bandits, respectively. For each model, we propose a learning algorithmic framework that can be integrated with different differential privacy models. We show that the proposed algorithms can achieve a near-optimal regret in a communication-efficient manner while protecting users' data privacy "for free". Our findings reveal that our developed algorithms outperform the state-of-the-art solutions in terms of the tradeoff among the regret, communication efficiency, and computation complexity. In addition, our proposed models and online learning algorithms can also be applied to several other real-world applications, e.g., dynamic pricing and public policy making, which may be of independent interest to a broader research community.

Online Learning for Resource Allocation in Wireless Networks: Fairness, Communication Efficiency, and Data Privacy

Fengjiao Li

(GENERAL AUDIENCE ABSTRACT)

As the Next-Generation (NextG) wireless network supports a wider range of services, optimization of resource allocation plays a crucial role in ensuring efficient use of the (limited) available network resources. Note that resource allocation may require knowledge of network parameters (e.g., channel state information and available power level) for packet schedule. However, wireless networks operate in an uncertain environment where, in many practical scenarios, these parameters are unknown before decisions are made. In the absence of network parameters, a network controller, who performs resource allocation, may have to make decisions (aimed at optimizing network performance and satisfying users' QoS requirements) while *learning*. To that end, this dissertation studies two novel online learning problems that are motivated by resource allocation in the presence uncertainty in NextG.

Key contributions of the dissertation are two-fold. First, we study reward maximization under uncertainty with fairness constraints, which is motivated by wireless scheduling with Quality of Service constraints (e.g., minimum delivery ratio requirement) under uncertainty. We formulate a framework of combinatorial bandits with fairness constraints and develop a fair learning algorithm that successfully addresses the tradeoff between reward maximization and fairness constraints. This framework can also be applied to several other real-world applications, such as online advertising and crowdsourcing. Second, we consider global reward maximization under uncertainty with distributed biased feedback, which is motivated by the problem of cellular network configuration for optimizing network-level performance (e.g.,

average user-perceived Quality of Experience). We consider both the linear-parameterized and non-parametric (unknown) global reward functions, which are modeled as distributed linear bandits and kernelized bandits, respectively. For each model, we propose a learning algorithmic framework that integrate different privacy models according to different privacy requirements or different scenarios. We show that the proposed algorithms can learn the unknown functions in a communication-efficient manner while protecting users' data privacy "for free". Our findings reveal that our developed algorithms outperform the state-of-the-art solutions in terms of the tradeoff among the regret, communication efficiency, and computation complexity. In addition, our proposed models and online learning algorithms can also be applied to several other real-world applications, e.g., dynamic pricing and public policy making, which may be of independent interest to a broader research community.

Dedication

To my beloved family.

Acknowledgments

First of all, I would like to express my deepest gratitude to my Ph.D. advisor, Dr. Bo Ji. During the past five years, he has been so patient in guiding me in my research and taught me invaluable lessons in both doing research and handling problems in life. He not only asks us to give a big picture of our research (including motivation of a research topic and the intuition of a solution) but also takes care of the details of our research work. This rigorous attitude towards doing research influences me a lot, which, I believe, is really precious for my future research. In addition, I would like to thank Dr. Ji's encouragement when I was stuck in my research and got frustrated. He also inspired me how to think and how to find a solution when focusing on a specific research topic. What's more, he was always trying his best to help and support me, e.g., introducing collaborators in specific areas to me to motivate me with new insights. I am so fortunate to have Dr. Bo Ji as my advisor when pursuing my Ph.D. degree.

As introduced by Dr. Ji, I have the chance to collaborate with Dr. Jia Liu and Dr. Xingyu Zhou. Both of them helped me a lot in this dissertation. I appreciate Dr. Jia Liu who guided me in exploring solutions to the problems I met when working on part I of this dissertation and in writing papers when I was new to my Ph.D program. At the same time, I would like to thank Dr. Xingyu Zhou, who not only collaborates with me working on the second part of my dissertation but also taught me how to think and how to do research using his own experience, which is invaluable for my research. What's more, discussion with Dr. Zhou always inspires me to move deeper in understanding a problem, which, I believe, values more than the objective problem itself.

My thanks are also extended to my Ph.D. committee members: Dr. Ismini Lourentzou, Dr.

Sharath Raghvendra, Dr. Jie Wu, and Dr. Yu Wang for their time and valuable comments on this dissertation. Their suggestions help improve the quality of this dissertation to a large extent.

I would like to thank all my current and former colleagues in SNAIL Lab for their help in my research work and their friendship: Zhongdong Liu, Yu Sang, Gamal Sallam, Duo Cheng, and Osama Bajaber. I started my Ph.D. study at almost the same time as Zhongdong, who helped and supported me a lot in dealing with difficulties in my Ph.D. life besides research. In addition, I would like to extend my thanks to the friends I have at Blacksburg: Shaoran Li, Mo Hu, and Yan Huang, who are so nice and helped me when I started my life at Blacksburg, and my roommates: Yixuan Dou and Yunqian Zhang for their accompanying and making me happy.

Last but not least, I want to thank my parents, Chunlan Fan, and Xianhe Li. Their love and support make me insist and complete this program, especially their support when I got frustrated and felt tough in life. I also want to thank my husband, Haotian Chi, who brought me to U.S., which provided me a good opportunity to applying for Ph.D. program, especially joining Dr. Ji's lab. Besides the external factors, he also guided me how to read paper before starting Ph.D. study and encouraged me a lot when I got frustrated in my research.

Funding Acknowledgments

This research was supported in part by the National Science Foundation (NSF) under Grants CCF-1657162 and CNS-2112694.

Contents

List of Figures

List of Tables

Notations

Throughout this thesis, we use lower-case letters (e.g., x) for scalars, lower-case boldface letters (e.g., $\mathbf{x}$) to denote vectors; regular font symbols with subscript i (e.g., x_i) denote the coordinate corresponding to element i; $\mathbf{x}^\mathsf{T}$ denotes the transpose of $\mathbf{x}$; $\mathbf{x} \vee \mathbf{y}$ denotes the coordinate-wise maximum of $\mathbf{x}$ and $\mathbf{y}$; $\mathbf{1}$ denotes the all-ones vector; $\mathbf{0}$ denotes the all-zeros vector; $\mathbf{1}_u$ denotes the standard basis vector whose coordinates are all zero, except the one corresponding to element u being 1; $\mathbb{R}$ (resp., $\mathbb{R}_+$) is the set of (resp., nonnegative) real numbers.

In addition, we let $[N] \triangleq \{1, \ldots, N\}$ for any positive integer N; $|S|$ denotes the cardinality of set S and $\|\mathbf{x}\|_2$ denotes the ℓ_2-norm of vector $\mathbf{x} \in \mathbb{R}^d$. The inner product is denoted by $\langle \cdot, \cdot \rangle$. For a positive definite matrix $A \in \mathbb{R}^{d \times d}$, the weighted l_2-norm of vector $\mathbf{x} \in \mathbb{R}^d$ is defined as $\|\mathbf{x}\|_A \triangleq \sqrt{\mathbf{x}^\top A \mathbf{x}}$. For any sequence $\{a_t\}_{t=1}^{\infty}$, we use $a_{i:j}$ to denote the subsequence $a_i, \ldots, a_j$.

Chapter 1

Introduction

1.1 Motivation and Objectives

Over the past few decades, many optimization frameworks have studied how to manage resources in wireless networks to satisfy users' quality-of-service (QoS) requirements [64, 107, 109], e.g., fairness, latency requirement, delivery ratio, and priority-based spectrum allocation. As cellular wireless networks support a wider range of services from generation to generation, it becomes more necessary to ensure efficient use of the (limited) available network resources. Therefore, the problem of resource allocation plays an important role in wireless networks, especially NextG (5G and beyond), which is (expected) to serve an even larger diversity of spectrum, deployment options, and use cases [7, 96, 135]. Note that resource optimization may require knowledge of network parameters (e.g., channel state information and available power level for packet scheduling). However, wireless networks operate in stochastic environment under uncertainty where, in many practical scenarios, these parameters are unknown at the time of making decisions. Without knowing the network parameters, a network controller, who performs resource allocation, may have to make decisions (targeting at optimizing network performance and satisfying the requirements) while *learning*.

To this end, one tends to handle resource allocation in wireless networks via online learning which is a powerful and popular way of dealing with sequential decision making and predic-

tion problems. The goal of the decision-maker (i.e., the network controller) is to decide which action to choose at each decision time in order to maximize cumulative reward in the face of uncertainty. However, existing online learning model neglects several important factors associated with the system of cellular networks, which makes the existing learning algorithms inapplicable directly. Consider two main network resource management problems: 1) *Wireless scheduling with fairness or QoS constraints* and 2) *Cellular network configurations*. We list three main challenges of online learning approach below.

- In addition to network optimization, ensuring fairness or providing QoS guarantees to users is a key design concern in wireless scheduling [66, 95], as well as in network resource allocation [83]. However, it remains largely unexplored in the literature to carefully integrate all the factors of unknown network parameters, fairness constraints, fading channels, etc., into a unified online learning model.

- When setting a configuration in cellular networks, the controller cares more about the network-level performance, which counts the local performance of every user within the coverage. However, it incurs a prohibitively high cost or could be impossible to collecting feedback from the entire population, especially when the population is large. In order to learn the network-level performance of a configuration, the controller may have to choose a subset of users and collect the data of local performance from these distributed users. Due to heterogeneity among users, data of the local performance from a subset of users is still biased as feedback with respect to the target network-level performance.

- As communicating data between the users and the BS consumes radio resources, it requires to design a communication-efficient protocol. Furthermore, users may require that their messages/packets (e.g., user-perceived Quality of Experience (QoE)) be

protected from an adversary. In addition, as a plethora of options and parameters involved in resource management, the complexity for the high-dimension computation is another critical bottleneck in the data-driven approach.

Therefore, this dissertation, considering the above challenges associated with online learning approach for resource allocation in wireless networks, aims to build corresponding online learning models for resource optimization that handles three critical issues in cellular networks: fairness, communication efficiency, and data privacy. Specifically, motivated by the aforementioned two types of network resource management problems, the objectives of this dissertation are as follows.

- **Scheduling with fairness constraints.** Motivated by the problem of wireless scheduling with fairness or QoS requirements, we first study how to do traffic scheduling so as to maximize the reward (e.g., network utility) while satisfying the fairness requirements. Recall that *regret* as the common metric for online learning measures the loss in the obtained reward when not scheduling optimally. The goal of the scheduler becomes minimizing regret while satisfying the fairness constraints.

- **Learning with biased feedback.** The cellular network configuration problem motivates us to study global reward maximization under uncertainty with distributed biased feedback. Then, the second line of work in this dissertation to explore online learning with biased feedback with the goal of achieving sublinear regret.

- **Scalability of a design.** In the cellular networks, scalability of a design mainly refers to communication cost, which measures the amount of radio resources it consumes. In addition, we also analyze the computation complexity of a design for the high-dimensional scenarios. As users require privacy protection, a design is supposed to provide a certain privacy guarantee in some practical scenarios.

1.2 Dissertation Outline and Contributions

With the aforementioned objectives in mind, this dissertation focuses on two types of optimization problems arising from resource allocation in wireless networks and formulates four specific problems. The diagram of the dissertation is presented in Fig. 1.1.

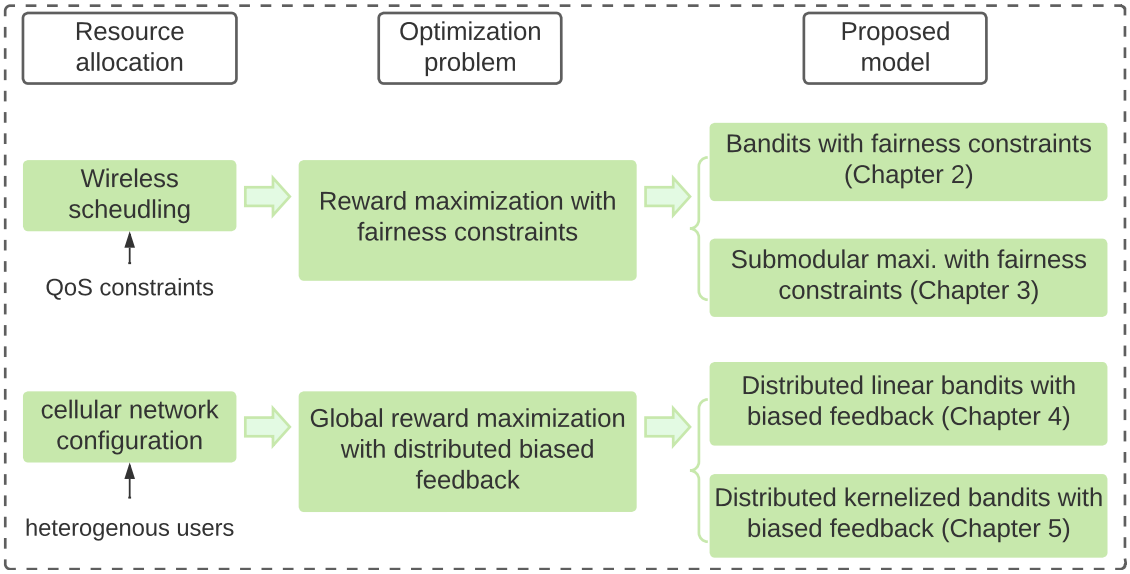


Figure 1.1: Dissertation structure.

The first type of problem is *reward maximization with fairness constraints*, which is the essense of wireless scheduling with fairness or QoS (of delivery ratio) requirements. Regarding this type of problem, this dissertation starts with formulating a combinatorial linear model of: 1) bandits with fairness constraints, and extends it to nonlinear combinatorial model of 2) submodular maximization with fairness constraints. In the following, we describe the challenges associated with the two models and list the contributions right after each model.

- **Bandits with fairness constraints.** Due to the uncertainty in the wireless scheduling problem, the reward under each possible scheduling is unknown in advance while only reward of this scheduling can be observed afterwards. This setting coincides with

the *bandit* model and motivates us to study online bandit learning with fairness constraints. In addition, in wireless network, multiple clients can be scheduled according to multiplexing techniques, and some channels might be off due to channeling fading or mobility of users. Combining these additional combinatorial and volatile structures in practice, we formulate a *combinatorial sleeping bandit with fairness constraints*, called CSMAB-F. The objective is now to maximize the reward while satisfying the fairness requirement. To tackle this new problem with a linear combinatorial reward, we develop a new algorithm, called learning with fairness guarantee (LFG), by integrating an online learning algorithm *Upper Confidence Bound* (UCB) and the virtual queue technique. In addition, we show that not only LFG is feasibility-optimal, but it also achieves a good regret upper bound.

Contribution 1: Formulate a unified CSMAB-F framework and propose an algorithm that achieves near-optimal reward while providing fairness guarantees.

Remark. In addition to scheduling of real-time traffic in wireless networks, we highlight that the unified CSMAB-F framework can be applied to many other real-world applications, including online advertising and crowdsourcing.

- **Submodular (reward/utility) maximization with fairness constraints.** While the LFG algorithm is designed for linear combinatorial reward, the combinatorial reward might be non-linear for some applications, e.g., worker selection in federated learning (FL). Notice that the utility (e.g., the training accuracy) of the participating workers exhibits a diminishing returns property, which can be modeled as a submodular function. In such scenarios, one needs to select a subset of workers for each training task to maximize the average utility over all tasks while ensuring fairness among workers. Hence, we study a new problem of multi-round (one round corresponds to one task in FL) monotone submodular maximization with cardinality and fairness con-

straints, the objective of which is to maximize the time-average utility over rounds while satisfying an additional fairness requirement. Note that the traditional submodular maximization with a cardinality constraint with known parameters is already a well-known NP-Hard problem, the fairness constraints in the the multi-round setting adds an extra layer of difficulty, and the corresponding offline problem is not trivial. We attempt to design approximate solutions to the offline problem as the first step. To address new challenges introduced by the multi-round nature and the fairness constraint, we propose three new algorithms – Fair Continuous Greedy (FairCG1 and FairCG2) and Fair Discrete Greedy (FairDG) – and provide nontrivial lower bounds on the achieved lower-average utility while all of the proposed algorithms satisfy the fairness requirement.

Contribution 2: Provide both long-term and short-term fairness guarantees when the objective reward function is submodular.

The second type of problem is *global reward maximization under uncertainty with distributed biased feedback*, which is motivated by the cellular network configuration problem and can be applied to other real-world applications as well, e.g, dynamic pricing and public policy selection. When collecting distributed data from users, scalability is also considered, including communication cost and data privacy, as well as the computation complexity when addressing high-dimensional situations. In addition, we notice a high correlation between different configurations in terms of the reward, which can be represented as a function of the decision. To that end, we start with considering linear-parameterized (unknown) reward functions and then extend to studying non-parameterized nonlinear functions. Specifically, we formulate the corresponding problem in the following two settings: 1) distributed linear bandits and 2) distributed kernelized bandits.

- **Distributed linear bandits.** We first study the linear bandit setting where different actions are correlated via an unknown parameter. Specifically, a learning agent (decision maker/learner) aims to maximize a global reward parameterized by an unknown θ^*, called the global model. Each user u observes a local reward with an unknown parameter θ_u, called the local model. In order to find the optimal action (with regard to the global reward), the agent learns the global model by collecting these clients' local feedback. To account for scalability, the agent collects distributed feedback from users periodically instead of immediately after making each decision. We call the time duration between two communications as a *phase* In a particular phase, only a subset of users from the population are selected (called clients) to participate in the learning process, and the agent (decision maker/learner) learns the global model from such biased feedback (from only a subset of users). Additionally, we resort to *differential privacy* (DP) to provide privacy guarantee. Considering different trust mechanisms in different scenarios, we propose a unified algorithmic learning framework, called differentially private distributed phased elimination (DP-DPE), which can be naturally integrated with popular differential privacy (DP) models (including central DP, local DP, and shuffle DP) while achieving both sublinear regret and sublinear communication cost.

  Contribution 3: Formulate a distributed linear bandits model and propose an unified differential private distributed phased-elimination algorithmic framework that achieves sublinear regret and communication cost while providing differential privacy guarantees.

- **Distributed kernelized bandits.** Then, we extend this work to distributed kernelized bandits that capture general *non-linear* and even *non-convex* reward functions. We assume the reward function lives in a reproducing kernel Hilbert space (RKHS) with a known kernel. In addition to the challenges in linear bandits setting, the strict generalization introduces three new challenges: (i) different from the linearly parame-

terized bandits where the bias in the feedback can be quantified with a same-dimension random vector (i.e., $\xi_u \triangleq \theta_u - \theta^* \in R^d$ at each user u), it is unclear how to make an assumption of the bias in the non-parametric kernelized bandits setting in order to learn the unknown global reward function; (ii) how to handle the possible infinite feature dimension of the objective function (in an RKHS); and (iii) how to address the bottleneck of computation complexity associated with kernelized bandits. To address all the challenges, we design an algorithm, called distributed phase-then-batch elimination algorithm (DPBE), which achieves sublinear regret, sublinear communication cost while incurring lower computation complexity than the state-of-the-art. We also highlight that thanks to the flexibility of this algorithm structure, one can easily integrate three commonly used DP models in the algorithm as well.

Contribution 4: Build distributed kernelized bandits model with distributed biased feedback and propose an algorithm that achieves near-optimal regret and sublinear communication cost while incurring lower computation complexity than the state-of-the-art.

Dissertation Structure: The rest of this Ph.D. dissertation is organized as follows. Chapter 2 and Chapter 3 present my work related to the problem of reward maximization with fairness constraints. Then, the following Chapters 4 and 5 present my work on the problem of global reward maximization with distributed biased feedback. Chapter 6 summarizes this Ph.D. dissertation and presents some interesting future work.

Chapter 2

Bandits with Fairness Constraints

2.1 Introduction

As wireless networks operate in stochastic environment under uncertainty with unknown network parameters in advance, the network controller may have to do real-time traffic scheduling sequentially while learning. This motivates us to apply online learning approach to make decisions. Note that only reward of this scheduling can be observed afterward, which coincides with the *bandit* model (with bandit observations of the unknown.)

The basic *multi-armed bandit (MAB)* model has been widely adopted for studying many practical optimization problems (network resource allocation, ad placement, crowdsourcing, etc.) with unknown parameters (see, e.g., [18]). In the basic stochastic MAB setting, there are N arms (i.e., actions), each of which, if played, returns a random reward to the player (i.e., the decision maker). The random reward of each arm takes values in $[0, 1]$ and is assumed to be *independent and identically distributed (i.i.d.)* over time. However, the reward distributions and the mean rewards are unknown *a priori*. The player decides which single arm to play in each round for a given time horizon of T rounds, with a goal of maximizing the cumulative reward in the face of unknown mean rewards.

However, this basic MAB model neglects several important factors of the system of wireless networks, where multiple actions can be simultaneously taken and an action could some-

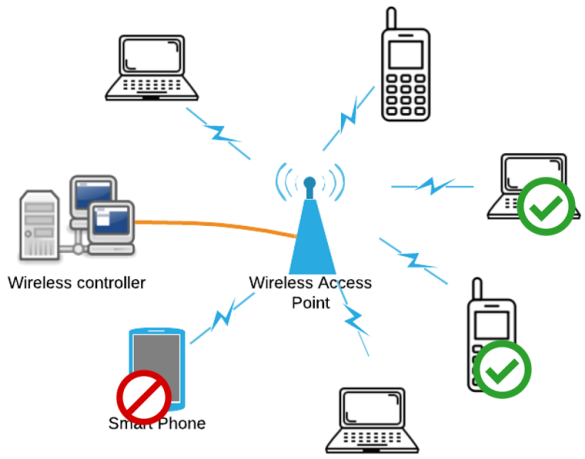


Figure 2.1: Wireless scheduling

times be "sleeping" (i.e., unavailable). Take wireless scheduling in Figure 2.1 for example: multiple clients compete for a shared wireless channel to transmit packets to a common access point (AP). The AP decides which client(s) can transmit at what times. A successfully delivered packet will generate a random reward, which could represent the value of the information contained in the packet. In each scheduling cycle, multiple clients could be scheduled for simultaneous transmissions as the channel can typically be divided into multiple "sub-channels" using multiplexing technologies [111]. On the other hand, some clients may be unable to transmit packets when experiencing a poor channel condition (due to fading or mobility). Furthermore, in addition to maximizing the reward, ensuring *fairness* among the clients or providing *Quality of Service (QoS)* guarantees to the clients is also a key design concern in wireless scheduling [66, 95], as well as in network resource allocation in general [83]. These important factors (i.e., combinatorial actions, availability of actions, and fairness) are commonly shared by many other applications too (see more detailed discussions in Section 2.6). However, it remains largely unexplored in the literature to carefully integrate all these factors into a unified MAB model.

To that end, in this chapter we propose a new *Combinatorial Sleeping MAB model with*

Fairness constraints, called *CSMAB-F*, aiming to address the aforementioned modeling issues, which are practically important for a wide variety of applications. Compared to the basic MAB setting, in the proposed framework the set of available arms follows a certain distribution that is assumed to be *i.i.d.* over time and is unknown *a priori*. However, the information of available arms will be revealed at the beginning of each round. The player can then play multiple, but no more than m, available arms and receives a compound reward being the weighted sum of the rewards of the played arms. We also impose fairness constraints that the player must ensure a (possibly different) minimum selection fraction for each individual arm. The goal is now to maximize the reward while satisfying the fairness requirement. We summarize our main contributions as follows.

- First, to the best of our knowledge, *this is the first work that integrates all three critical factors of combinatorial arms, availability of arms, and fairness into a unified MAB model*. The proposed CSMAB-F framework successfully addresses these crucial modeling issues. This new problem, however, becomes much more challenging. In particular, integrating fairness constraints adds a new layer of difficulty to the combinatorial sleeping MAB problem that is already quite challenging. This is because not only the player encounters a *fundamental tradeoff* between *exploitation* (i.e., staying with the currently-known best option) and *exploration* (i.e., seeking better options) when attempting to maximize the reward, but she is also faced with a *new dilemma*: how to manage the balance between maximizing the reward and satisfying the fairness requirement? Several well-known MAB algorithms can successfully handle the exploitation-exploration tradeoff, but none of them was designed with fairness constraints in mind.

- To address this new challenge, we extend an online learning algorithm, called *Upper Confidence Bound (UCB)*, to deal with the exploitation-exploration tradeoff and

employ the *virtual queue technique* to properly handle the fairness constraints. By carefully integrating these two techniques, we develop a new algorithm, called *Learning with Fairness Guarantee (LFG)*, for the CSMAB-F problem. Further, we rigorously prove that not only LFG is *feasibility-optimal*, but it also has a time-average *regret* (i.e., the reward difference between an optimal algorithm that has *a priori* knowledge of the mean rewards and the considered algorithm) upper bounded by $\frac{N}{2\eta} + \frac{\beta_1\sqrt{mNT\log T}+\beta_2 N}{T}$, where β_1 and β_2 are constants and η is a design parameter that we can tune. Note that our regret analysis is more challenging as the traditional regret analysis becomes inapplicable here due to the integration of virtual queues for handling the fairness constraints.

- Finally, we conduct extensive simulations to elucidate the effectiveness of the proposed algorithm. From the simulation results, we observe that LFG can effectively meet the fairness requirement while achieving a good regret performance. Interestingly, the simulation results also reveal a critical tradeoff between the regret and the speed of convergence to a point satisfying the fairness constraints. We can control and optimize this tradeoff by tuning the value of parameter η.

The rest of the chapter is organized as follows. We first discuss related work and describe the proposed CSMAB-F framework in Sections 2.2 and 2.3, respectively. Then, we develop the LFG algorithm for the CSMAB-F problem in Section 2.4, followed by the performance analysis in Section 2.5. Detailed discussions about several real-world applications are provided in Section 2.6. Finally, we present simulation results in Section 2.7 and make concluding remarks in Section 2.7.

2.2 Related Work

Following the line of variants, a recent study in [31] considers combinatorial sleeping MAB with submodular reward functions in the contextual bandit setting. This work develops a solution based on a well-known greedy algorithm for submodular maximization and prove that it can achieve a sublinear regret, which is in comparison to the greedy algorithm in the setting with known rewards.

Note that the existing types of constraints in MAB (e.g., budget/knapsack constraints) are very different from the *long-term* fairness constraints we consider in this chapter. Some very recent work considers multi-type rewards [43] and multi-level rewards [21, 30]. They introduce a minimum guarantee requirement that the total reward of some type/level must be no smaller than a given threshold. However, these studies differ significantly from ours in the following key aspects. First, and most importantly, their constraints do not model fairness among arms. The required minimum guarantee is for the total rewards (of some type/level) rather than for each individual arm. Second, no learning algorithm is proposed in [43]; the proposed learning algorithms in [21, 30] may violate the constraints, although they show provable violation bounds. Third, they assume that all the arms are available at all times. Last but not least, the proof techniques for regret analysis in [21, 30] are very different from ours.

Fairness in online learning has been studied in [72, 73]. A key idea of their proposed fair algorithm is that two arms should be played with equal probability until they can be distinguished with a high confidence. Another work [128] studies how to learn proportionally fair allocations by considering the maximization of a logarithmic utility function. These studies are less relevant to our work, although they share some high-level similarities with ours in modeling fairness.

At a technical level, the work of [67] that integrates learning and queueing is most related to ours. We follow a similar line of regret analysis in [67] for deriving the upper bound. However, they do not explicitly model fairness constraints, nor do they consider the availability of arms.

We notice that since the publication of our conference version [87], the work of [108] follows our model with a stronger fairness notion and proposes algorithms that can achieve an improved accumulative regret that is logarithmic. However, their proposed algorithms either are T-aware (i.e., assuming the knowledge of the length of the time horizon, T) or provide fairness guarantees for a special homogeneous case only, where every individual arm has the same minimum selection fraction requirement.

2.3 System Model and Problem Formulation

In this section, we describe the detailed setting of our proposed CSMAB-F framework. Let $\mathcal{N} = \{1, 2, \ldots, N\}$ denote the set of N arms. Each arm $i \in \mathcal{N}$ is associated with a reward $X_i(t)$ in round t, where $t = 0, 1, 2, \ldots$. The reward is a random variable on $[0, 1]$ and follows a certain distribution with mean μ_i. We assume that the reward for each arm is *i.i.d.* over time. The mean reward vector $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_N)$ is unknown *a priori*. In our setting, an arm could sometimes be "sleeping" (i.e., unavailable). Let $A(t) \in \mathcal{P}(\mathcal{N})$ denote the set of available arms in round t, where $\mathcal{P}(\mathcal{N})$ is the power set of $\mathcal{N}$. We use $P_{\mathbf{A}}(Z) \triangleq P(A(t) = Z)$, where $Z \in \mathcal{P}(\mathcal{N})$, to denote the distribution of available arms, which is assumed to be *i.i.d.* over time. This distribution is unknown *a priori*, but the set of available arms $A(t)$ will be revealed to the player at the beginning of each round t.

In each round, the player is allowed to play multiple, but no more than m, available arms (i.e., arms belonging to $A(t)$). Each subset of available arms is also called a *super arm* [33]. We restrict the size of a chosen super arm to be no larger than m so as to account for resource

constraints (see discussions on applications in Section 2.6). Let $\mathcal{S}(Z)$ represent the set of all feasible super arms when the set of available arms Z is observed, i.e., $\mathcal{S}(Z) \triangleq \{S \subseteq Z : |S| \leqslant m\}$, where $|S|$ denotes the cardinality of set S. In round t, a player selects a super arm $S(t) \in \mathcal{S}(A(t))$ and receives a compound reward $R(t)$, which is a weighted sum of the rewards of the played arms, i.e., $R(t) \triangleq \sum_{i \in S(t)} w_i X_i(t)$, where w_i is the weight of arm i. We assume that the weights w_i are fixed positive numbers known *a priori* and are upper bounded by a finite constant $w_{\max} > 0$. The goal of the player is to maximize the expected time-average reward for a given time horizon of T rounds, i.e., $\mathbb{E}[\frac{1}{T}\sum_{t=0}^{T-1} R(t)]$.

To describe the action for each individual arm, we use a binary vector $\mathbf{d}(t) = (d_1(t), \ldots, d_N(t))$ to indicate whether each arm is played or not in round t, where $d_i(t) = 1$ if arm i is played, i.e., $i \in S(t)$; otherwise, $d_i(t) = 0$. Then, the action vector $\mathbf{d}(t)$ must satisfy $\sum_{i=1}^{N} d_i(t) \leqslant m$ for all $t \geq 0$.

As we discussed in the introduction, in addition to maximize the reward, ensuring fairness among the arms is also a key design concern for many real-world applications. To model the fairness requirement, we introduce the following constraints on a minimum selection fraction for each individual arm:

$$\liminf_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[d_i(t)] \geqslant r_i \ \forall i \in \mathcal{N}, \tag{2.1}$$

where $r_i \in (0, 1)$ is the required minimum fraction of rounds in which arm i is played. The minimum selection fraction vector $\mathbf{r} = (r_1, \ldots, r_N)$ is said to be *feasible* if there exists a policy that makes a sequence of decisions $S(t)$ for $t \geq 0$ such that (2.1) is satisfied. Then, the *maximal feasibility region* $\mathcal{C}$ is defined as the set of all such feasible vectors $\mathbf{r} \in (0, 1)^N$. A policy is said to be *feasibility-optimal* if it can support any vector $\mathbf{r}$ (i.e., (2.1) is satisfied) strictly inside the maximal feasibility region $\mathcal{C}$.

We now consider the special class of stationary and randomized policies called *A-only policies*.

An A-only policy observes the set of available arms $A(t)$ for each round t and independently chooses a super arm $S(t) \in \mathcal{S}(A(t))$ as a (possibly randomized) function of the observed $A(t)$ only. An A-only policy α is characterized by a group of probability distributions, denoted by $\mathbf{q} = [q_S(Z), \forall S \in \mathcal{S}(Z), \forall Z \in \mathcal{P}(\mathcal{N})]$, where $q_S(Z)$ is the probability that policy α chooses super arm $S \in \mathcal{S}(Z)$ when observing the set of available arms $Z \in \mathcal{P}(\mathcal{N})$, and $\sum_{S \in \mathcal{S}(Z)} q_S(Z) = 1$ for all $Z \in \mathcal{P}(\mathcal{N})$. Then, under policy α, the action $d_i^\alpha(t)$ is *i.i.d.* over time with the following mean:

$$\mathbb{E}[d_i^\alpha(t)] = \sum_{Z \in \mathcal{P}(\mathcal{N})} P_{\mathbf{A}}(Z) \sum_{S \in \mathcal{S}(Z): i \in S} q_S(Z), \tag{2.2}$$

for every arm $i \in \mathcal{N}$ and for all $t \geq 0$, and thus, constraint (2.1) is equivalent to $\mathbb{E}[d_i^\alpha(t)] \geqslant r_i$ for every arm $i \in \mathcal{N}$. Further, we have the following lemma.

Lemma 2.1. *If a vector* $\mathbf{r}$ *is strictly inside the maximal feasibility region* $\mathcal{C}$*, then there exists an* A*-only policy that can support vector* $\mathbf{r}$*.*

Proof. The proof is omitted as it is quite standard and follows a similar line of analysis in the proof of Theorem 4.5 in [104] (see [104, pp. 92-95]). □

Lemma 2.1 implies that there exists an optimal A-only policy. Hence, assuming that the mean reward vector $\boldsymbol{\mu}$ is known in advance, one can formulate the reward maximization problem with minimum selection fraction constraint as the following linear program (LP):

$$\underset{\mathbf{q}}{\text{maximize}} \quad \sum_{Z\in\mathcal{P}(\mathcal{N})} P_{\mathbf{A}}(Z) \sum_{S\in\mathcal{S}(Z)} q_S(Z) \sum_{i\in S} w_i\mu_i \tag{2.3a}$$

$$\text{subject to} \quad \sum_{Z\in\mathcal{P}(\mathcal{N})} P_{\mathbf{A}}(Z) \sum_{S\in\mathcal{S}(Z):i\in S} q_S(Z) \geqslant r_i, \forall i \in \mathcal{N}, \tag{2.3b}$$

$$\sum_{S\in\mathcal{S}(Z)} q_S(Z) = 1, \forall Z \in \mathcal{P}(\mathcal{N}), \tag{2.3c}$$

$$q_S(Z) \in [0,1], \forall S \in \mathcal{S}(Z), \forall Z \in \mathcal{P}(\mathcal{N}). \tag{2.3d}$$

Suppose that an optimal solution to the above LP is $\mathbf{q}^* = [q_S^*(Z), \forall S \in \mathcal{S}(Z), \forall Z \in \mathcal{P}(\mathcal{N})]$. Then an optimal A-only policy α^* characterized by $\mathbf{q}^*$ obtains the maximum reward:

$$R^* \triangleq \sum_{Z\in\mathcal{P}(\mathcal{N})} P_{\mathbf{A}}(Z) \sum_{S\in\mathcal{S}(Z)} q_S^*(Z) \sum_{i\in S} w_i\mu_i. \tag{2.4}$$

However, the mean reward vector $\boldsymbol{\mu}$ is unknown to the player in advance. Hence, the player not only needs to maximize the reward based on the estimated mean rewards (i.e., exploitation), but she also has to simultaneously learn to obtain a more accurate estimate of the mean rewards (i.e., exploration). Such a learning process typically incurs a loss in the obtained reward, which is called the *regret*. Formally, the time-average regret of a policy π for a time horizon of T rounds, denoted by $R_\pi(T)$, is defined as the difference between the maximum reward R^* and the expected time-average reward obtained under policy π that chooses super arm $S(t)$ in round t, i.e.,

$$R_\pi(T) \triangleq R^* - \mathbb{E}\left[\frac{1}{T}\sum_{t=0}^{T-1}\sum_{i\in S(t)} w_i X_i(t)\right]. \tag{2.5}$$

Note that minimizing the regret is equivalent to maximizing the reward. Hence, the regret

Table 2.1: Summary of key notations

Notations	Meaning
$\mathcal{N}$; N	Set of arms; number of arms
$\mathcal{P}(\mathcal{N})$	Power set of $\mathcal{N}$
T	Time horizon
m	Maximum number of simultaneously played arms
μ_i	Mean reward of arm i
w_i	Weight of arm i
r_i	Required minimum selection fraction for arm i
$X_i(t)$	Reward of arm i in round t
$\hat{\mu}_i(t)$	Sample mean of the observed reward of arm i up to round t
$\bar{\mu}_i(t)$	UCB estimate of arm i in round t
$h_i(t)$	Number of times arm i has been played up to round t
$d_i(t)$	Indicator of whether arm i is played or not in round t
$Q_i(t)$	Virtual queue length for arm i in round t
$A(t)$	Set of available arms in round t
$S(t)$	Super arm played in round t
$P_{\mathbf{A}}(Z)$	Probability that the set of available arms is Z
$\mathcal{S}(Z)$	Set of feasible super arms when observing available arms Z
$q_S(Z)$	Probability that an A-only policy α chooses super arm S when observing available arms Z
$\mathcal{C}$	Maximal feasibility region
R^*	Maximum reward with *a priori* knowledge of $\boldsymbol{\mu}$
$R_\pi(T)$	Time-average regret of policy π

is a commonly used metric in the MAB literature for measuring the performance of learning algorithms. In this chapter, we will adopt the time-average regret defined in (2.5) as the main performance metric.

The key notations in this chapter are listed in Table 2.1.

2.4 The LFG Algorithm

In this section, by carefully integrating the key ideas of UCB [8, 82] and the virtual queue technique [104], we develop a new algorithm, called *Learning with Fairness Guarantee (LFG)*,

to tackle the CSMAB-F problem. While UCB is extended to deal with the exploitation-exploration tradeoff, the virtual queue technique is employed to handle the fairness constraints.

There are two main challenges in designing an efficient algorithm for the CSMAB-F problem: (i) how to maximize the reward in the face of unknown mean rewards and (ii) how to satisfy the fairness constraints. Note that these two challenges cannot be addressed separately as they are tightly coupled together. Therefore, we need a holistic approach to manage the balance between maximizing the reward and satisfying the fairness constraints. In what follows, we will first discuss the key ideas for addressing each individual challenge and then propose the LFG algorithm by carefully integrating them.

The key of maximizing the reward with uncertainty is to strike a balance between exploitation (i.e., choosing the option that gave highest rewards in the past) and exploration (i.e., seeking new options that might give higher rewards in the future). We extend a simple UCB policy based on the concept of optimism in the face of uncertainty to address this challenge and describe the details as follows.

Let $h_i(t)$ be the number of times arm i has been played by the end of round t, i.e., $h_i(t) \triangleq \sum_{k=0}^{t} d_i(k)$. We set $h_i(-1) = 0$ as the system begins at $t = 0$. Also, let $\hat{\mu}_i(t)$ be the sample mean of the observed rewards of arm i by the end of round t, i.e., $\hat{\mu}_i(t) \triangleq \frac{\sum_{k=0}^{t} X_i(k)d_i(k)}{h_i(t)}$. We set $\hat{\mu}_i(t) = 1$ if arm i has not been played yet by the end of round t (i.e., if $h_i(t) = 0$). We use $\bar{\mu}_i(t)$ to denote the UCB estimate of arm i in round t, which is given as follows:

$$\bar{\mu}_i(t) \triangleq \min\left\{\hat{\mu}_i(t-1) + \sqrt{\frac{3\log t}{2h_i(t-1)}}, 1\right\}, \tag{2.6}$$

where $\hat{\mu}_i(t-1)$ and $\sqrt{\frac{3\log t}{2h_i(t-1)}}$ correspond to exploitation and exploration, respectively. We use the above truncated version of the UCB estimate (i.e., capped at 1) as the actual reward

must be in $[0, 1]$. Similarly, we set $\bar{\mu}_i(t) = 1$ if $h_i(t-1) = 0$.

In the basic MAB setting, the classic UCB policy simply selects the arm that has the largest UCB estimate in each round [8, 82]. However, in the CSMAB-F setting we are faced with several new challenges introduced by combinatorial arms, availability of arms, and fairness constraints. In particular, integrating fairness constraints adds a new layer of difficulty to the combinatorial sleeping MAB problem that is already quite challenging. This is because not only the player is faced with the exploitation-exploration dilemma when attempting to maximize the reward, but she also encounters a new tradeoff between maximizing the reward and satisfying the fairness requirement. Therefore, directly applying the UCB policy will not work as it was designed without fairness constraints in mind. Next, we will explain how to use the virtual queue technique to properly handle the fairness constraints, as well as how to cohesively integrate it with UCB to address the overall challenge of the CSMAB-F problem.

Following the framework developed in [104], we create a virtual queue Q_i for each arm i to handle the fairness constraints in (2.1). By slightly abusing the notation, we also use $Q_i(t)$ to denote the queue length of Q_i at the beginning of round t, which is a counter that keeps track of the "debt" to arm i up to round t. Specifically, the virtual queue length $Q_i(t)$ evolves according to the following dynamics:

$$Q_i(t) = [Q_i(t-1) + r_i - d_i(t-1)]^+, \tag{2.7}$$

where $[x]^+ \triangleq \max\{x, 0\}$. We set $Q_i(0) = 0$ as the system begins at $t = 0$. As can be seen in the above queue-length evolution, the "debt" to arm i increases by r_i in each round as r_i is the minimum selection fraction, and it decreases by one if arm i is selected in round $t-1$ (i.e., $d_i(t-1) = 1$).

Having introduced the UCB estimate and the virtual queues, we are now ready to describe

the proposed LFG algorithm, which is presented in Algorithm 1. At the very beginning, we initialize $h_i(-1) = 0$ and $Q_i(0) = 0$ for all arms $i \in \mathcal{N}$ (lines 1-3). In each round t, we first update the UCB estimates $\bar{\mu}_i(t)$ and the virtual queue lengths $Q_i(t)$ according to (2.6) and (2.7) for all arms $i \in \mathcal{N}$, respectively, based on the decision and the feedback from the previous rounds (lines 4-11); we set $\bar{\mu}_i(t) = 1$ if $h_i(t-1) = 0$. Then, we observe the set of available arms $A(t)$ (line 12) and select a super arm $S(t) \in \mathcal{S}(A(t))$ that maximizes the compound value of the updated $\bar{\mu}_i(t)$ and $Q_i(t)$ as follows (line 13):

$$S(t) \in \operatorname*{argmax}_{S \in \mathcal{S}(A(t))} \sum_{i \in S} \left(Q_i(t) + \eta w_i \bar{\mu}_i(t)\right), \tag{2.8}$$

where η is a positive parameter we can tune to manage the balance between the reward and the virtual queue lengths. Note that the size of $\mathcal{S}(A(t))$ is exponential in m. Hence, the complexity of selecting a super arm $S(t)$ according to (2.8) could be prohibitively high in general. However, thanks to the special structure of linear compound reward, we can efficiently solve (2.8) and find a best super arm $S(t)$ by iteratively selecting best individual arms. Specifically, we select a super arm $S(t)$ consisting of the top-m^* arms in $A(t)$, where $m^* \triangleq \min\{m, |A(t)|\}$. That is, starting with an empty $S(t)$, we iteratively select arm i^* such that

$$i^* \in \operatorname*{argmax}_{i \in A(t) \setminus S(t)} Q_i(t) + \eta w_i \bar{\mu}_i, \tag{2.9}$$

and after each iteration, we update super arm $S(t)$ by adding arm i^* to it, i.e., $S(t) = S(t) \cup \{i^*\}$. Repeating the above procedure for m^* iterations solves (2.8) and finds a best super arm $S(t)$. After we play arms in $S(t)$ and set vector $\mathbf{d}(t)$ accordingly (line 14), we observe the reward $X_i(t)$ for all played arms $i \in S(t)$ (lines 15-17) and update $h_i(t)$ and $\hat{\mu}_i(t)$ accordingly for all arms $i \in \mathcal{N}$ (lines 18-20).

Remark: As we mentioned earlier, we introduce a design parameter η to manage the balance

Algorithm 1 Learning with Fairness Guarantee (LFG)

```
for i ∈ N do
    Initialize h_i(−1) = 0 and Q_i(0) = 0;
end for
In each round t:
for i ∈ N do
    if h_i(t − 1) > 0 then
        Update μ̄_i(t) according to (2.6);
    else
        Set μ̄_i(t) = 1;
    end if
    Update Q_i(t) according to (2.7);
end for
Observe the set of available arms A(t);
Select super arm S(t) according to (2.8);
Play arms in S(t) and set vector d(t) accordingly;
for i ∈ S(t) do
    Observe the reward X_i(t);
end for
for i ∈ N do
    Update h_i(t) and μ̂_i(t) according to d_i(t) and X_i(t).
end for
```

between the reward and virtual queue lengths. When η is large, the LFG algorithm gives a higher priority to maximizing the reward compared to meeting the fairness constraints. This is because an arm with a large estimated reward (i.e., UCB estimate) will be favored, compared to another arm that has a small estimated reward but a large "debt" (i.e., virtual queue length). In contrast, when η is small, the LFG algorithm gives a higher priority to meeting the fairness constraints because an arm with a large virtual queue length will be favored even if it has a small estimated reward. Indeed, our simulation results presented in Section 2.7 reveal an interesting tradeoff between the regret and the speed of convergence to a point satisfying the fairness constraints. Note that the LFG algorithm adopts a linear combination of the virtual queue length and the UCB estimate to address the trade-off between reward maximization and fairness guarantee. The reason that such a natural integration

works is partially due to the linearity of the offline problem (i.e., Eq. (2.3)). In particular, the objective function (i.e., Eq. (2.3a)) is linear because we consider a linear reward function. In the settings with more general nonlinear reward functions, such as a submodular reward function, even the offline problem with known rewards could easily become intractable (e.g., NP-hard) [80]. In such cases, it remains unclear how to design efficient algorithms that can achieve a good regret performance while satisfying the fairness constraints. We leave this question to our future work.

In addition, our proposed LFG algorithm is based on the drift-plus-penalty approach [104]. As explained in [104], this approach can be viewed as a dual-based approach to the stochastic optimization problem (i.e., the linear program formulated in Eq. (2.3)), and it reduces to the well-known dual subgradient algorithm for linear and convex programs when applied to non-stochastic optimization problems. However, to the best of our knowledge, our work is the first to employ the drift-plus-penalty approach to solve a new MAB problem with fairness constraints. The integration of the virtual queue technique and the UCB algorithm renders the regret analysis more challenging as the traditional regret analysis for the UCB algorithm becomes inapplicable here.

2.5 Main Results

In this section, we analyze the performance of our proposed LFG algorithm and present our main results. Specifically, we show that the LFG algorithm is feasibility-optimal (i.e., it can satisfy any feasible requirement of minimum selection fraction for each individual arm) in Section 2.5.1 and derive an upper bound on the time-average regret in Section 2.5.2.

2.5.1 Feasibility Optimality

We first present the feasibility-optimality result. That is, the LFG algorithm can satisfy the fairness constraints in (2.1) for any minimum selection fraction vector $\mathbf{r}$ strictly inside the maximal feasibility region $\mathcal{C}$.

Note that the constraints in (2.1) are satisfied as long as the virtual queue system defined in (2.7) is *mean rate stable* [104, pp. 56-57], i.e., $\lim_{T\to\infty} \frac{\mathbb{E}[\sum_{i=1}^{N} Q_i(T)]}{T} = 0$. In our virtual queue system, mean rate stability is implied by a stronger notion called *strong stability*, i.e., $\limsup_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}[\sum_{i=1}^{N} Q_i(t)] < \infty$. Therefore, in order to prove feasibility-optimality, it is sufficient to show that the virtual queue system is strongly stable whenever the minimum selection fraction vector $\mathbf{r}$ is strictly inside $\mathcal{C}$. We state this result in Theorem 2.2.

Theorem 2.2. *The LFG algorithm is feasibility-optimal. Specifically, for any minimum selection fraction* $\mathbf{r}$ *strictly inside the maximal feasibility region* $\mathcal{C}$*, the virtual queue system defined in* (2.7) *is strongly stable under LFG. That is,*

$$\limsup_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\left[\sum_{i=1}^{N} Q_i(t)\right] \leqslant \frac{B}{\varepsilon} < \infty, \tag{2.10}$$

where $B \triangleq \frac{N}{2} + \eta m w_{\max}$ *and* ε *is some positive constant satisfying that* $\mathbf{r} + \varepsilon\mathbf{1}$ *is still strictly inside* $\mathcal{C}$*, with* $\mathbf{1}$ *being the* N*-dimensional vector of all ones.*

We prove Theorem 2.2 by using standard Lyapunov-drift analysis [104]. The detailed proof is provided in Appendix A.1.

Remark: Note that the work of [21] also studies an MAB problem with minimum-guarantee constraints. However, their work differs significantly from ours because their considered minimum guarantee is for the total rewards (of some type/level) rather than for each individual arm, i.e., fairness among arms is not modeled. More importantly, the proposed learning

algorithm in [21] may violate the constraints. Although they show that the violations are upper bounded by $O(T^{5/6})$, this upper bound implies that the constraints may not be satisfied even after a long enough time. In stark contrast, Theorem 2.2 states that our proposed LFG algorithm can satisfy the (long-term) fairness constraints as long as the requirement is feasible. Another difference is that they do not consider sleeping bandits, which can further complicate the problem.

2.5.2 Upper Bound on Regret

In this subsection, we prove an upper bound on the time-average regret (as defined in (2.5)) under the LFG algorithm. This upper bound is achieved uniformly over time (i.e., for any finite time horizon T) rather than asymptotically when T goes to infinity. We state this result in Theorem 2.3.

Theorem 2.3. *Under the LFG algorithm, the time-average regret defined in* (2.5) *has the following upper bound:*

$$R_{LFG}(T) \leqslant \frac{N}{2\eta} + \frac{\beta_1\sqrt{mNT\log T} + \beta_2 N}{T}, \tag{2.11}$$

where $\beta_1 \triangleq 2\sqrt{6}w_{\max}$*, and* $\beta_2 \triangleq (1 + \frac{5\pi^2}{12})w_{\max}$.

We prove Theorem 2.3 by using a similar line of regret analysis in [67]. The detailed proof is provided in Appendix A.2.

Remark: The derived regret upper bound in (2.11) is quite appealing as it separately captures the impact of the fairness constraints and the impact of the uncertainty in the mean rewards for any finite time horizon T. Note that the regret upper bound in (2.11) has two terms. The first term $\frac{N}{2\eta}$ is inversely proportional to η and is attributed to the impact of the fairness

constraints. Specifically, when η is small, the LFG algorithm gives a higher priority to meeting the fairness requirement by favoring an arm with a larger "debt" (i.e., virtual queue length) as in (2.9), even if this arm has a small estimated reward. This results in a larger regret captured in the first term. Similarly, a larger η leads to a smaller regret captured in the first term, but it will take longer for the LFG algorithm to converge to a point satisfying the fairness constraints. This interesting tradeoff can also be observed from our simulation results in Section 2.7. The second term $\frac{\beta_1\sqrt{mNT\log T}+\beta_2 N}{T}$ is of the order $O(\sqrt{\log T/T})$. This part of the regret corresponds to the notion of regret in typical MAB problems and is attributed to the cost that needs to be paid in the learning/exploration process. Note that the second term is an *instance-independent* upper bound that does not depend on the problem-specific parameter $\boldsymbol{\mu}$. Our derived bound on the time-average regret is consistent with the instance-independent result for basic MAB problems [18, Ch. 2.4.3][1].

2.6 Applications

In this section, we provide more detailed discussions about real-world applications of our proposed CSMAB-F framework. Specifically, we will discuss the following three applications as examples: scheduling of real-time traffic in wireless networks [66], ad placement in online advertising systems [3], and task assignment in crowdsourcing platforms [12].

2.6.1 Scheduling of Real-time Traffic in Wireless Networks

Consider the problem of scheduling real-time traffic with QoS constraints in a single-hop wireless network. Assume that there are N clients competing for a shared wireless channel to transmit packets to a common AP (see, e.g., [66]). Time is slotted. The AP decides

[1] Time-average regret $O(\sqrt{\log T/T})$ vs. cumulative regret $O(\sqrt{T\log T})$.

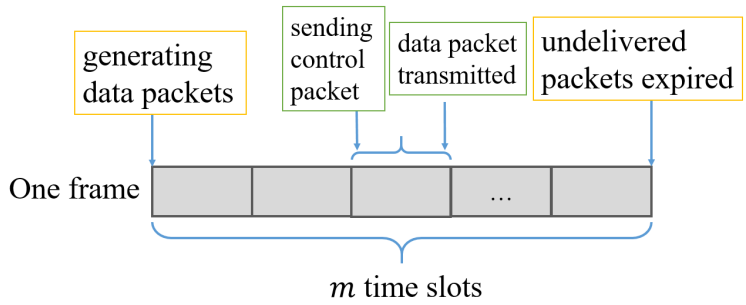


Figure 2.2: Scheduling of real-time traffic

which client(s) can transmit at what times. Consider a scheduling cycle, called a frame, that consists of m consecutive time slots. Every client generates one data packet at the beginning of each frame. To avoid interference, we assume that at most one client can transmit in each time slot. Note that some clients may sometimes be unable to transmit when experiencing poor channel conditions (due to fading or mobility). Assume that the channel conditions remain unchanged during a frame but may vary over frames and that the AP obtains the exact knowledge about the channel conditions through probing messages.

At the beginning of each frame, the AP makes scheduling decisions by selecting an available client to transmit in each of the m time slots; at the beginning of each time slot, the AP broadcasts a control packet that announces the scheduling decision, and then, the selected client transmits a packet to the AP in that time slot. We model real-time traffic by assuming that packets have a lifetime of m time slots and expire at the end of the frame. The above framework is illustrated in Figure 2.2. While a successfully delivered packet will generate a utility, which could represent the value of the information contained in the packet, an expired packet will be dropped at the end of the frame. We assume that the utility corresponding to each client is a random variable, and its mean is unknown *a priori*. There is a weight associated with each client, indicating the importance of the information provided by the client.

The goal of the AP is to maximize the cumulative utilities by scheduling packet transmissions in the face of unknown mean utilities. In addition, each client has a QoS requirement that a minimum delivery ratio must be guaranteed. Clearly, the scheduling problem with minimum delivery ratio guarantee can naturally be formulated as a CSMAB-F problem.

2.6.2 Ad Placement in Online Advertising Systems

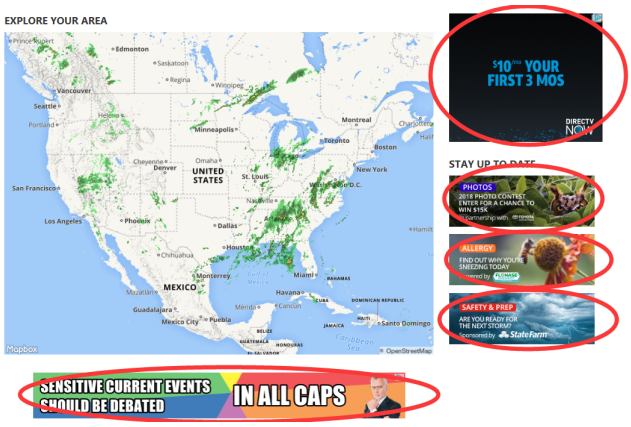


Figure 2.3: Ad placement

Online advertising has emerged as a very popular Internet application [3]. Take a page of *Weather.com* website shown in Figure 2.3 for example. When an Internet user visits the webpage, the publisher dynamically chooses multiple ads from the ads pool to display in the ad-mix areas (highlighted by red circles in Figure 2.3). We assume that the ads pool consists of N ads, and the ad-mix area has a limited capacity, which allows displaying no more than m ads simultaneously. Note that some ads are irrelevant to certain users, depending on the context including users' characteristics (gender, interest, location, etc.) and content of the webpage. Hence, such irrelevant ads can be viewed as unavailable to those users, and the availability of ads depends on the distribution of the context. After seeing a displayed ad, the user may or may not click it. The click-through rate (i.e., the rate at which the ad is clicked) of each ad is unknown *a priori*. Each click of an ad will potentially generate a

revenue for the advertiser, which can be viewed as the weight of the ad.

The goal of the ad publisher is to maximize the cumulative revenues by determining a best subset of ads to display in the face of unknown click-through rates. In addition, the publisher must guarantee a minimum display frequency for advertisers who pay a fixed cost over a specified period, regardless of users' responses to the displayed ads. Obviously, the ad placement problem with minimum display frequency guarantee fits perfectly into our proposed CSMAB-F framework.

2.6.3 Task Assignment in Crowdsourcing Platforms

Figure 2.4: Task assignment in crowdsourcing

The increasing application of crowdsourcing is significantly changing the way people conduct business and many other activities [12]. Consider a crowdsourcing platform such as Amazon Mechanical Turk, Amazon Flex (for package delivery), and Testlio (for software testing), as shown in Figure 2.4. Tasks arriving to the crowdsourcing platform will be assigned to a group of workers with different unknown skill levels. Specifically, when a task arrives, the platform may divide the task into multiple sub-tasks; then the sub-tasks will be assigned to no more than m workers from a pool of N workers, due to the number of sub-tasks or a limited budget. Note that some workers could be unavailable to take certain tasks due to various reasons (time conflicts, location constraints, limited skills, preferences, etc.). Each

completed task will generate a payoff that depends on the quality or efficiency of the workers. The payoff is a random variable, and its mean is unknown *a priori* due to unknown skill levels of workers.

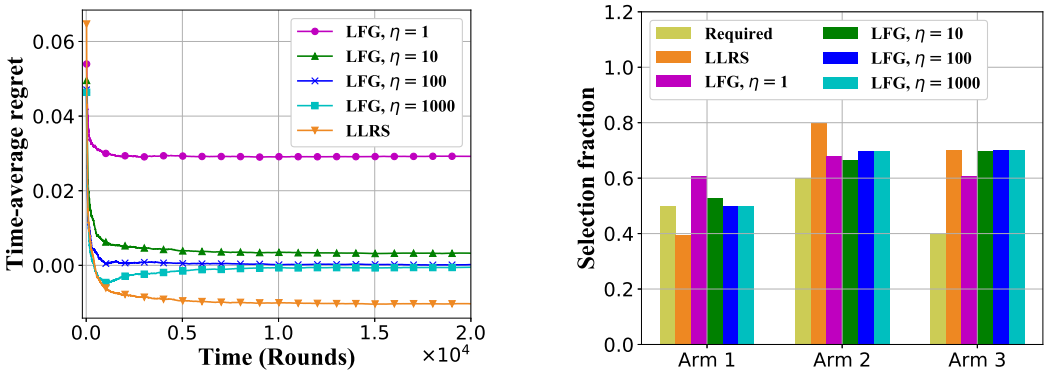
The goal of the crowdsourcing platform is to maximize the cumulative payoffs by determining an optimal task allocation in the face of unknown mean payoffs. In addition, the platform has to take fairness towards workers into account through a minimum assignment ratio guarantee for each worker. This fairness guarantee helps maintain a healthy and sustainable platform with improved worker satisfaction and higher worker participation. Apparently, our proposed CSMAB-F framework can be applied to address the task assignment problem with minimum assignment ratio guarantee.

2.7 Numerical Results

In this section, we conduct simulations to evaluate the performance of our proposed LFG algorithm and discuss several interesting observations based on the simulation results.

We consider two scenarios for the simulations: (i) $N = 3$ and $m = 2$; (ii) $N = 10$ and $m = 6$. Since the observations are similar for these two scenarios, we will focus on the discussion about the first scenario due to space limitations. We assume that the availability of arm i is a binary random variable that is *i.i.d.* over time with mean p_i. Then, the distribution of available arms can be computed as $P_{\mathbf{A}}(Z) = \prod_{i \in Z} p_i \prod_{i \notin Z}(1 - p_i)$ for all $Z \in \mathcal{P}(\mathcal{N})$. We also assume binary rewards with the same unit weight (i.e., $w_i = 1$) for all the arms. The detailed setting of other parameters is as follows: $\boldsymbol{\mu} = (0.4, 0.5, 0.7)$, $\mathbf{r} = (0.5, 0.6, 0.4)$, and $\mathbf{p} = (p_1, p_2, p_3) = (0.9, 0.8, 0.7)$.

First, in order to demonstrate that LFG can effectively meet the fairness requirement, we



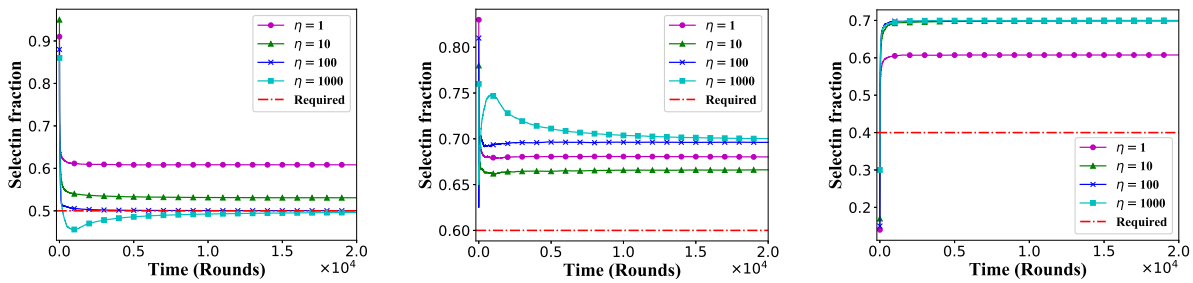
(a) Time-average regret (b) Selection fraction

Figure 2.5: Performance comparisons of different algorithms

compare LFG with a fairness-oblivious combinatorial MAB algorithm, called *Learning with Linear Rewards (LLR)* [57]. We modify the LLR algorithm to accommodate sleeping bandits; the modified version is called *LLR for Sleeping bandits (LLRS)*. In each round t, observing the set of available arms $A(t)$, LLRS selects a super arm $S(t)$ that has the largest weighted sum of the UCB estimates among all the feasible super arms in $\mathcal{S}(A(t))$, i.e., $S(t) \in \text{argmax}_{S \in \mathcal{S}(A(t))} \sum_{i \in S} w_i \bar{\mu}_i(t)$. Note that LLRS is oblivious of the fairness constraints in (2.1).

We simulate LFG with $\eta \in \{1, 10, 100, 1000\}$ and LLRS for $T = 2 \times 10^4$ rounds (at which all the considered algorithms are observed to converge) and present the results in Figure 2.5. Figure 2.5a shows the time-average regret over time for the considered algorithms; Figure 2.5b shows the selection fraction of each arm at the end of the simulation (i.e., at $T = 2 \times 10^4$). From Figure 2.5a, we can make the following observations: (i) LFG with a larger η results in a smaller regret, and LFG with $\eta \geq 100$ approaches a zero regret; (ii) LLRS achieves the smallest regret, which is even negative (i.e., it achieves a reward larger than the optimal R^*). Observation (i) is expected, as we explained in Section 2.5.2: the upper bound on regret in (2.11) approaches zero when both η and T become large. Observation (ii) is not surprising because LLRS is fairness-oblivious and may produce an infeasible solution.

Indeed, Figure 2.5b shows that Arm 1's selection fraction under LLRS is smaller than the required value (0.4 vs. 0.5). This is because Arm 1 has the smallest mean reward and is not favored under LLRS, which is unaware of the fairness contraints. On the other hand, Figure 2.5b also shows that with different values of η, LFG consistently satisfies the required minimum selection fraction, which verifies our theoretical result on feasibility-optimality of LFG (Theorem 2.2). At first glance, the above observations seem to suggest that LFG with



(a) Arm 1 (b) Arm 2 (c) Arm 3

Figure 2.6: Selection fraction over time under LFG with different values of η

a large η is desirable because that leads to a vanishing regret while still providing fairness guarantee. However, what is missing here is the speed of convergence to a point satisfying the fairness requirement, which is another critical design concern in practice. To understand the convergence speed of LFG with different values of η, in Figure 2.6 we plot the selection fraction over time for each arm. Taking Figure 2.6a for example, we can observe that the convergence slows down as η increases. In addition, before LFG with $\eta = 1000$ converges (e.g., when $T \leq 10^4$), the actual selection fraction of Arm 1 does not meet the required minimum value of 0.5. Since the constraints may be temporarily violated, the regret could even be negative before LFG converges (see $\eta = 1000$ in Figure 2.5a). Therefore, the simulation results reveal an interesting tradeoff between the regret and the convergence speed. We can control and optimize this tradeoff by tuning η. For example, for the considered scenario, LFG with $\eta = 100$ seems to achieve a good balance between the regret and the convergence speed. Finally, we want to investigate the tightness of the upper bound derived in (2.11).

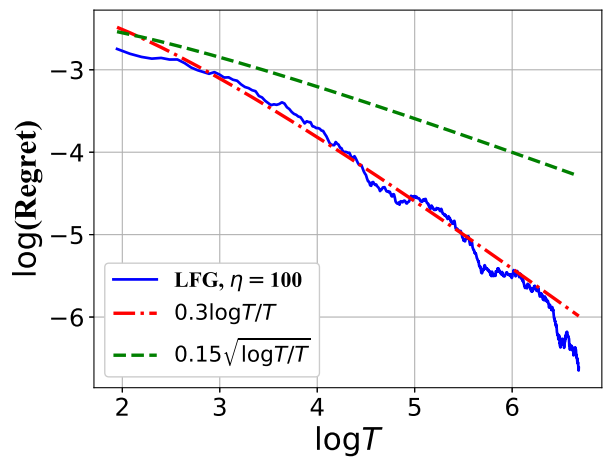


Figure 2.7: Regret vs. T

Consider the average of 100 independent simulation runs for LFG with $\eta = 100$. Figure 2.7 shows the time-average regret vs. the time horizon T in a log-log plot. Recall that the upper bound in (2.11) has two terms. The impact of T appears in the second term that is of the order $\sqrt{\log T/T}$. When T becomes large, it becomes difficult to see the impact of T on the regret as the first term $\frac{N}{2\eta}$ becomes dominant. Therefore, we consider the region with $T \leq 1000$ (i.e., $\log T \leq 6.9$). Figure 2.7 seems to suggest that the time-average regret follows the order $\log T/T$ rather than $\sqrt{\log T/T}$. This implies that the upper bound in (2.11) is not tight. One reason could be that the $\sqrt{\log T/T}$ bound is instance-independent. It remains open whether one can come up with novel analytical techniques to derive a better bound of $\log T/T$.

2.8 Chapter Summary

In this chapter, we proposed a unified CSMAB-F framework that integrates several critical factors (i.e., combinatorial actions, availability of actions, and fairness) of the system in many real-world applications. In particular, no prior work has studied MAB problems with

fairness constraints on a minimum selection fraction for each individual arm. To address the new challenges introduced by modeling these factors, we developed a new LFG algorithm that achieves a provable regret upper bound while effectively providing fairness guarantee.

Chapter 3

Submodular Reward Maximization with Fairness Constraints

The preceding chapter studies the problem of reward maximization with fairness constraints where the combinatorial reward is linear. For some real-world applications, the combinatorial reward might be nonlinear, e.g., the utility (say training accuracy) with respect to the participating clients in federated learning (FL). Notice that combinatorial maximization with submodular objective functions has been extensively studied due to a wide range of real-world applications. This chapter, motivated by the application of worker selection in federated learning, studies the problem of submodular maximization with fairness constraints.

3.1 Introduction

In the conventional machine learning paradigm, a large amount of data is often stored at a centralized server (e.g., a single machine or a datacenter) for training some learning model (e.g., a deep neural network) [2]. However, not only is this paradigm expensive in terms of data collection and storage, it also has a high risk in leaking users' data privacy [91]. For example, in 2019, hundreds of millions of Facebook user records were compromised on Amazon cloud servers [1]. To address such privacy concerns, Federated Learning (FL) has recently become a popular learning paradigm, where a myriad of worker devices collaboratively learn

a global model, while all training data are kept on the worker devices [2].

In a FL system, worker devices often have heterogeneous computation or sensing capabilities. Due to limited resources (e.g., communication bandwidth) [94, 139], it is not only desirable but also necessary for the FL server to pick a subset of workers to participate for a training task. Different worker selections yield different training accuracy levels that can be expressed as the utility of the selected workers. Note that the accuracy of a model has diminishing improvement as a function of sample size [100]. Considering the heterogeneity of the worker devices (e.g., different dataset sizes) and the redundancy among workers, the utility of the participating workers exhibits a diminishing returns property, which can be modeled as a submodular function [11]. In such scenarios, one needs to select a subset of workers for each training task to maximize the average utility over all tasks. On the other hand, ensuring fairness among the participating workers is one of the most important issues in such FL systems [92, 94]. One common fairness requirement is that each worker must be selected at least for a certain fraction of time in the long run. Introducing such fairness requirements reduces the discrimination towards those workers with weaker sensing and computing capabilities, thus providing an incentive that encourages more workers to participate in FL. Hence, there is a compelling need that motivates the worker selection problem in an FL system such that the total utility is maximized while some fairness guarantee among the workers is also ensured.

In this work, we call the entire procedure of completing a training task from selecting participants to the end of training process a "round". Noting that the training accuracy is typically a monotone submodular function of the selected workers, we formulate the FL worker selection problem as a multi-round monotone submodular maximization problem with cardinality and fairness constraints, called MMSM-CF. Different from the traditional single-round version of the problem that is focused on one-shot optimization, the goal here is to maximize the time-average utility subject to an additional fairness requirement that each worker must be

selected for a minimum fraction of time. Our main contributions are summarized as follows:

- We study the problem of fair worker selection in FL systems and formulate it as a problem of multi-round monotone submodular maximization with cardinality and fairness constraints. To the best of our knowledge, this is the first study that considers submodular maximization in the multi-round setting with fairness constraints. It is well known that the single-round version of this problem even without fairness constraint is already NP-hard. Accounting for the fairness constraint in the multi-round setting adds an extra layer of difficulty as decisions could be coupled in a sophisticated fashion over multiple rounds.
- To address this new challenge, we consider the multilinear extension of the submodular function and develop continuous greedy-based algorithms. At the first glance, it is unclear whether the original continuous greedy algorithm can achieve a provable performance guarantee in the setting with an additional fairness requirement. In particular, the initial and intermediate points of the iterative continuous greedy algorithm could even be infeasible. Interestingly, we show that a straightforward variant of the original continuous greedy algorithm, called FairCG1, does achieve an approximation ratio of $(1 - 1/e)$ for the time-average utility while satisfying the fairness requirement whenever feasible. Furthermore, we develop a new variant of the continuous greedy algorithm, called FairCG2, which explicitly accounts for the feasibility of the intermediate points during the updating process and achieves a fine-grained lower bound on the time-average utility. In addition, we propose a new discrete greedy algorithm, called FairDG, which gives a higher priority to fairness and thus ensures a stronger short-term fairness guarantee that holds in every round. FairDG also enjoys a much lower complexity compared to FairCG1 and FairCG2.
- Finally, we perform extensive simulations to verify the effectiveness of the proposed algorithms. The simulation results show that our proposed algorithms empirically achieve a near-optimal time-average utility (within 1% of the optimal). Interestingly, while FairCG1

guarantees an approximation ratio of $(1-1/e)$ uniformly for any fairness requirement, the obtained lower bound for FairCG2 is getting tighter as the fairness requirement becomes stronger.

The rest of the chapter is organized as follows. We first discuss related work in Section 3.2 and formulate the MMSM-CF problem in Section 3.3. To address this new problem with the long-term fairness constraint, we consider two fair continuous greedy algorithms (FairCG1 and FairCG2) and analyze their performance in Section 3.4. In Section 3.5, we propose a fair discrete greedy algorithm (FairDG) and show that FairDG ensures a stronger short-term fairness guarantee. Finally, we present simulation results in Section 3.6 and make concluding remarks in Section 3.7.

3.2 Related Work

Worker Selection in FL: Worker selection is an important factor of FL systems and has recently been studied in [106, 139]. The work of [106] proposed an FL protocol called FedCS, which selects the maximum possible number of workers using a greedy algorithm based on wireless states and devices' computing capabilities. However, fairness is not considered in this work. In [139], an analytical model is developed to characterize the performance of FL with different worker scheduling policies in terms of the convergence rate performance. While proportional fairness is considered, scheduling decisions are made without accounting for the training accuracy. In [92, 103], the authors considered fair resource allocation in FL systems by assigning a higher weight to workers with a higher loss. This helps reduce the bias and leads to a lower variance in the testing accuracy. However, the problem formulated therein is not submodular-based and the fairness criterion they considered is very different from ours.

Submodular Maximization: Since the seminal work in [105], the problem of monotone submodular maximization with a cardinality constraint has been extensively studied in the literature (see, e.g., [19, 81]). It is well known that this problem is NP-hard [55]. One important approximate solution is the classic (discrete) greedy algorithm, which iteratively selects an element with the largest marginal gain till the cardinality constraint is violated. It is shown that this greedy algorithm can achieve the best approximation ratio of $(1 - 1/e)$ [105]. For submodular maximization with a general matroid constraint, it has been shown that a continuous greedy algorithm combined with the pipage rounding technique can achieve the same $(1 - 1/e)$-approximation [136]. In [9], by integrating the idea of the classic discrete greedy, the authors developed a fast variant of the continuous greedy algorithm, which can significantly reduce the complexity while matching the same approximation ratio. The work of [28] studied a more general setting of maximizing a submodular function subject to a matroid and a set of linear packing constraints.

Multi-round Submodular Maximization: While most of the existing work has been focused on one-shot optimization, some recent studies investigate multi-round monotone submodular maximization with a cardinality constraint as we consider in this chapter. The most relevant work to ours is [126]. However, there are several key differences: i) their work studied a specific multi-round influence maximization in social networks; ii) fairness requirements are not considered there; iii) a set would not be selected more than once because doing so does not generate any additional gain in their model. The work of [86] also studied a similar problem of multi-round submodular maximization, with a focus on the online learning setting. Sequential submodular maximization has been considered in the active learning setting where a batch of data points are selected for labeling [138]. However, same data points would not be selected more than once either. Note that none of these studies addresses the same fairness concerns as ours. Incorporating fairness constraints

makes decisions coupled in a sophisticated fashion over multiple rounds. In addition, for FL systems, it is possible to select the same subset more than one round to maximize the average utility in our problem.

Fair Resource Allocation: Algorithmic fairness has attracted tremendous interests in the machine learning community over the past few years in various contexts [36, 52]. Various fairness criteria and learning paradigms have been discussed in the literature. Submodular maximization concerning privacy protections and fairness criteria in learning representation has been studied in [77]. However, they do not consider the multi-round setting and the fairness criterion is formulated as a robustness constraint, which is very different from ours. The work of [88] studies fair resource allocation and learning in a multi-armed bandit setting. While the long-term fairness requirement we consider is the same, the objective function considered in [88] is linear rather than submodular.

3.3 System Model and Problem Formulation

In this section, we describe the system model and formulate the MMSM-CF problem for worker selection in FL systems.

In the context of FL, we use $\mathcal{N}$ to denote the set of workers that are equipped with computing devices and are willing to participate in the training tasks. For each training task, indexed by t, the FL protocol selects a subset of workers $S_t \subseteq \mathcal{N}$ whose size is at most $k \in \{1, 2, \ldots, n\}$. This cardinality constraint is used to model limited resources (e.g., communication bandwidth). After the entire training process of task t is completed, a certain utility (which represents the training accuracy), denoted by $f_t(S_t)$, is achieved. Assume that these tasks are of the same type and have the same utility function, i.e., $f(\cdot) \triangleq f_t(\cdot), \forall t$, and

that $f(\cdot)$ is a monotone submodular function[1]. Let $f(S)$ be the utility of the selected workers S and $\mathcal{N}_k$ be the collection of all subsets of $\mathcal{N}$ of size at most k, i.e., $\mathcal{N}_k \triangleq \{S \subseteq \mathcal{N} : |S| \leqslant k\}$. We require $S_t \in \mathcal{N}_k$ for all t. Over a sequence of T training tasks, we choose a sequence of worker sets $(S_1, S_2, \dots, S_T)$ to engage in the training tasks $\{1, 2, \cdots, T\}$ and receive an average utility of $\frac{1}{T}\sum_{t=1}^{T} f(S_t)$. Throughout the rest of this chapter, we simply use round t to represent the entire training process of training task t and call each worker an element of the ground set $\mathcal{N}$.

Furthermore, we consider a fairness criterion of a minimum selection fraction for each individual worker and define a (long-term) fairness requirement in the following form:

$$\liminf_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\left[\mathbb{I}_{\{u\in S_t\}}\right] \geqslant r_u, \ \forall u \in \mathcal{N}, \tag{3.1}$$

where $\mathbb{I}_{\{\cdot\}}$ is the indicator function and $r_u \in [0,1]$ is the minimum element u has to be selected. Vector $\mathbf{r} = [r_u]_{u\in\mathcal{N}}$ is said to be *feasible* if there exists an algorithm that selects a sequence of sets $\mathcal{S} \triangleq (S_1, S_2, \dots)$ such that Eq. (3.1) is satisfied. Given a feasible $\mathbf{r}$, our goal is to schedule a sequence of sets $\mathcal{S}$ that maximizes the average expected utility while satisfying the fairness requirement in Eq. (3.1). This leads to the following problem of multi-round monotone submdoular maximization with cardinality and fairness constraints (MMSM-CF):

$$\underset{\mathcal{S}}{\text{maximize}} \quad \liminf_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\left[f(S_t)\right] \tag{3.2a}$$

$$\text{subject to} \quad (3.1) \text{ and } S_t \in \mathcal{N}_k, \forall t \in \{1, 2, \dots\}, \tag{3.2b}$$

[1]Consider a ground set $\mathcal{N}$. A set function $f : 2^{\mathcal{N}} \to \mathbb{R}_+$ is submodular if $f(A \cup \{u\}) - f(A) \geqslant f(B \cup \{u\}) - f(B)$ for every $A \subseteq B \subseteq \mathcal{N}$ and for every $u \in \mathcal{N} \setminus B$. Function f is monotone if $f(A) \leqslant f(B)$ for every $A \subseteq B \subseteq \mathcal{N}$. We assume that $f(\cdot)$ is bounded and $f(\emptyset) = 0$.

where the expectation is taken over all possible randomness of the considered algorithms. Assume a feasible fairness requirement $\mathbf{r}$. Let U_{opt} be the supremum value of the utility metric (3.2a) over all feasible algorithms. Note that U_{opt} varies with different fairness requirements $\mathbf{r}$. When $\mathbf{r} = \mathbf{0}$, we have $U_{\text{opt}} = \text{OPT}$, where $\text{OPT} \triangleq \max_{S \in \mathcal{N}_k} f(S)$ is the highest utility associated with any feasible worker set, i.e., the optimal value for the single-round version of the problem.

It is not difficult to see that the MMSM-CF problem is NP-hard. Consider the special case with fairness requirement $\mathbf{r} = \mathbf{0}$. Then, the MMSM-CF problem degenerates to finding a subset of size at most k that maximizes the submodular utility function in each round. In this case, the problem is exactly the classic one-shot submodular maximization with a cardinality constraint, which is a well-known NP-hard problem [55]. This simply implies that the MMSM-CF problem is also NP-hard.

Note that the fairness defined in Eq. (3.1) represents a long-term requirement without any short-term guarantees. We can further strengthen the constraint to other forms of short-term fairness guarantees that hold uniformly over time. For example, consider the following short-term fairness requirement[108]:

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{I}_{\{u \in S_t\}} \geqslant r_u - \frac{1}{T^{\alpha}}, \forall u \in \mathcal{N}, \forall T \in \{1, 2, \dots\}, \tag{3.3}$$

where $\alpha > 0$. We call an algorithm α-fair if it satisfies Eq. (3.3).

Clearly, by taking expectation of both sides of Eq. (3.3) and letting T go to infinity, the short-term fairness requirement with any given $\alpha > 0$ implies the long-term fairness requirement in Eq. (3.1). Also, the larger the value of α, the more stringent the short-term fairness requirement.

We note that the MMSM-CF formulation is fairly general and finds applications not only in

FL, but also in various networking and machine learning problems, including sensor scheduling in wireless sensor networks, task assignment in crowdsourcing, and data subset selection in machine learning. (See Appendix A for more detailed discussions.) In the following, we will first focus on the design of approximation algorithms for the MMSM-CF problem with long-term fairness guarantees in Section 3.4 and then design an α-fair algorithm with short-term fairness guarantees in Section 3.5.

3.4 MMSM-CF with Long-term Fairness Guarantees

In this section, we first reformulate the MMSM-CF problem with long-term fairness requirement as a Linear Programming (LP) by considering a class of stationary randomized policies. Then, we develop two continuous greedy algorithms (FairCG1 and FairCG2) and show that they both can approximately solve the MMSM-CF problem. Specifically, we show that the fairness requirement in Eq. (3.1) is satisfied whenever feasible (Theorems 3.3 and 3.5) and prove nontrivial lower bounds on the achieved time-average utility (Theorems 3.4 and 3.6).

3.4.1 LP-based Reformulation

Since the utility function does not change over rounds, the order of the selected worker sets $S_1, S_2, \ldots$ does not impact the average utility. Hence, it suffices to find an optimal assignment of time fractions among all the sets in $\mathcal{N}_k$. Then, each set will be selected in the corresponding fraction of rounds.

We consider a class of *stationary randomized policies* that randomly choose a set in each round. Consider such a stationary randomized policy π, which is characterized by a probability distribution $\mathbf{q} = [q_S]_{S \in \mathcal{N}_k}$, where q_S is the probability of choosing set $S \in \mathcal{N}_k$. Then,

we have $\sum_{S\in\mathcal{N}_k} q_S = 1$. Under policy π, the following is satisfied for all $u \in \mathcal{N}$ and for all $t \in \{1, 2, \dots\}$:

$$\begin{aligned}\mathbb{E}\left[\mathbb{I}_{\{u\in S_t\}}\right] &= \mathbb{E}\left[\sum_{S\in\mathcal{N}_k:u\in S} \mathbb{I}_{\{S_t=S\}}\right] \\ &= \sum_{S\in\mathcal{N}_k:u\in S} \mathbb{E}\left[\mathbb{I}_{\{S_t=S\}}\right] = \sum_{S\in\mathcal{N}_k:u\in S} q_S.\end{aligned} \tag{3.4}$$

Then, Eq. (3.1) can be rewritten as $\sum_{S\in\mathcal{N}_k:u\in S} q_S \geqslant r_u$ for all $u \in \mathcal{N}$. The time-average utility in Eq. (3.2a) can also be rewritten as $\sum_{S\in\mathcal{N}_k} q_S f(S)$. Therefore, for this class of stationary randomized policies, the MMSM-CF problem can be reformulated as the following LP :

$$\underset{\mathbf{q}}{\text{maximize}} \quad \sum_{S\in\mathcal{N}_k} q_S f(S) \tag{3.5a}$$

$$\text{subject to} \quad \sum_{S\in\mathcal{N}_k:u\in S} q_S \geqslant r_u, \quad \forall u \in \mathcal{N} \tag{3.5b}$$

$$\sum_{S\in\mathcal{N}_k} q_S = 1 \tag{3.5c}$$

$$q_S \in [0, 1], \forall S \in \mathcal{N}_k. \tag{3.5d}$$

Lemma 3.1. *Suppose that a fairness requirement* $\mathbf{r}$ *is feasible. Then, there is a stationary randomized policy that is optimal for the MMSM-CF problem in* (3.2).

We omit the proof of Lemma 3.1, as it follows directly from [104, Theorem 4.5], where the objective is to minimize a time-average penalty (equivalent to maximizing the time-average utility in Eq. (3.2a)) while keeping all the queues mean rate stable (equivalent to the fairness requirement in Eq. (3.1) being satisfied). Due to Lemma 3.1, we can focus on finding an optimal stationary randomized policy for the MMSM-CF problem.

While an LP can be solved in polynomial time with respect to the number of variables, in the above LP problem, the number of variables is exponential in the size of the input (i.e.,

n and k). Solving the above LP requires $O(n^k)$ *oracle queries*[2] to obtain the value of $f(S)$ for every set $S \in \mathcal{N}_k$. Since the MMSM-CF problem is NP-hard, we aim to develop efficient approximation algorithms.

In the following, we present two Fair Continuous Greedy algorithms, FairCG1 and FairCG2, by carefully integrating randomized dependent rounding with the (modified) continuous greedy algorithm based on multilinear extension [136].

3.4.2 FairCG1

As discussed in Subsection 3.4.1, solving the MMSM-CF problem is equivalent to finding a distribution that leads to a fractional vector on $[0,1]^{\mathcal{N}}$. This motivates us to explore continuous extensions of submodular functions and continuous methods that perform on $[0,1]^{\mathcal{N}}$. The multilinear extension is an important extension of submodular functions and has unique properties that are useful for (single-round) submodular maximization subject to a matroid[3] constraint. We restate the definition of multilinear extension [136] as follows.

Definition 3.2. For a set function $f : 2^{\mathcal{N}} \to \mathbb{R}$, its multilinear extension $F : [0,1]^{\mathcal{N}} \to \mathbb{R}$ is defined as

$$F(\mathbf{y}) \triangleq \sum_{S \subseteq \mathcal{N}} f(S) \prod_{u \in S} y_u \prod_{v \notin S} (1 - y_v). \tag{3.6}$$

From the above definition, we can see that the multilinear extension $F(\mathbf{y})$ is the expectation of $f(S)$ with S determined by selecting each element u independently with probability y_u. Consider submodular maximization subject to a general matroid. One can obtain the relaxed

[2]We assume access to a *value oracle* [136]. One oracle query means the value of $f(S)$ returned by the value oracle, provided an input $S \subseteq \mathcal{N}$.

[3]A *matroid* is a pair $(\mathcal{N}, \mathcal{I})$ such that $\mathcal{N}$ is a finite set, and $\mathcal{I}$ is a non-empty collection of subsets of $\mathcal{N}$ satisfying the following properties: a) $A \subseteq B \subseteq \mathcal{N}$ and $B \in \mathcal{I}$ implies $A \in \mathcal{I}$; b) for any two sets $A, B \in \mathcal{I}$, with $|A| < |B|$, there exists an element $u \in B \backslash A$ such that $A \cup \{u\} \in \mathcal{I}$. The sets in $\mathcal{I}$ are called *independent sets* [24, 81].

maximization problem by replacing the submodular function with its multilinear extension and the original matroid with its corresponding matroid polytope[4]. Then, one can attempt to approximate the original integer problem through the following two steps: i) finding an approximate fractional solution to the relaxed problem; ii) rounding the fractional solution to an integral solution without losing too much objective value [19].

The continuous greedy algorithm is an efficient method (polynomial in n) to approximate the relaxed problem. It maintains a vector $\mathbf{y}(\tau)$ that evolves during the time interval $\tau \in [0, 1]$. Specifically, it starts with a zero-vector solution, i.e., $\mathbf{y}(0) = \mathbf{0}$, gradually updates the vector on the coordinates with maximal improvement, and finally generates a fractional vector $\mathbf{y}(1)$ in the polytope. In [136], it is shown that the output $\mathbf{y}(1)$ achieves an approximation ratio of $(1-1/e)$, i.e., $F(\mathbf{y}(1)) \geqslant (1-1/e)\cdot\mathrm{OPT}$ (with a small discretization error $o(1)$). Furthermore, thanks to the convexity of the multilinear extension in any direction $\mathbf{d} = \mathbf{1}_u - \mathbf{1}_v$ for any two different elements $u, v \in \mathcal{N}$, one can perform pipage rounding [6] on $\mathbf{y}(1)$ to obtain a subset $S \subseteq \mathcal{N}$ that satisfies $f(S) \geqslant F(\mathbf{y}(1)) \geqslant (1 - 1/e) \cdot \mathrm{OPT}$.

Next, we leverage the properties of the multilinear extension and extend the continuous greedy algorithm to address the MMSM-CF problem.

Algorithm Design

Let $\mathbf{y} = [y_u]_{u\in\mathcal{N}} \in [0, 1]^{\mathcal{N}}$ with y_u being the probability of selecting element u. Obviously, under a randomized policy, y_u is the expected time fraction of selecting u. To comply to the constraints of Problem (3.5), $\mathbf{y}$ needs to satisfy $\mathbf{r} \leqslant \mathbf{y} \leqslant \mathbf{1}$ (fairness constraint) and $\mathbf{y}^\mathsf{T}\mathbf{1} \leqslant k$

[4]For a matriod $(\mathcal{N}, \mathcal{I})$, the matroid polytope is the convex hull of all the characteristic vectors of the independent sets in $\mathcal{I}$. Here, the characteristic vector of a set S is the n-dimensional vector form of S, where the coordinate corresponding to every element $u \in S$ is equal to 1 and the other coordinates are all equal to 0.

(cardinality constraint). Let P_f be the feasible region for $\mathbf{y}$:

$$P_f \triangleq \{\mathbf{y} \in [0,1]^{\mathcal{N}} : \mathbf{r} \leqslant \mathbf{y} \leqslant \mathbf{1} \text{ and } \mathbf{y}^\top \mathbf{1} \leqslant k\}. \tag{3.7}$$

Note that P_f is a polytope. Hence, one intuitive approach is to apply continuous greedy and pipage rounding technique presented in [136], where they study submodular maximization subject to a matroid contraint. In [19], it is shown that the continuous greedy method can be applied to arbitrary convex body with zero-vector inside and achieves an approximation ratio of $(1 - 1/e)$. In our problem, however, the feasible region P_f does not contain the origin or any of the characteristic vectors of the sets of size k when $\mathbf{r} > 0$. This results in infeasible intermediate points (i.e., outside of the feasible region P_f) during the iterative process. Therefore, it is unclear whether this algorithm can achieve the same approximation ratio of $(1-1/e)$ or not for the MMSM-CF problem. Specifically, the following three aspects are unclear:

i) Since the initial vector and the intermediate vectors during the iterative process of continuous greedy may be infeasible, it is unclear whether the final solution $\mathbf{y}(1)$ *is feasible or not.*

ii) Both the objective function and the form of the optimal solution of the MMSM-CF problem are very different from the problems studied in the literature. It is unclear whether directly applying continuous greedy on P_f *can still achieve an approximation ratio of* $(1 - 1/e)$ *for the MMSM-CF problem.*

iii) A deterministic pipage rounding does not work for the MMSM-CF problem with multi-round nature since it only provides a one-shot performance guarantee for the achieved utility without respecting the fairness requirement.

To address the new MMSM-CF problem, we develop the first fair continuous greedy algo-

Algorithm 2 Fair Continuous Greedy with Randomized Dependent Rounding 1 (FairCG1)

Input: $\mathbf{r}$, k, and $\mathcal{N}$
Output: $\mathcal{S} = (S_1, S_2, \dots)$
//Run Fair Continuous Greedy in P_f and get $\mathbf{y}(1)$ Set $\tau = 0$ and $\mathbf{y}(0) = \mathbf{0}$
while $\tau < 1$ **do**
  for $u \in \mathcal{N}$ **do**
    Let $w_u(\tau) = F(\mathbf{y}(\tau) \vee \mathbf{1}_u) - F(\mathbf{y}(\tau))$
  end for
  Let $\mathbf{w}(\tau) = [w_u(\tau)]_{u \in \mathcal{N}}$
  Find $\mathbf{x}(\tau) = \text{argmax}_{\mathbf{x} \in P_f} \mathbf{x}^\mathsf{T} \mathbf{w}(\tau)$
  Increase $\mathbf{y}(\tau)$ at a rate of $\frac{d\mathbf{y}(\tau)}{d\tau} = \mathbf{x}(\tau)$, where τ is increased by $d\tau$
end while
//Perform randomized dependent rounding on $\mathbf{y}(1)$
for $t = 1, 2, \cdots$ **do**
  $S_t =$DepRounding$(\mathbf{y}(1))$
end for
function DepRoundingy
  while $\exists u \in \mathcal{N}$ with $y_u \in (0, 1)$ **do**
    find $u, v,$ and $u \neq v$ such that $y_u, y_v \in (0, 1)$
    $a = \min\{1 - y_u, y_v\}$, $b = \min\{y_u, 1 - y_v\}$
    $(y_u, y_v) = \begin{cases} (y_u + a, y_v - a), & \text{w.p. } \frac{b}{a+b} \\ (y_u - b, y_v + b), & \text{w.p. } \frac{a}{a+b} \end{cases}$
  end while
  return $S = \{u \in \mathcal{N} : y_u = 1\}$
end function

rithm, called *FairCG1*, by directly employing the continuous greedy algorithm on P_f in the first step. After obtaining the fractional vector, we sequentially perform randomized dependent rounding (i.e., randomized pipage rounding) on the fractional output. Interestingly, we can show that not only can FairCG1 satisfy the fairness requirement in Eq. (3.1), but it also achieves $(1 - 1/e)$-approximation.

The detailed operations of FairCG1 are presented in Algorithm 2. It consists of two main steps.

Step 1: Run the original continuous greedy algorithm in the polytope P_f defined in Eq. (3.7).

FairCG1 keeps a fractional vector $\mathbf{y}(\tau)$ starting from $\mathbf{0}$ and evolving during the time interval $[0, 1]$. At each time point τ, it finds a vector $\mathbf{x}(\tau)$ in P_f that maximizes the dot product of $\mathbf{x}$ and $\mathbf{w}(\tau)$ and updates $\mathbf{y}(\tau)$ with a rate of $\mathbf{x}(\tau)$. By the end of this step, we obtain a fractional vector $\mathbf{y}(1)$. It is not difficult to show $\mathbf{y}(1)^\mathsf{T}\mathbf{1} = k$.

Step 2: With the fractional vector $\mathbf{y}(1)$ obtained from Step 1, we employ the randomized dependent rounding function, DepRounding, and obtain a random set S_t in each round t. Given any input $\mathbf{y}$ satisfying $\mathbf{y}^\mathsf{T}\mathbf{1} = k$, function DepRounding selects k elements from $\mathcal{N}$ with a probability specified by vector $\mathbf{y}$. Here, we only consider set S_t of size k since the utility function $f(\cdot)$ is monotone. The detailed operations of DepRounding are presented in Lines 13-19.

Remark: In the first step of FairCG1, we implement the fair continuous greedy method by performing discretization in two aspects as in [136]: i) we increase time τ by small steps of $d\tau = 1/n^2$; ii) each $F(\mathbf{y})$ is evaluated with n^5 independent samples of $f(R(\mathbf{y}))$, where $R(\mathbf{y})$ is a random set where each element u appears independently with probability y_u. The value of $f(R(\mathbf{y}))$ for each sample is obtained via an oracle query. More details of the discretization process can be found in [136], where they showed that the above discretization error is $o(1)$.

Performance of FairCG1

We present the main results regarding the performance of FairCG1: i) whether it satisfies the fairness constraints or not and ii) the achieved time-average utility. First, we show that FairCG1 guarantees the long-term fairness requirement in Eq. (3.1) as long as the requirement vector $\mathbf{r}$ is feasible. We state this result in Theorem 3.3.

Theorem 3.3. *For any feasible requirement* $\mathbf{r}$*, FairCG1 satisfies the fairness requirement in Eq.* (3.1)*. Furthermore, for any finite* T*, the selection fraction of each element* u *deviates*

from its fairness requirement r_u by any $\delta > 0$ with probability at most $e^{-2T\delta^2}$. That is, we have

$$\mathbb{P}\left\{\frac{1}{T}\sum_{t=1}^{T}\mathbb{I}_{\{u\in S_t\}} \leqslant r_u - \delta\right\} \leqslant e^{-2T\delta^2}. \tag{3.8}$$

To prove Theorem 3.3, we first show that the fractional vector $\mathbf{y}(1)$ obtained from Step 1 of FairCG1 satisfies $\mathbf{y}(1) \geq \mathbf{r}$. Then, in each round t, we select each element u with probability $y_u(1)$ using DepRounding. Therefore, the fairness requirement of Eq. (3.1) is satisfied. Furthermore, by applying the Hoeffding Bound [65], we can also show the probabilistic guarantee for fairness satisfaction in Eq. (3.8) for any finite T.

Next, we prove a nontrivial lower bound on the time-average utility achieved by FairCG1. Recall that U_{opt} is the optimal value of Problem (3.2). We state the lower bound in Theorem 3.4.

Theorem 3.4. *The expected time-average utility under FairCG1 has the following lower bound:*

$$\liminf_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[f(S_t)\right] \geqslant (1-1/e)\cdot U_{\text{opt}}. \tag{3.9}$$

To prove Theorem 3.4, we first show that the fractional vector $\mathbf{y}(1)$ satisfies $F(\mathbf{y}(1)) \geqslant (1-1/e)U_{\text{opt}}$ and then prove $\mathbb{E}[f(S_t)] \geqslant F(\mathbf{y}(1))$ for every round $t \in \{1, 2, \dots\}$. This further implies Eq. (3.9) and completes the proof. For the first part, we follow a similar line of analysis for the continuous greedy algorithm in [19]. The difference lies in the different forms of the optimal solution and the optimal value. The optimal solution we consider is a distribution $\mathbf{q}^*$ over feasible sets in $\mathcal{N}_k$, while in [19], it is a set $S^* \in \text{argmax}_{S\in\mathcal{N}_k} f(S)$. Besides, the optimal value of the MMSM-CF problem is $U_{\text{opt}} = \sum_{S\in\mathcal{N}_k} q_S^* f(S)$ compared to $f(S^*)$ in [19]. An interesting insight we obtain from the analysis is the following: Despite the above difference, the continuous greedy algorithm can still achieve an approximation ratio of $(1-1/e)$ even if it is applied to a convex region that does not contain the origin or the

characteristic vectors of any sets with size k.

The above lower bound holds uniformly for any feasible fairness requirement $\mathbf{r}$. If $\mathbf{r} = \mathbf{0}$, we have $U_{\text{opt}} = \text{OPT}$. Thus, we have $\mathbb{E}[f(S_t)] \geqslant (1 - 1/e) \cdot \text{OPT}$, which recovers the best possible approximation ratio for the classic one-shot monotone submodular maximization with a cardinality constraint.

3.4.3 FairCG2

In this subsection, we go one step further and develop a new variant of the continuous greedy algorithm, called *FairCG2*, which starts with an initial point $\mathbf{y}(0) = \mathbf{r}$ such that the intermediate points $\mathbf{y}(\tau)$ are always kept in the feasible region P_f during the updating process. We prove that FairCG2 achieves a fine-grained lower bound on the time-average utility, which can be characterized by the fairness requirement $\mathbf{r}$.

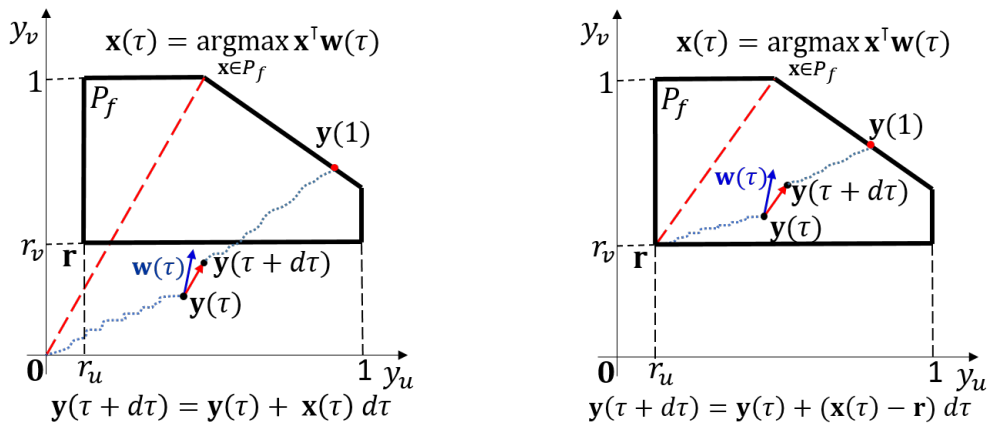
Algorithm Design

In the first step, FairCG2 starts with $\mathbf{r}$ (instead of $\mathbf{0}$ as in FairCG1) and updates the vector with an adjusted rate. We present the details of FairCG2 in Algorithm 3. Similar to FairCG1, FairCG2 consists of two main steps. The differences are as follows: i) the initial point is $\mathbf{r}$ rather than $\mathbf{0}$ (Line 2); ii) the updating rate of $\mathbf{y}(\tau)$ is $\frac{d\mathbf{y}(\tau)}{d\tau} = \mathbf{x}(\tau) - \mathbf{r}$ instead of $\mathbf{x}(\tau)$ (Line 9), where $\mathbf{x}(\tau) \triangleq \text{argmax}_{\mathbf{x} \in P_f} \mathbf{x}^\mathsf{T} \mathbf{w}(\tau)$. To better understand the difference between FairCG1 and FairCG2, we illustrate the updating processes of $\mathbf{y}(\tau)$ under these two algorithms in Fig. 3.1. By starting from $\mathbf{0}$, some points $\mathbf{y}(\tau)$ along the path taken by FairCG1 may not be feasible (i.e., outside of P_f) while every point $\mathbf{y}(\tau)$ under FairCG2 is feasible. However, the output $\mathbf{y}(1)$ is feasible under both FairCG1 and FairCG2. We also have $\mathbf{y}(1)^\mathsf{T}\mathbf{1} = k$ under FairCG2. Then, we round the fractional vector $\mathbf{y}(1)$ with randomized dependent rounding

Algorithm 3 Fair Continuous Greedy with Randomized Dependent Rounding 2 (FairCG2)

1: Same as Algorithm 2 except for Lines 2 and 9:
Line 2: Set $\tau = 0$ and $\mathbf{y}(0) = \mathbf{r}$
Line 9: Increase $\mathbf{y}(\tau)$ at a rate of $\frac{d\mathbf{y}(\tau)}{d\tau} = \mathbf{x}(\tau) - \mathbf{r}$, where τ is increased by $d\tau$

method in each round. In addition, the discretization in FairCG2 is performed in the same way as that in FairCG1.



(a) FairCG1 (b) FairCG2

Figure 3.1: Updating process of $\mathbf{y}(\tau)$ under FairCG1 and FairCG2.

Performance of FairCG2

We present the main results for FairCG2 in terms of fairness satisfaction and utility.

First, it is not difficult to show that FairCG2 guarantees the long-term fairness requirement in Eq. (3.1) whenever the requirement $\mathbf{r}$ is feasible. We state this result in Theorem 3.5 and omit the proof since it is almost the same as that of Theorem 3.3.

Theorem 3.5. *For any feasible requirement* $\mathbf{r}$*, FairCG2 satisfies the fairness requirement in Eq.* (3.1) *and also guarantees Eq.* (3.8)*.*

Next, we show that by starting from $\mathbf{r}$ and updating the process with rate $\mathbf{x}(\tau) - \mathbf{r}$, FairCG2

offers a fine-grained lower bound on the achieved time-average utility, which can be characterized by the fairness requirement $\mathbf{r}$. We present this result in Theorem 3.6. Recall that U_{opt} is the optimal value of Problem (3.2) and that $F(\cdot)$ is the multilinear extension of $f(\cdot)$.

Theorem 3.6. *The time-average expected utility under FairCG2 has the following lower bound:*

$$\liminf_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\left[f(S_t)\right] \geqslant (1-1/e^{c_{\mathbf{r}}})\cdot U_{\text{opt}} + F(\mathbf{r})/e^{c_{\mathbf{r}}}, \tag{3.10}$$

where $c_{\mathbf{r}} \triangleq 1-\max\{\max_u r_u, \mathbf{r}^{\mathsf{T}}\mathbf{1}/k\}$.

Similar to the analysis of FairCG1, we prove Theorem 3.6 as follows: we first show that the fractional vector $\mathbf{y}(1)$ satisfies $F(\mathbf{y}(1)) \geqslant (1-1/e^{c_{\mathbf{r}}})\cdot U_{\text{opt}} + F(\mathbf{r})/e^{c_{\mathbf{r}}}$ and then prove $\mathbb{E}[f(S_t)] \geqslant F(\mathbf{y}(1))$ for each t. The analysis is slightly different since both the starting point and the updating rate depend on the fairness requirement $\mathbf{r}$.

Remark: The lower bound in Theorem 3.6 appears to be getting tighter as the fairness requirement becomes more stringent. This can be observed in the simulation results (see Fig. 3.5). Furthermore, we can show that the lower bound is at least $(1-1/e)\cdot U_{\text{opt}}$ in two extreme cases: i) $\mathbf{r}=\mathbf{0}$; and ii) $\mathbf{r}=\mathbf{r}'$ with $\mathbf{r}'^{\mathsf{T}}\mathbf{1}=k$. When $\mathbf{r}=\mathbf{0}$, we have $c_{\mathbf{r}}=1$, $F(\mathbf{r})=0$, and $U_{\text{opt}}=\text{OPT}$. This implies $\mathbb{E}\left[f(S_t)\right] \geqslant (1-1/e)\cdot\text{OPT}$, which recovers the best possible approximation ratio. In another extreme case of $\mathbf{r}=\mathbf{r}'$, we have $c_{\mathbf{r}'}=0$, $P_f=\{\mathbf{r}'\}$, and thus $\mathbf{y}(1)=\mathbf{r}'$. According to [29, Theorem II.1], we obtain $\mathbb{E}\left[f(S_t)\right] \geqslant F(\mathbf{r}')$. Let $U_{\text{opt}}(\mathbf{r}')$ denote the optimal time-average utility with fairness requirement $\mathbf{r}'$. Combining the result in [24], it is not difficult to derive $F(\mathbf{r}') \geqslant (1-1/e)\cdot U_{\text{opt}}(\mathbf{r}')$, and then, we have $\mathbb{E}\left[f(S_t)\right] \geqslant (1-1/e)\cdot U_{\text{opt}}$ as well for $\mathbf{r}=\mathbf{r}'$.

3.4.4 Complexity of FairCG1 and FairCG2

To study the complexity of an algorithm for submodular optimization, it is common to analyze the number of oracle queries required by the algorithm [19]. Here, we consider not only the number of oracle queries but also the running time of FairCG1 and FairCG2, since the problem itself has a multi-round nature. Specifically, the first step of FairCG1 and FairCG2 is a variant of the continuous greedy algorithm, implemented with discretization, which results in $2n^8$ oracle queries in the first step. Consider T rounds. During the second step, performing $\textsc{DepRounding}(\mathbf{y}(1))$ takes at most n iterations in each round, which results in $O(nT)$ running time with no oracle queries. Therefore, the complexity of FairCG1 and FairCG2 is[5] $(O^{\dagger}(n^8) + O(nT))$, which could be quite high as n gets large. This motivates us to further develop low-complexity approximation algorithms to solve MMSM-CF.

The work of [9] developed a fast variant of the continuous greedy algorithm that integrates the discrete greedy algorithm and has a significantly lower complexity, while matching the best known approximation ratio of $(1 - 1/e)$. The basic idea is the following. In each while loop iteration (i.e., Lines 3-10) of Algorithm 2, continuous greedy aims to find some $\mathbf{x}(\tau)$ in the polytope, which is the characteristic vector of some size-k subset when the constraint is a matroid. Such a subset can be approximately found using the idea of discrete greedy with a much lower complexity. However, the fairness constraint we consider renders this fast continuous greedy algorithm inapplicable. This is because $\mathbf{x}(\tau) \in P_f$ is not necessarily a characteristic vector of some size-k subset in our case. Therefore, discrete greedy cannot be applied here. It also remains unclear how one can adapt such a fast continuous greedy algorithm to address the MMSM-CF problem.

[5] We use $O^{\dagger}(\cdot)$ to denote the number of oracle queries.

Algorithm 4 Fair Discrete Greedy (FairDG)

Input: $\mathbf{r}$, k, and $\mathcal{N}$
Output: $\mathcal{S} = (S_1, S_2, \dots)$
Initialize $N_{u,0} = 0$ for all $u \in \mathcal{N}$
for $t = 1, 2, \cdots$ **do**
  $A_t = \{u \in \mathcal{N} : r_u t - N_{u,t-1} \geqslant 0\}$
  $l = |A_t|$
  if $l < k$ **then**
    $B_0 = A_t$
    for $j = 1, \cdots, k - l$ **do**
      $v \in \operatorname{argmax}_{u \in \mathcal{N} \setminus B_{j-1}} \Delta(u|B_{j-1})$
      $B_j = B_{j-1} \cup \{v\}$
    end for
    $S_t = B_{k-l}$
  else
    $B_0 = \emptyset$
    for $j = 1, \cdots, k$ **do**
      $v \in \operatorname{argmax}_{u \in \mathcal{N} \setminus B_{j-1}} (r_u t - N_{u,t-1})$
      $B_j = B_{j-1} \cup \{v\}$
    end for
    $S_t = B_k$
  end if
end for

3.5 MMSM-CF with Short-Term Fairness Guarantees

In this section, by taking into account the fairness requirement, we develop a new variant of the discrete greedy algorithm, which has a much lower complexity than FairCG1 and FairCG2, especially when the size of the ground set (i.e., n) is large. Moreover, this new algorithm ensures the stronger short-term fairness requirement in Eq. (3.3). Recall that the classic discrete greedy algorithm for single-round monotone submodular maximization with a cardinality constraint starts with an empty set and iteratively adds an element with the largest marginal gain until the cardinality constraint is violated. This simple and efficient greedy algorithm achieves the best possible approximation ratio of $(1 - 1/e)$ [105]. Hence, a natural idea is to give a higher priority to the fairness requirement while being greedy.

Specifically, in each round, we first check the violation of the fairness requirement for each element and then make decisions in a greedy manner by giving the unsatisfied element a higher priority. We call this new algorithm Fair Discrete Greedy (*FairDG*). The detailed operations of FairDG are presented in Algorithm 4.

Let $N_{u,t} \triangleq \sum_{t'=1}^{t} \mathbb{I}_{u \in S_{t'}}$ denote the number of times element u has been selected by the end of round t, and let $\Delta(u|S) \triangleq f(S \cup \{u\}) - f(S)$ be the marginal gain of adding element u to current set S. At the beginning of each round t, we find a set A_t consisting of elements u with a nonnegative "debt" (i.e., violation of fairness requirement), i.e., $r_u t - N_{u,t-1} \geqslant 0$. Then, depending on the size of A_t, FairDG performs in two different ways. Let $|A_t| = l$. If there are less than k violated elements, i.e., $l < k$, then we pick all of these violated elements and select $k - l$ elements from the remaining (satisfied) elements according to their marginal gain $\Delta(u|S)$ as in the classic discrete greedy method; if there are at least k violated elements, i.e., $l \geqslant k$, then we select k of them according to their "debts" in a greedy manner. By aggressively selecting the unsatisfied elements in each round, FairCG2 guarantees the short-term fairness in Eq. (3.3) with a homogeneous fairness requirement:

Theorem 3.7. *Assume $r_u = r$ for every element $u \in \mathcal{N}$, where $r \geqslant 0$ and $nr \leqslant k$. The FairDG algorithm is 1-fair.*

The proof is inspired by [108]. We show that the fairness "debt" for each element u by the end of round t is less than one, i.e., $rt - N_{u,t} < 1$ for all u in $\mathcal{N}$ and each t. However, our proof is more involved because of the combinatorial nature in each round. Details can be found in Appendix B.5.1.

Remark: Theorem 3.7 implies that "debt" of each element will go to zero as T goes to infinity. This further implies that the long-term fairness requirement in Eq. (3.1) can be satisfied. FairDG satisfies a short-term fairness requirement by giving a higher priority

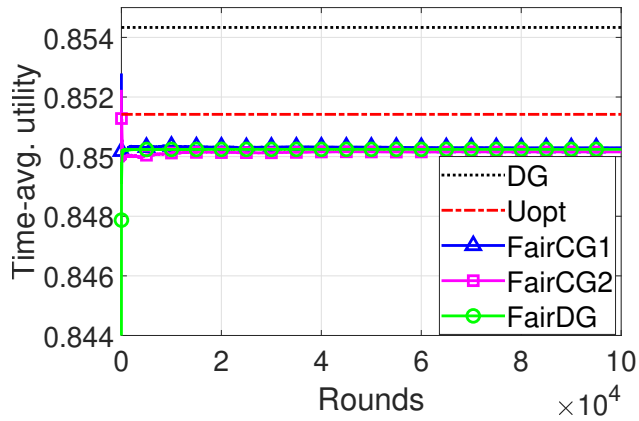


Figure 3.2: Utility over rounds

to fairness in each round. However, this priority in fairness makes it quite challenging to analyze the time-average utility since FairDG has to select part of a set without accounting for the utility. We leave the utility analysis under FairDG for future work. Note that the optimal time-average utility U_{opt} for MMSM-CF with the long-term fairness requirement offers a natural upper bound on the optimal time-average utility with a short-term fairness constraint.

FairDG has a complexity of $O^{\dagger}(knT)$, as there are T rounds, and in each round, it performs at most $O^{\dagger}(kn)$ oracle queries.

3.6 Numerical Results

In this section, we conduct simulations to evaluate the performance of our proposed algorithms (FairCG1, FairCG2, and FairDG). Specifically, our simulations are designed to answer the following questions: i) Whether the proposed algorithms satisfy the fairness requirements? ii) How well do the proposed algorithms perform in term of the time-average utility? iii) How does the fairness requirement impact the achieved utility and the derived lower bounds of our proposed algorithms? iv) How tight are the theoretical lower bounds?

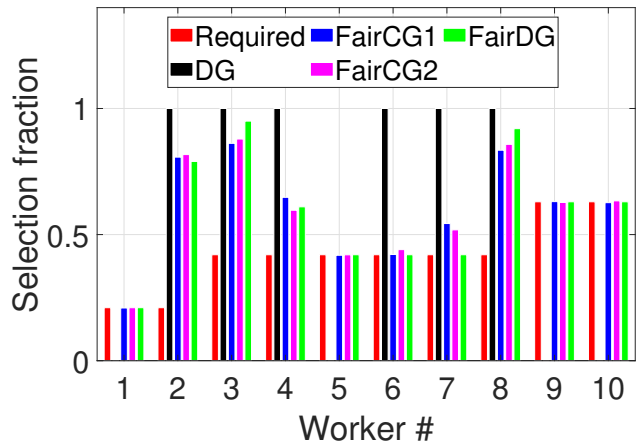


Figure 3.3: Selection fractions in T rounds

Table 3.1: Parameters settings

Worker index	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8	u_9	u_{10}
$L_u/1000$	0.2	0.8	1	0.5	0.1	0.3	0.4	0.9	0.1	0.2
$\mathbf{r}_{\text{base}}$	0.5	0.5	1	1	1	1	1	1	1.5	1.5
$\mathbf{r}$	$\mathbf{r} = \beta \mathbf{r}_{\text{base}}$									

In simulations, we assume *i.i.d.* datasets and the same computing capabilities across workers for simplicity and use the accuracy function in [56, Eq. (1)] by setting minimum achievable error $a = 0.05$, learning rate $b = 0.5$, and decay rate $c = -0.2$. Let L_u be the number of samples involved in worker u. Then, the expected utility of selecting set S is

$$f(S) = (1 - a) - b * \left(\sum_{u \in S} L_u \right)^c . \tag{3.11}$$

Throughout the simulations, we set $n = 10, k = 6$, and $T = 10^5$. Other parameters are presented in Table 3.1. The optimal average utility U_{opt} is obtained by running an LP solver in Matlab (`linprog`) to solve Problem (3.5).

First, we evaluate the performance of the three proposed algorithms (FairCG1, FairCG2, and FairDG) in terms of the time-average utility and the fairness requirement satisfaction. Let

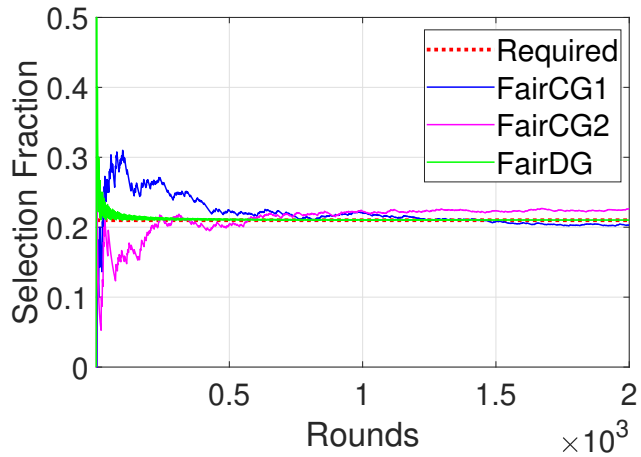


Figure 3.4: Selection fraction of worker u_1

$\beta = 0.42$. Then, $\mathbf{r} = \beta \mathbf{r}_{\text{base}} = [.21, .21, .42, .42, .42, .42, .42, .42, .63, .63]$, which is feasible since $\mathbf{r}^\top \mathbf{1} = 4.2 < k = 6$. For comparisons, we also consider the discrete greedy algorithm (denoted by DG). The simulation results are presented in Fig. **??**. Specifically, in Fig. 3.2, we display the time-average utility over T rounds for the considered algorithms, including the optimal value U_{opt}. Fig. 3.3 shows the selection fraction of each element at the end of T rounds. Based on the simulation results, we make the following observations: i) From Fig. 3.2, we observe that the achieved utility of our proposed algorithms is very close to the optimal value U_{opt} (within 1%). Also, Fig. 3.3 indicates that the selection fraction of each worker under each of our proposed algorithms satisfies the required selection fraction. ii) From Fig. 3.2, DG appears to achieve the largest time-average utility, which is even higher than the optimal value U_{opt}. However, as shown in Fig. 3.3, only k workers $\{u_2, u_3, u_4, u_6, u_7, u_8\}$ are repeatedly selected in every round under DG, which implies that the fairness requirement is not satisfied for the other workers.

In addition, we consider the selection fraction of each worker over rounds and only present the results for worker u_1 over the first $2,000$ rounds in Fig. 3.4 as a representative example. While FairCG1 and FairCG2 perform slighly better than FairDG in terms of the time-

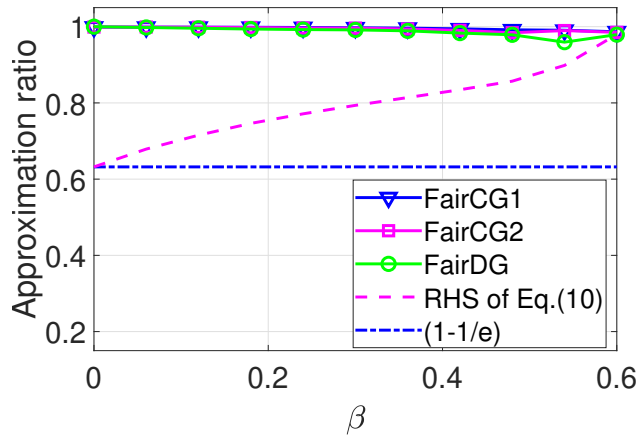


Figure 3.5: Impact of fairness requirement on time-average utility.

average utility as shown in Fig. 3.2, we observe from Fig. 3.4 that FairDG converges to a point satisfying the fairness requirement (i.e., 0.21) much faster than FairCG1 and FairCG2 do. This is not surprising because FairDG gives a higher priority to satisfying the fairness requirement.

Finally, we investigate the impact of the fairness requirement $\mathbf{r}$ on the time-average utility and the tightness of the theoretical lower bounds derived in Theorems 3.4 and 3.6. We set different values of the fairness requirement $\mathbf{r} = \beta \mathbf{r}_{\text{base}}$ by scaling the value of β. A larger value of β means a stronger fairness requirement. Consider $\beta \in \{0, .06, .12, .18, .24, .30, .36, .42, .48, .54, .60\}$, i.e., we have 11 distinct fairness requirement vectors, all of which are feasible. We run the proposed algorithms for each of them and plot the corresponding approximation ratio (the time-average utility over the optimal value U_{opt}) in Fig. 3.5. The ratios of the lower bounds in Theorems 3.4 and 3.6 over U_{opt} are presented as well. From Fig. 3.5, we can observe that the approximation ratios under FairCG1, FairCG2, and FairDG are close to one for each fairness requirement. Interestingly, while FairCG1 is guaranteed to achieve an approximation ratio of $(1-1/e)$ uniformly for different fairness requirements (Theorem 3.4), the ratio for the theoretical lower bound of FairCG2 increases with the fairness requirement and approaches one.

Hence, we have a tighter bound for FairCG2 as the fairness requirement becomes stronger. By having a lower bound with respect to fairness requirement $\mathbf{r}$ in Theorem 3.6, we obtain a tighter characterization for FairCG2 compared to FairCG1.

3.7 Chapter Summary

In this chapter, we formulated the fair worker selection in FL systems as a novel problem of multi-round monotone submodular maximization with cardinality and fairness constraints. To address this new problem, we proposed three carefully designed algorithms (i.e., FairCG1, FairCG2, and FairDG). We presented both theoretical and simulation results to demonstrate the effectiveness of our proposed algorithms. Our study in this chapter raises several interesting questions that are worth investigating as future work. For example, can we establish approximation guarantees for FairDG that satisfies the short-term fairness criterion? While we assume the same utility function over rounds in our model, it would be interesting to consider the setting with round-dependent (i.e., task-dependent) utility functions, which better suits certain applications. In this setting, we may still adopt FairCG1 and FairCG2 by applying them to each round with a different utility function, but that would incur a much higher complexity. In contrast, FairDG can be directly applied to this setting with the same complexity and the short-term fairness guarantee. However, it remains open to analyze the achieved utility under FairDG. Finally, if the submodular function under consideration is unknown in advance, it is highly interesting to investigate the joint learning and selection problem of multi-around submodular optimization.

Chapter 4

Distributed Linear Bandits with Biased Feedback

4.1 Introduction

The NextG cellular networks specify a plethora of options and parameters for resource management that need to adapt to different scenarios with the goal of achieving a high network-level performance [96], i.e., cellular network configuration. Without knowing the network-level performance of each configuration in advance, the central controller may employ bandits learning to balance *exploitation* and *exploration* when applying a parameter configuration.

While the stochastic multi-armed bandits (MAB) model is useful for many real-world applications [82], one key limitation is that actions are assumed to be independent, which, however, is usually not the case in practice. Therefore, the linear bandit model that captures the correlation among actions has been extensively studied [4, 84, 90].

In this chapter, we introduce a new linear bandit setting where the reward of an action could be from a large population. Take the cellular network configuration as an example (see Fig. 4.1). The configuration (antenna tilt, maximum output power, inactivity timer, etc.) of a base station (BS) – we denote its associated feature vector by $\mathbf{x} \in \mathbb{R}^d$ – influences all

the users under the coverage of this BS [97]. After a configuration is applied, the BS receives a reward in terms of the network-level performance, which accounts for the performance of all users within the coverage (e.g., average user throughput). Specifically, let the mean global reward of configuration x be $f(\mathbf{x}) = \langle \theta^*, \mathbf{x} \rangle$, where $\theta^* \in \mathbb{R}^d$ represents the unknown global parameter. While some configuration may work best for a specific user, only one configuration can be applied at the BS at a time, which, however, simultaneously influences all the users within the coverage. Therefore, the goal here is to find the best configuration that maximizes the global reward (i.e., the network-level performance). At first glance, it

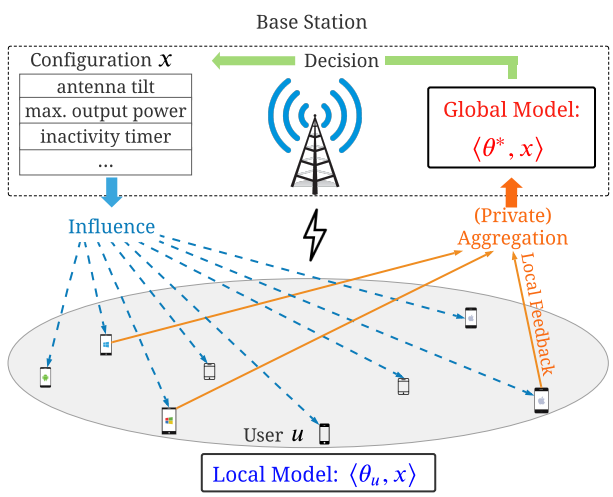


Figure 4.1: Cellular network configuration: a motivating application of global reward maximization with partial feedback in a linear bandit setting.

seems that one can address the above problem by applying existing linear bandit algorithms (e.g., LinUCB [90]) to learn the global parameter θ^*. However, this would require collecting reward feedback from the entire population, which could incur a prohibitively high cost or could even be impossible to implement in practice when the population is large. To learn the global parameter, one natural way is to sample a subset of users from the population and aggregate this distributed partial feedback. This leads to a new problem we consider in this chapter: *global reward maximization with partial feedback in a distributed linear bandit*

Table 4.1: Summary of main results

Algorithm[1]	Regret[2]	Communication cost[3]	Privacy
DPE	$O\left(T^{1-\alpha/2}\sqrt{\log(kT)}\right)$	$O(dT^{\alpha})$	None
CDP-DPE	$O\left(T^{1-\alpha/2}\sqrt{\log(kT)} + d^{3/2}T^{1-\alpha}\sqrt{\ln(1/\delta)\log(kT)}/\varepsilon\right)$	$O(dT^{\alpha})$	(ε,δ)-DP
LDP-DPE	$O\left(T^{1-\alpha/2}\sqrt{\log(kT)} + d^{3/2}T^{1-\alpha/2}\sqrt{\ln(1/\delta)\log(kT)}/\varepsilon\right)$	$O(dT^{\alpha})$	(ε,δ)-LDP
SDP-DPE	$O\left(T^{1-\alpha/2}\sqrt{\log(kT)} + d^{3/2}T^{1-\alpha}\ln(d/\delta)\sqrt{\log(kT)}/\varepsilon\right)$	$O(dT^{3\alpha/2})$ (bits)	(ε,δ)-SDP

[1]DPE is the non-private DP-DPE algorithm; CDP-DPE, LDP-DPE, and SDP-DPE represent the DP-DPE algorithm in the central, local, and shuffle models, respectively, which guarantee (ε,δ)-DP, (ε,δ)-LDP, and (ε,δ)-SDP, respectively.
[2]In the regret upper bounds, we ignore lower-order terms for simplicity. T is the time horizon, k is the number of actions, d is the dimension of the action space, and α is a design parameter that can be used to tune the tradeoff between the regret and the communication cost.
[3]While the communication cost of CDP-DPE and LDP-DPE is measured in the number of real numbers transmitted between the clients and the server, SDP-DPE directly uses bits for reporting feedback. A detailed discussion is provided in Section 4.5.

setting. As in many distributed supervised learning problems [13, 61, 62], privacy protection is also of significant importance in our setting as clients' local feedback may contain their sensitive information. In summary, we are interested in the following fundamental question: *How to privately achieve global reward maximization with only partial distributed feedback?*

To that end, we introduce a new model called *differentially private distributed linear bandit (DP-DLB)*. In DP-DLB, there is a global linear bandit model $f(\mathbf{x}) = \langle \theta^*, \mathbf{x} \rangle$ with an unknown parameter $\theta^* \in \mathbb{R}^d$ at the central server (e.g., the BS); each user u of a large population has a local linear bandit model $f_u(\mathbf{x}) = \langle \theta_u, \mathbf{x} \rangle$, which represents the mean local reward for user u. Here, we assume that each user u has a local parameter $\theta_u \in \mathbb{R}^d$, motivated by the fact that the mean local reward (e.g., the expected throughput of a user under a certain network configuration) varies across the users. In addition, each local parameter θ_u is unknown and is assumed to be a realization of a random vector with the mean being the global model parameter θ^*. The server makes decisions based on the estimated global model, which can be learned through sampling a subset of users (referred to as clients) and iteratively aggregating these distributed partial feedback. While sampling more clients could improve the learning accuracy and thus lead to a better performance, it also incurs a higher communication

cost. Therefore, it is important to address this tradeoff in the design of communication protocols. Furthermore, to protect users' privacy, we resort to *differential privacy (DP)* to guarantee that clients' sensitive information will not be inferred by an adversary. Therefore, the goal is to maximize the cumulative global reward (or equivalently minimize the regret due to not choosing the optimal action in hindsight) in a communication-efficient manner while providing privacy guarantees for the participating clients. Our main contributions are summarized as follows.

- We present the first work that considers global reward maximization with partial feedback in the distributed linear bandit setting. In addition to the traditional tradeoff between exploitation and exploration, learning with distributed feedback introduces two practical challenges: communication efficiency and privacy concerns. This adds an extra layer of difficulty in the design of learning algorithms.
- To address these challenges, we introduce a DP-DLB model and develop a carefully-crafted algorithmic learning framework, called differentially private distributed phased elimination (DP-DPE), which allows the server and the clients to work in concert and can be naturally integrated with several state-of-the-art DP trust models (including central model, local model, and shuffle model). This unified framework enables us to systemically study the key regret-communication-privacy tradeoff.
- We then establish the regret-communication-privacy tradeoff of DP-DPE in various settings including the non-private case as well as the central, local, and shuffle DP models. Our main results are summarized in Table 4.1. These results reveal that DP-DPE achieves privacy "for-free" in the central and shuffle models, in the sense that the additional regret due to privacy protection is only a lower-order additive term. Moreover, this is the first work that considers the shuffle model in distributed linear bandits to attain a better regret-privacy tradeoff, i.e., guaranteeing a similar privacy protection as the strong local model

while achieving the same regret as the central model. We further perform simulations on synthetic data to corroborate our theoretical results.

- Finally, we provide an interesting discussion about achieving privacy "for free". We first highlight an interesting connection between our introduced DP-DLB formulation and the differentially private stochastic convex optimization (DP-SCO) problem in terms of achieving privacy "for-free". This bridge between our online bandit learning and the standard supervised learning might be of independent interest. Furthermore, with minor modifications of our developed techniques, we can establish that standard linear bandits can also achieve privacy "for-free" in the central and shuffle model.

4.2 Related Work

The bandit models (including linear bandits) and their variants have proven to be useful for many real-world applications and have been extensively studied (see, e.g., [18, 84, 124] and references therein). Most of the existing studies assume that the exact reward feedback is available to the learning agent for updating the model. However, there is a key difference in the new linear bandit setting we consider: while an action is taken at a central server, it influences a large population of users that contribute to the global reward, which, unfortunately, is not fully observable. Instead, one can learn the global model at the server by randomly sampling a subset of users from the population and iteratively aggregating such partial distributed feedback. While this setting shares some similarities with distributed bandits, federated bandits, and multi-agent cooperative bandits, our motivation and model are very different from theirs, which leads to different regret definitions (global regret vs. group regret; see Section 4.3) and algorithmic solutions. In the following, we discuss the most relevant work in the literature and highlight the key differences.

Linear bandits. While the stochastic multi-armed bandits (MAB) model has been extensively studied for a wide range of applications, its modeling power is limited by the assumption that actions are independent. In contrast, the linear bandit model captures the correlation among actions via an unknown parameter [4, 42, 116]. The best-known regret upper bound for stochastic linear bandits is $O(d\sqrt{T\log(T)})$ in [4], which holds for an almost arbitrary, even infinite, bounded subset of a finite-dimensional vector space. For a special setting where the set of actions is finite and does not change over time, it is shown in [84] that a *phased elimination with G-optimal exploration* algorithm guarantees a regret upper bounded by $O(\sqrt{dT\log(kT)})$. This new bound is better by a factor of $\sqrt{d}$, which deserves the effort when $d \geqslant \log(k)$. However, none of these studies consider the scenario where an action influences a large population and the exact reward feedback is unavailable, which is a key challenge in our problem. Note that the linear bandits model we consider is different from the contextual linear bandits in [39, 90] where the parameter is not shared by actions (although assuming linear reward function), and thus, the actions are not correlated through the parameter.

Differentially private online learning and bandits. Since proposed in[51], differential privacy (DP) has become the *de facto* privacy preserving model in many applications, including online learning [69] and bandits problems [101]. Specifically, in [115, 129, 131], MAB has been studied in the central, local, and shuffle DP models, respectively. In [120], the authors explore DP in contextual linear bandits and introduce joint DP as ensuring the standard DP incurs a linear regret. As stronger privacy protection, local DP is also studied for contextual linear bandits [142] and Bayesian optimization [144]. Very recently, shuffle model for linear *contextual* bandits have been studied in [38]. As already highlighted in Remark 4.17, the additional protection of context information leads to a higher cost of privacy compared to linear bandits considered in this chapter, where only rewards are private information.

Distributed bandits. Another line of related work is on multi-agent collaborative learning in the distributed bandits setting [5, 27, 47, 48, 98, 137]. The most relevant work to ours is the distributed linear bandit problem studied in [137]. Similarly, they design a distributed phased elimination algorithm where a central server aggregates data provided by the local clients and iteratively eliminates suboptimal actions. However, there are two key differences: i) they consider the standard group regret minimization problem with homogeneous clients that have the same unknown parameter; ii) the clients send the rewards to the central server without any data privacy protection.

Federated bandits. Federated learning (FL) has received substantial attention since its introduction in [99]. The main idea of FL is to enable collaborative learning among heterogeneous devices while preserving data privacy. Very recently, bandit problems have also been studied in the federated setting, including *federated multi-armed bandits* [121, 122, 146], *federated linear bandits* [46, 68], and *federated Bayesian optimization* [40, 41]. Among all the above work, the two most relevant studies are [68] and [46]. While they both consider the case where all heterogeneous users share the same unknown parameter with heterogeneous decision sets, in our problem of global reward maximization, the users have heterogeneous unknown local parameters.

In addition to the differences in model and problem formulation, we also highlight our main technical contributions compared to these works in the following. While a phased elimination algorithm is also employed in [68], there are two key differences: i) They do not consider the correlation among the actions. That is, the linear bandits setting plays a different role in their work. Specifically, they consider a linear reward for contextual bandits while still studying multi-armed bandits with independent actions, each of which is associated with a distinct parameter vector. Differently, the linear bandits formulation in our work is used to capture the correlation among the actions and can be extended to the case with an infinite number of

actions; ii) When aggregating users' data for learning the global parameter, we protect users' data privacy using rigorous differential privacy guarantees, which, however, is not considered in their design. While DP is also employed to protect users' data privacy in [46], they require that both the Gram matrix of actions (of size $O(d^2)$) and reward vectors (of size $O(d)$) be periodically communicated using some DP mechanisms (e.g., the Gaussian mechanism). Instead, in our algorithm, only private average local reward for the chosen actions (of size $O(d \log\log d)$) would be communicated in each phase. Moreover, while they only consider a variant of the central DP model, our DP-DPE solution provides a unified algorithmic learning framework, which can be instantiated with different DP models. Specifically, DP-DPE with the shuffle model enables us to achieve a finer regret-privacy-communication tradeoff (see Table 4.1). That is, not only can it achieve nearly the same regret performance as the central model (yet without trusting the central server), but it requires the users to report feedback in bits only throughout the learning process.

Discussion. One may wonder whether we can follow the idea of federated learning to share clients' locally learned model parameters only. This way, one can avoid sharing raw data, which is another way of protecting clients' data privacy. However, we argue that the additional benefit is marginal. On the one hand, by employing different DP mechanisms, our proposed DP-DPE algorithms already ensure provable privacy guarantees. On the other hand, the communication cost of transmitting the (private) average rewards is nearly the same as that of transmitting the local model parameters. Specifically, in each phase, a client in our DP-DPE algorithm needs to send a $|\text{supp}(\pi_l)|$-dimensional vector in DP-DPE, compared to a d-dimensional vector when sending the local model parameters. Therefore, the difference is really marginal since we have $|\text{supp}(\pi_l)| \leqslant 4d \log\log d + 16$.

Some perceptive readers might think reducing the model to a problem where each user u can observe *i.i.d.* rewards with mean $\langle \theta^*, x \rangle$ by treating $\langle \theta_u - \theta^*, x \rangle$ as an additional noise to η_t.

Note that the uncertainty introduced by this noise has to be addressed through sampling enough clients, e.g., one client per round. Considering DP, this problem essentially reduces to the differential private linear bandit in our Appendix C.3 (with a larger noise variance), where the same results in terms of regret (order-wise) and privacy can be achieved. However, one new user is sampled in each round to collect reward observation, which requires $O(T)$ users in total to obtain the optimal regret while ensuring the privacy guarantee. Instead, the DP-DPE framework in this work provides an approach where it collects feedback from multiple users for the selected action in each round while each user serves for multiple rounds to maintain (or improve) sample efficiency. Specifically, it samples $2^{\alpha l}$ clients for 2^l plays (rounds in the l-th phase), which is T^α users in total. In addition, by only collecting feedback after preprocessing reward observations at the end of each phase, this carefully designed phased elimination algorithm in different DP models reduces the communication cost from $O(T)$ to $O(dT^\alpha)$. We have to mention that choosing $\alpha < 1$, however, will incur a larger privacy cost (see Theorem 4.11, D.10, and D.11 with $\sigma = 0$). Therefore, there is a tradeoff between the regret penalty due to privacy and the communication and sampling efficiency, which can be balanced by tuning α properly.

4.3 System Model and Problem Formulation

4.3.1 Global Reward Maximization with Partial Feedback

We consider the global reward maximization problem over a large population, which is a sequential decision making problem. In each round t, the learning agent (e.g., the BS or the policy maker) selects an action $\mathbf{x}_t$ from a finite decision set $\mathcal{D} \subseteq \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\|_2^2 \leqslant 1\}$ with $|\mathcal{D}| = k$. This action leads to a global reward with mean $\langle \theta^*, \mathbf{x}_t \rangle$, where $\theta^* \in \mathbb{R}^d$ with

$\|\theta^*\|_2 \leqslant 1$ is unknown to the agent. This global reward captures the overall effectiveness of action $\mathbf{x}_t$ over a large population $\mathcal{U}$. The local reward of action $\mathbf{x}_t$ at user u has a mean $\langle \theta_u, \mathbf{x}_t \rangle$, where $\theta_u \in \mathbb{R}^d$ is the local parameter, which is assumed to be a realization of a random vector with mean θ^* and is also unknown. Let $\mathbf{x}^* \triangleq \text{argmax}_{\mathbf{x} \in \mathcal{D}} \langle \theta^*, \mathbf{x} \rangle$ be the unique global optimal action. Then, the objective of the agent is to maximize the cumulative global reward, or equivalently, to minimize the regret defined as follows:

$$R(T) \triangleq T \langle \theta^*, \mathbf{x}^* \rangle - \sum_{t=1}^{T} \langle \theta^*, \mathbf{x}_t \rangle. \tag{4.1}$$

At first glance, standard linear bandit algorithms (e.g., LinUCB in [90]) can be applied to addressing the above problem. However, the exact reward here is a global quantity, which is the average over the entire population. The learning agent may not be able to observe this exact reward, since collecting such global information from the entire population incurs a prohibitively high cost, is often impossible to implement in practice, and could lead to privacy concerns.

4.3.2 Differentially Private Distributed Linear Bandits

To address the above problem, we consider a *differentially private distributed linear bandit (DP-DLB)* formulation, where there are two important entities: a central server (which wants to learn the global model) and participating clients (i.e., a subset of users from the population who are willing to share their feedback). In the following, we discuss important aspects of the DP-DLB formulation.

Server. The server aims to learn the global linear bandit model, i.e., unknown parameter θ^*. In each round t, it selects an action x_t with the objective of maximizing the cumulative global

reward $\sum_{t=1}^{T}\langle\theta^*, \mathbf{x}_t\rangle$. Without observing the exact reward of action x_t, the server collects only partial feedback from a subset of users sampled from the population, called *clients*, and then aggregates this partial feedback to update the estimate of the global parameter θ^*. Based on the updated model, the server chooses an action in the next round.

Clients. We assume that each participating client is randomly sampled from the population and is independent from each other and also from other randomness. Specifically, we assume that local parameter θ_u at client u satisfies $\theta_u = \theta^* + \xi_u$, where $\xi_u \in \mathbb{R}^d$ is a zero-mean σ-sub-Gaussian random vector[4] and is independently and identically distributed (*i.i.d.*) across all clients. Let U_t be the set of clients in round t. After action x_t is chosen by the server in round t, each client $u \in U_t$ observes a noisy local reward: $y_{u,t} = \langle\theta_u, \mathbf{x}_t\rangle + \eta_{u,t}$, where $\eta_{u,t}$ is a conditionally 1-sub-Gaussian[5] noise and *i.i.d.* across the clients and over time. We also assume that the local rewards are bounded, i.e., $|y_{u,t}| \leqslant B$, for all $u \in \mathcal{U}$ and $t \in [T]$.

Communication. The communication happens when the clients report their feedback to the server. At the beginning of each communication step, each participating client reports feedback to the server based on the local reward observations during a certain number of rounds. In particular, the time duration between reporting feedback is called a phase. By aggregating such feedback from the clients, the server estimates the global parameter θ^* and adjusts its decisions in the following rounds accordingly. We assume that the clients do not quit before a phase ends. By slightly abusing the notation, we use U_l to denote the set of clients in the l-th phase.

The communication cost is a critical factor in DP-DLB. As in [137], we define the communication cost as the total number of real numbers (or bits, depending on the adopted DP

[4]A random vector $\xi \in \mathbb{R}^d$ is said to be σ-sub-Gaussian if $\mathbb{E}[\xi] = 0$ and $v^\top\xi$ is σ-sub-Gaussian for any unit vector $v \in \mathbb{R}^d$ and $\|v\|_2 = 1$ [20].

[5]Consider noise sequence $\{\eta_t\}_{t=1}^{\infty}$. As in the general linear bandit model [84], η_t is assumed to be conditionally 1-sub-Gaussian, meaning $\mathbb{E}[e^{\lambda\eta_t}|x_{1:t}, \eta_{1:t}] \leqslant \exp(\lambda^2/2)$ for all $\lambda \in \mathbb{R}$, where $a_{i:j}$ denotes the subsequence $a_i, \ldots, a_j$.

model) communicated between the server and the clients. Let L be the number of phases in T rounds, and let N_l be the number of real numbers (or bits) communicated in the l-th phase. Then, the total communication cost, denoted by $C(T)$, is

$$C(T) \triangleq \sum_{l=1}^{L} |U_l| N_l. \tag{4.2}$$

Data privacy. In practice, even if users are willing to share their feedback, they typically require privacy protection as a premise. To that end, we resort to *differential privacy (DP)* [51] to formally address the privacy concerns in the learning process. More importantly, instead of only considering the standard central model where the central server is responsible for protecting the privacy, we will also incorporate other popular DP models, including the stronger local model (where each client directly protects her data) [76] and the recently proposed shuffle model (where a trusted shuffler between clients and server is adopted to amplify privacy) [34], in a unified algorithmic learning framework.

4.4 Algorithm Design

In this section, we first present the key challenges associated with the introduced DP-DLB model and then explain how the developed DP-DPE framework addresses these challenges.

4.4.1 Key Challenges

To solve the problem of global reward maximization with partial distributed feedback using the DP-DLB formulation, we face four key challenges, discussed in detail below.

As in the standard stochastic bandits problem, there is an uncertainty due to noisy rewards

of each chosen action, which is called the *action-related uncertainty*. In addition to this, we face another type of uncertainty related to the sampled clients in DP-DLB, called the *client-related uncertainty*. The client-related uncertainty lies in estimating the global model at the server based on randomly sampled clients with *biased* local models. Note that the global model may not be accurately estimated even if exact rewards of the sampled clients are known when the number of clients is insufficient. Therefore, the first challenge lies in *simultaneously addressing both types of uncertainty in a sample-efficient way* (Challenge ⓐ).

To handle the newly introduced client-related uncertainty, we must sample a sufficiently large number of clients so that the global parameter can be accurately estimated using the partial distributed feedback. However, too many clients result in a large communication cost (see Eq. (4.2)). Therefore, the second challenge is to *decide the number of sampled clients to balance the regret (due to the client-related uncertainty) and the communication cost* (Challenge ⓑ).

Finally, to ensure privacy guarantees for the clients, one needs to add additional perturbations (or noises) to the local feedback. *Such randomness introduces another type of uncertainty to the learning process* (Challenge ⓒ), and *it is unclear how to integrate different trust DP models into a unified algorithmic learning framework* (Challenge ⓓ). These add an extra layer of difficulty to the design of learning algorithms.

Main ideas. In the following, we present our main ideas for addressing the above challenges. We propose a phased elimination algorithm that gradually eliminates suboptimal actions by periodically aggregating the local feedback from the sampled clients in a privacy-preserving manner. To address the multiple types of uncertainty when estimating the global reward (ⓐ and ⓒ), we carefully construct a confidence width to incorporate all three types of uncertainty. To achieve a sublinear regret while saving communication cost (ⓑ), we increase both the phase length and the number of clients exponentially. To ensure privacy guarantees

(ⓓ), we introduce a Privatizer that can be easily tailored under different DP models. The Privatizer is a process consisting of tasks to be collaboratively completed by the clients, the server, and/or even a trusted third party. To keep it general, we use $\mathcal{P} = (\mathcal{R}, \mathcal{S}, \mathcal{A})$ to denote a Privatizer, where $\mathcal{R}$ is the procedure at each client (usually a local randomizer), $\mathcal{S}$ is a trusted third party that helps privatize data (e.g., a shuffler that permutes received messages), and $\mathcal{A}$ is an analyzer operated at the central server. Next, we will show how to integrate these main ideas into a unified algorithmic learning framework.

4.4.2 Differentially Private Distributed Phased Elimination (DP-DPE)

With the main ideas presented above, we now propose a unified algorithmic learning framework, called *differentially private distributed phased elimination (DP-DPE)*, which is presented in Algorithm 5. The DP-DPE runs in phases and operates with the coordination of the central server and the participating clients in a synchronized manner. At a high level, each phase consists of the following three steps:

- **Action selection (Lines 4-6):** computing a near-G-optimal design (i.e., a distribution) over a set of possibly optimal actions and playing these actions;
- **Clients sampling and private feedback aggregation (Lines 7-16):** sampling participating clients and aggregating their local feedback in a privacy-preserving fashion;
- **Parameter estimation and action elimination (Lines 17-20):** using (privately) aggregated data to estimate θ^* and eliminating actions that are likely to be suboptimal.

In the following, we describe the detailed operations of DP-DPE. We begin by giving some necessary notations. Consider the l-th phase. Let t_l and T_l be the index of the starting round

Algorithm 5 Differentially Private Distributed Phased Elimination (DP-DPE)

Input: $\mathcal{D} \subseteq \mathbb{R}^d$, $\alpha \in (0,1)$, $\beta \in (0,1)$, and σ_n
Initialization: $l = 1$, $t_1 = 1$, $\mathcal{D}_1 = \mathcal{D}$, and $h_1 = 2$
while $t_l \leqslant T$ **do**
  Find a distribution $\pi_l(\cdot)$ over $\mathcal{D}_l$ such that $g(\pi_l) \triangleq \max_{\mathbf{x}\in\mathcal{D}_l} \|\mathbf{x}\|^2_{V(\pi_l)^{-1}} \leqslant 2d$ and $|\text{supp}(\pi_l)| \leqslant 4d \log\log d + 16$, where $V(\pi_l) \triangleq \sum_{\mathbf{x}\in\mathcal{D}_l} \pi_l(\mathbf{x})\mathbf{x}\mathbf{x}^\top$
  Let $T_l(\mathbf{x}) = \lceil h_l \pi_l(\mathbf{x}) \rceil$ for each $x \in \text{supp}(\pi_l)$ and $T_l = \sum_{x\in\text{supp}(\pi_l)} T_l(\mathbf{x})$
  Play each action $\mathbf{x} \in \text{supp}(\pi_l)$ exactly $T_l(\mathbf{x})$ times if not reaching T
  Randomly select $\lceil 2^{\alpha l} \rceil$ participating clients U_l
  # Operations at each client
  for each client $u \in U_l$ **do**
    for each action $\mathbf{x} \in \text{supp}(\pi_l)$ **do**
      Compute average local reward over $T_l(\mathbf{x})$ rounds: $y_l^u(\mathbf{x}) = \frac{1}{T_l(\mathbf{x})} \sum_{t\in\mathcal{T}_l(\mathbf{x})} (\langle \theta_u, x\rangle + \eta_{u,t})$
    end for
    Let $\vec{y}_l^u = (y_l^u(\mathbf{x}))_{x\in\text{supp}(\pi_l)}$
    # Apply the PRIVATIZER $\mathcal{P} = (\mathcal{R}, \mathcal{S}, \mathcal{A})$
    # The local randomizer $\mathcal{R}$ at each client:
    Run the local randomizer $\mathcal{R}$ and send the output $\mathcal{R}(\vec{y}_l^u)$ to $\mathcal{S}$
  end for
  # Computation $\mathcal{S}$ at a trusted third party:
  Run the computation function $\mathcal{S}$ and send the output $\mathcal{S}(\{\mathcal{R}(\vec{y}_l^u)\}_{u\in U_l})$ to the analyzer $\mathcal{A}$
  # The analyzer $\mathcal{A}$ at the server:
  Generate the privately aggregated statistics: $\tilde{y}_l = \mathcal{A}(\mathcal{S}(\{\mathcal{R}(\vec{y}_l^u)\}_{u\in U_l}))$
  Compute the following quantities:

$$\begin{cases} V_l = \sum_{x\in\text{supp}(\pi_l)} T_l(\mathbf{x})\mathbf{x}\mathbf{x}^\top \\ G_l = \sum_{x\in\text{supp}(\pi_l)} T_l(\mathbf{x}) x \tilde{y}_l(\mathbf{x}) \\ \tilde{\theta}_l = V_l^{-1} G_l \end{cases}$$

  Find low-rewarding actions with confidence width W_l:

$$E_l = \left\{ \mathbf{x} \in \mathcal{D}_l : \max_{\mathbf{b}\in\mathcal{D}_l} \langle \tilde{\theta}_l, \mathbf{b} - \mathbf{x} \rangle > 2W_l \right\}$$

  Update: $\mathcal{D}_{l+1} = \mathcal{D}_l \backslash E_l$, $h_{l+1} = 2h_l$, $t_{l+1} = t_l + T_l$, and $l = l + 1$
end while

and the length of the l-th phase, respectively. Then, let $\mathcal{T}_l \triangleq \{t \in [T] : t_l \leqslant t < t_l + T_l\}$ be the round indices in the l-th phase, let $\mathcal{T}_l(\mathbf{x}) \triangleq \{t \in \mathcal{T}_l : \mathbf{x}_t = \mathbf{x}\}$ be the time indices in the l-th phase when action x is selected, and let $\mathcal{D}_l \subseteq \mathcal{D}$ be the set of active actions in the l-th phase.

Action selection (Lines 4-6): In the l-th phase, the action set $\mathcal{D}_l$ consists of active actions that are possibly optimal. We compute a distribution $\pi_l(\cdot)$ over $\mathcal{D}_l$ and choose actions according to $\pi_l(\cdot)$. We briefly explain the intuition below. Let $V(\pi) \triangleq \sum_{\mathbf{x}\in\mathcal{D}} \pi(\mathbf{x})\mathbf{x}\mathbf{x}^\top$ and $g(\pi) \triangleq \max_{x\in\mathcal{D}} \|\mathbf{x}\|^2_{V(\pi)^{-1}}$. According to the analysis in [84, Chapter 21], if action $\mathbf{x} \in \mathcal{D}$ is played $\lceil h\pi(\mathbf{x})\rceil$ times (where h is a positive constant), the estimation error associated with the action-related uncertainty for action x is at most $\sqrt{2g(\pi)\log(1/\beta)/h}$ with probability $1-\beta$ for any $\beta \in (0,1)$. That is, for a fixed number of rounds, a distribution $\pi(\cdot)$ with a smaller value of $g(\pi)$ helps achieve a better estimation. Note that minimizing $g(\cdot)$ is a well-known *G-optimal design* problem [110]. By the Kiefer-Wolfowitz Theorem [78], one can find a distribution π^* minimizing $g(\cdot)$ with $g(\pi^*) = d$, and the support set[6] of π^*, denoted by $\text{supp}(\pi^*)$, has a size no greater than $d(d+1)/2$. In our problem, however, it suffices to solve it near-optimally, i.e., finding a distribution π_l such that $g(\pi_l) \leqslant 2d$ with $|\text{supp}(\pi_l)| \leqslant 4d\log\log d + 16$ (Line 4), which follows from [85, Proposition 3.7]. The near-G-optimal design reduces the complexity to $O(kd^2)$ while keeping the same order of regret.

Clients sampling and private feedback aggregation (Lines 7-16): The central server randomly samples a subset U_l of $\lceil 2^{\alpha l}\rceil$ users (called clients) from the population $\mathcal{U}$ to participate in the global bandit learning (Line 7). Each sampled client $u \in U_l$ collects their local reward observations of each chosen action $\mathbf{x} \in \text{supp}(\pi_l)$ by the server and computes the average $y_l^u(\mathbf{x})$ as feedback (Line 10). Before being used to estimate the global parameter

[6]The support set of a distribution π over set $\mathcal{D}$, denoted by $\text{supp}_{\mathcal{D}}(\pi)$, is the subset of elements with a nonzero $\pi(\cdot)$, i.e., $\text{supp}_{\mathcal{D}}(\pi) \triangleq \{x \in \mathcal{D} : \pi(\mathbf{x}) \neq 0\}$. We drop the subscript $\mathcal{D}$ in $\text{supp}_{\mathcal{D}}(\pi)$ for notational simplicity.

by the central server, these feedback $\vec{y}_l^u \triangleq (y_l^u(\mathbf{x}))_{x\in\text{supp}(\pi_l)} \in \mathbb{R}^{|\text{supp}(\pi_l)|}$ are processed by a PRIVATIZER $\mathcal{P}$ to ensure differential privacy. Recall that a PRIVATIZER $\mathcal{P} = (\mathcal{R}, \mathcal{S}, \mathcal{A})$ is a process completed by the clients, the server, and/or a trusted third party. In particular, according to the privacy requirement under different DP models, the PRIVATIZER $\mathcal{P}$ enjoys flexible instantiations (see detailed discussions in Section 4.5). Generally, a PRIVATIZER works in the following manner: each client u runs the randomizer $\mathcal{R}$ on its local average reward $\vec{y}_l^u$ (over T_l pulls) and then sends the resulting (potentially private) messages $\mathcal{R}(\vec{y}_l^u)$ to $\mathcal{S}$ (Line 13). The computation function in $\mathcal{S}$ operates on these messages and then sends results $\mathcal{S}(\{\mathcal{R}(\vec{y}_l^u)\}_{u\in U_l})$ to the analyzer $\mathcal{A}$ at the central server (Line 15). Finally, the analyzer $\mathcal{A}$ aggregates received messages (potentially in a privacy-preserving manner) and outputs a private averaged local reward $\tilde{y}_l(\mathbf{x})$ (over participating clients U_l) for each action $\mathbf{x} \in \text{supp}(\pi_l)$ (Line 16). We provide the rigorous formulation of different DP models for PRIVATIZER $\mathcal{P}$ in Section 4.5, with corresponding detailed instantiations of $\mathcal{R}, \mathcal{S}$, and $\mathcal{A}$.

Parameter estimation and action elimination (Lines 17-20): Using privately aggregated feedback (i.e., the private averaged local reward $\tilde{y}_l$ of the chosen actions $\mathbf{x} \in \text{supp}(\pi_l)$), the central server computes the least-square estimator $\tilde{\theta}_l$ (Line 17). Action elimination is based on the following confidence width:

$$W_l \triangleq \left(\underbrace{\sqrt{\frac{2d}{|U_l|h_l}}}_{\text{action-related}} + \underbrace{\frac{\sigma}{\sqrt{|U_l|}}}_{\text{client-related}} + \underbrace{\sigma_n}_{\text{privacy noise}} \right) \sqrt{2\log\left(\frac{1}{\beta}\right)}, \tag{4.3}$$

where σ is the standard variance associated with client sampling, σ_n is related to the privacy noise determined by the DP model, and β is the confidence level. We choose this confidence width based on the concentration inequality for sub-Gaussian variables. Specifically, the three terms in Eq. (4.3) capture the action-related uncertainty, client-related uncertainty,

and the added noise for privacy guarantees, respectively. This privacy noise σ_n depends on the adopted DP model. Using this confidence width W_l and the estimated global model parameter $\tilde{\theta}_l$, we can identify a subset of suboptimal actions E_l with high probability (Line 18). At the end of the l-th phase, we update the set of active actions $\mathcal{D}_{l+1}$ by eliminating E_l from $\mathcal{D}_l$ and double h_l (Line 20).

Finally, we make two remarks about the DP-DPE algorithm.

Remark 4.1. While a finite number of actions is assumed in this work, one could extend it to the case with an infinite number of actions by using the covering argument [84, Lemma 20.1]. Specifically, when the action set $\mathcal{D} \subseteq \mathbb{R}^d$ is infinite, we can replace $\mathcal{D}$ with a finite set $\mathcal{D}_{\varepsilon_0} \subseteq \mathbb{R}^d$ with $|\mathcal{D}_{\varepsilon_0}| \leqslant (3/\varepsilon_0)^d$ such that for all $\mathbf{x} \in \mathcal{D}$, there exists an $\mathbf{x}' \in \mathcal{D}_{\varepsilon_0}$ with $\|\mathbf{x} - \mathbf{x}'\|_2 \leqslant \varepsilon_0$.

Remark 4.2. In Algorithm 5, we assume that $\mathcal{D}_l$ spans $\mathbb{R}^d$ such that matrices $V(\pi_l)$ and V_l are invertible. Then, one could find the near optimal design $\pi_l(\cdot)$ (Line 4) and compute the least-square estimator $\tilde{\theta}_l$ (Line 17). When $\mathcal{D}_l$ does not span $\mathbb{R}^d$, one can simply work in the smaller space $\text{span}(\mathcal{D}_l)$ [85].

4.5 DP-DPE under Different DP Models

As alluded before, one of the key features of our general algorithmic framework DP-DPE is that it enables us to consider different trust models in DP (i.e., who the user can trust with her sensitive data) in a unified way by instantiating different mechanisms for the Privatizer. In this section, we formalize DP models integrated with our DP-DLB formulation and provide concrete instantiations for the Privatizer $\mathcal{P} = (\mathcal{R}, \mathcal{S}, \mathcal{A})$ in DP-DPE according to three representative DP trust models: the central, local, and shuffle models.

4.5.1 DP-DPE under the Central DP Model

In the central DP model, we assume that each client trusts the server, and hence, the server can collect clients' raw data (i.e., the local reward $y_l^u(\mathbf{x})$ for each chosen action x in our case). The privacy guarantee is that any adversary with arbitrary auxiliary information cannot infer a particular client's data by observing the outputs of the server. To achieve this privacy protection, the central DP model requires that the outputs of the server on two neighboring datasets differing in only one client are indistinguishable [51]. To present the formal definition in our case, recall that the DP-DPE algorithm (Algorithm 5) runs in phases, and in each phase l, a set of new clients U_l will participate in the global bandit learning by providing their feedback. Let[7] $\mathcal{U}_T \triangleq (U_l)_{l=1}^L \in \mathcal{U}^*$ be the sequence of all the participating clients in the total L phases (T rounds). We use $\mathcal{M}(\mathcal{U}_T) = (\mathbf{x}_1, \ldots, \mathbf{x}_T) \in \mathcal{D}^T$ to denote the sequence of actions chosen in T rounds by the central server. Intuitively, we are interested in a randomized algorithm such that the output $\mathcal{M}(\mathcal{U}_T)$ does not reveal "much" information about any particular client $u \in \mathcal{U}_T$. Formally, we have the following definition.

Definition 4.3. (Differential Privacy (DP)). For any $\varepsilon \geqslant 0$ and $\delta \in [0, 1]$, a DP-DPE instantiation is (ε, δ)-differentially private (or (ε, δ)-DP) if for every $\mathcal{U}_T, \mathcal{U}'_T \subseteq \mathcal{U}$ differing on a single client and for any subset of actions $Z \subseteq \mathcal{D}^T$,

$$\mathbb{P}[\mathcal{M}(\mathcal{U}_T) \in Z] \leqslant e^{\varepsilon}\mathbb{P}[\mathcal{M}(\mathcal{U}'_T) \in Z] + \delta. \tag{4.4}$$

According to the post-processing property of DP (cf. Proposition 2.1 in [50]) and parallel-composition (thanks to the uniqueness of client sampling), it suffices to guarantee that the final analyzer $\mathcal{A}$ in $\mathcal{P}$ is (ε, δ)-DP. That is, for any phase l, the PRIVATIZER $\mathcal{P}$ is (ε, δ)-DP if the following is satisfied for any pair of $U_l, U'_l \subseteq \mathcal{U}$ that differ by at most one client and for

[7] We use the superscript * to indicate that the length could be varying.

any output $\tilde{y}$ of $\mathcal{A}$:

$$\mathbb{P}[\mathcal{A}(\{\vec{y}_l^u\}_{u\in U_l}) = \tilde{y}] \leqslant e^{\varepsilon} \cdot \mathbb{P}[\mathcal{A}(\{\vec{y}_l^u\}_{u\in U_l'}) = \tilde{y}] + \delta.$$

To achieve this, we resort to standard Gaussian mechanism at the server where $\mathcal{A}$ computes the average of local rewards for each chosen action to guarantee (ε, δ)-DP. Specifically, in each phase l, the participating clients send their average local rewards $\{\vec{y}_l^u\}_{u\in U_l}$ directly to the central server, and the central server adds Gaussian noise to the average local feedback (over clients) before estimating the global parameter and deciding the chosen actions in the next phase. That is, in the central DP model, both $\mathcal{R}$ and $\mathcal{S}$ of the Privatizer $\mathcal{P}$ are identity mapping while $\mathcal{A}$ adds Gaussian noise when computing the average. In this case, $\mathcal{P} = \mathcal{A}$, and the private aggregated feedback for the chosen actions in the l-th phase can be represented as

$$\tilde{y}_l = \mathcal{P}\left(\{\vec{y}_l^u\}_{u\in U_l}\right) = \mathcal{A}\left(\{\vec{y}_l^u\}_{u\in U_l}\right) = \frac{1}{|U_l|}\sum_{u\in U_l} \vec{y}_l^u + (\gamma_1, \ldots, \gamma_{s_l}), \tag{4.5}$$

where $s_l \triangleq |\text{supp}(\pi_l)|$, $\gamma_j \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma_{nc}^2)$, and the variance σ_{nc}^2 is based on the ℓ_2 sensitivity of the average $\frac{1}{|U_l|}\sum_{u\in U_l} \vec{y}_l^u$. In the rest of the chapter, we will continue to use s_l instead of $|\text{supp}(\pi_l)|$ to denote the number of actions chosen in the l-th phase for notational simplicity, and it is also the dimension of $\vec{y}_l^u$ for all u.

With the above definition, we present the privacy guarantee of DP-DPE in the central DP model in Theorem 4.4.

Theorem 4.4. *The DP-DPE instantiation using the* Privatizer *in Eq.* (4.5) *with* $\sigma_{nc} = \frac{2B\sqrt{2s_l \ln(1.25/\delta)}}{\varepsilon|U_l|}$ *guarantees* (ε, δ)*-DP.*

The relatively high trust model in the central DP is not always feasible in practice since some clients do not trust the server and are not willing to share any of their sensitive data.

This motivates the introduction of a strictly stronger notion of privacy protection called the local DP [76], which is the main focus of the next subsection.

4.5.2 DP-DPE under the Local DP Model

In the local DP model, the privacy burden is now at each client's local side, in the sense that any data sent by any client must already be private. In other words, even though an adversary can observe the data communicated from a client to the server, the adversary cannot infer any sensitive information about the client. Mathematically, this requires a local randomizer $\mathcal{R}$ at each user's side to generate approximately indistinguishable outputs on any two different data inputs. In particular, let Y_u be the set of all possible values of the average local reward $\vec{y}_l^u$ for client u. Then, we have the following formal definition.

Definition 4.5. (Local Differential Privacy (LDP)). For any $\varepsilon \geqslant 0$ and $\delta \in [0, 1]$, a DP-DPE instantiation is (ε, δ)-local differentially private (or (ε, δ)-LDP) if for any client u, every two datasets $\vec{y}, \vec{y}' \in Y_u$ satisfies

$$\mathbb{P}[\mathcal{R}(\vec{y}) = o] \leqslant e^{\varepsilon}\mathbb{P}[\mathcal{R}(\vec{y}') = o] + \delta, \tag{4.6}$$

for every possible output $o \in \{\mathcal{R}(\vec{y}) | \vec{y} \in Y_u\}$.

That is, an instantiation of DP-DPE is (ε, δ)-LDP if the local randomizer $\mathcal{R}$ in $\mathcal{P}$ is (ε, δ)-DP. To this end, the randomizer $\mathcal{R}$ at each client employs a Gaussian mechanism, the shuffler $\mathcal{S}$ is a simple identity mapping, and the analyzer $\mathcal{A}$ at the server side conducts a simple averaging. Then, the overall output of the PRIVATIZER is the following:

$$\tilde{y}_l = \frac{1}{|U_l|} \sum_{u \in U_l} \mathcal{R}(\vec{y}_l^u) = \frac{1}{|U_l|} \sum_{u \in U_l} \left(\vec{y}_l^u + (\gamma_{u,1}, \ldots, \gamma_{u,s_l})\right), \tag{4.7}$$

where $\gamma_{u,j} \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma_{nl}^2)$, and the variance σ_{nl}^2 is based on the sensitivity of $\vec{y}_l^u$.

With the above definition, we present the privacy guarantee of DP-DPE in the local DP model in Theorem 4.6.

Theorem 4.6. *The DP-DPE instantiation using the* Privatizer *in Eq. (4.7) with* $\sigma_{nl} = \frac{2B\sqrt{2s_l \ln(1.25/\delta)}}{\varepsilon}$ *guarantees* (ε, δ)*-LDP.*

Although the local DP model offers a stronger privacy guarantee compared to the central DP model, it often comes at a price of the regret performance. As we will see, the regret performance of DP-DPE under the local DP model is much worse than that under the central DP model. Therefore, a fundamental question is whether there is a Privatizer for DP-DPE that can achieve the same regret as in the central DP Privatizer while assuming similar trust model as in the local DP Privatizer. This motivates us to consider a recently proposed *shuffle DP model* [34, 54], which is the main focus of the next subsection.

4.5.3 DP-DPE under the Shuffle DP Model

In the shuffle DP model, between the clients and the server, there exists a shuffler that permutes a batch of clients' randomized data before they are observed by the server so that the server cannot distinguish between two clients' data. Thus, an additional layer of randomness is introduced via shuffling, which can often be easily implemented using cryptographic primitives (e.g., mixnets) due to its simple operation [14]. Due to this, the clients now tend to trust the shuffler but still do not trust the central server as in the local DP model. This new trust model offers a possibility to achieve a better regret-privacy tradeoff. This is because the additional randomness of the shuffler creates a *privacy blanket* so that by adding much less random noise, each client can now hide her information in the crowd, i.e., privacy amplification by shuffling [60].

Formally, a standard one-round shuffle protocol consists of all the three parts: a (local) randomizer $\mathcal{R}$, a shuffler $\mathcal{S}$, and an analyzer $\mathcal{A}$. In this protocol, the clients trust the shuffler but not the analyzer. Hence, the privacy objective is to ensure that the outputs of the shuffler on two neighboring datasets are indistinguishable from the analyzer's point of view. Note that each client still does not send her raw data to the shuffler even though she trusts it. Due to this, a shuffle protocol often also offers a certain level of LDP guarantee.

In our case, the online learning procedure will proceed in multiple phases rather than a simple one-round computation. Thus, we need to guarantee that all the shuffled outputs are indistinguishable. To this end, we define the (composite) mechanism $\mathcal{M}_s(\mathcal{U}_T) \triangleq ((\mathcal{S} \circ \mathcal{R})(U_1), (\mathcal{S} \circ \mathcal{R})(U_2), \ldots, (\mathcal{S} \circ \mathcal{R})(U_L))$, where $(\mathcal{S} \circ \mathcal{R})(U_l) \triangleq \mathcal{S}(\{\mathcal{R}(\vec{y}_l^u)\}_{u \in U_l})$. We say a DP-DPE instantiation satisfies the shuffle differential privacy (SDP) if the composite mechanism $\mathcal{M}_s$ is DP, which leads to the following formal definition.

Definition 4.7. (Shuffle Differential Privacy (SDP)). For any $\varepsilon \geqslant 0$ and $\delta \in [0, 1]$, a DP-DPE instantiation is (ε, δ)-shuffle differential privacy (or (ε, δ)-SDP) if for any pair $\mathcal{U}_T$ and $\mathcal{U}_T^{'}$ that differ by one client, the following is satisfied for all $Z \subseteq Range(\mathcal{M}_s)$:

$$\mathbb{P}[\mathcal{M}_s(\mathcal{U}_T) \in Z] \leqslant e^{\varepsilon} \mathbb{P}[\mathcal{M}_s(\mathcal{U}_T^{'}) \in Z] + \delta. \tag{4.8}$$

Then, consider any phase l. Formally, the PRIVATIZER $\mathcal{P}$ is (ε, δ)-SDP if the following is satisfied for any pair of U_l, $U_l' \subseteq \mathcal{U}$ that differ by one client and for any possible output z of $\mathcal{S} \circ \mathcal{R}$:

$$\mathbb{P}[(\mathcal{S} \circ \mathcal{R})(U_l) = z] \leqslant e^{\varepsilon} \cdot \mathbb{P}[(\mathcal{S} \circ \mathcal{R})(U_l') = z] + \delta.$$

We present the concrete pseudocode of $\mathcal{R}$, $\mathcal{S}$, and $\mathcal{A}$ for the shuffle DP model PRIVATIZER $\mathcal{P}$ in Algorithm 7 (see Appendix C.1), which builds on the vector summation protocol recently proposed in [35]. Here, we provide a brief description of the process. Essentially, the

noise added in the shuffle model **Privatizer** relies on the upper bound of ℓ_2 norm of the input vectors. However, each component operates on each coordinate of the input vectors independently. Recall that the input of the shuffle model **Privatizer** is $\{\vec{y}_l^u\}_{u\in U_l}$ and that each chosen action x corresponds to a coordinate in the s_l-dimentional vector. Consider the coordinate j_x corresponding to action x, and the entry $y_l^u(\mathbf{x})$ at client u. First, the local randomizer $\mathcal{R}$ encodes the input $y_l^u(\mathbf{x})$ via a fixed-point encoding scheme [34] and ensures privacy by injecting binomial noise. Specifically, given any scalar $w \in [0,1]$, it is first encoded as $\hat{w} = \bar{w} + \gamma_1$ using an accuracy parameter $g \in \mathbb{N}$, where $\bar{w} = \lfloor wg \rfloor$ and $\gamma_1 \sim \texttt{Ber}(wg - \bar{w})$ is a Bernoulli random variable. Then, a binomial noise $\gamma_2 \sim \texttt{Bin}(b,p)$ is generated, where $b \in \mathbb{N}$ and $p \in (0,1)$ controls the level of the privacy noise. The output of the local randomizer for each coordinate is simply a collection of $g+b$ bits, where $\hat{w}+\gamma_2$ bits are 1's and the rest are 0's. Combining these $g+b$ bits for each coordinate j_x for $\mathbf{x} \in \text{supp}(\pi_l)$ yields the final outputs of the local randomizer $\mathcal{R}$ for the vector $\vec{y}_l^u$. Note that the output bits for each coordinate are marked with the coordinate index so that they will not be mixed up in the following procedures. After receiving the bits from all participating clients, the shuffler $\mathcal{S}$ simply permutes these bits uniformly at random and sends the output to the analyzer $\mathcal{A}$ at the central server. The analyzer $\mathcal{A}$ adds the received bits, removes the bias introduced by encoding and binomial noise (through simple shifting operations), and divides the result by $|U_l|$ for each coordinate. Finally, the analyzer $\mathcal{A}$ outputs a random s_l-dimensional vector $\tilde{y}_l$, whose expectation is the average of the input vectors. That is, $\mathbb{E}[\tilde{y}_l] = \frac{1}{|U_l|}\sum_{u\in U_l}\vec{y}_l^u$ (which is proven in Appendix C.1.3). In the shuffle model **Privatizer**, the three parameters g, b, and p need to be properly chosen according to the privacy requirement. Then, the final privately aggregated data is the following:

$$\tilde{y}_l = \mathcal{P}\left(\{\vec{y}_l^u\}_{u\in U_l}\right) = \mathcal{A}(\mathcal{S}(\{\mathcal{R}(\vec{y}_l^u)\}_{u\in U_l})). \tag{4.9}$$

With the above definition, we present the privacy guarantee of DP-DPE in the shuffle DP model in Theorem 4.8.

Theorem 4.8. *For any $\varepsilon \in (0, 15)$ and $\delta \in (0, 1/2)$, the DP-DPE instantiation using the* PRIVATIZER *specified in Algorithm 7 guarantees (ε, δ)-SDP.*

4.6 Main Results

In this section, we study the performance of DP-DPE under different DP models in terms of regret and communication cost. We start with the non-private DP-DPE algorithm (called DPE, with $\tilde{y}_l = \frac{1}{|U_l|}\sum_{u \in U_l} \vec{y}_l^u$ and $\sigma_n = 0$ for all l) and present the main result in Theorem 4.9.

Theorem 4.9 (DPE)**.** *Let $\beta = 1/(kT)$ and $\sigma_n = 0$ in Algorithm 5. Then, the non-private DP-DPE algorithm achieves the following expected regret:*

$$\mathbb{E}[R(T)] = O(\sqrt{dT\log(kT)}) + O\left(\sigma T^{1-\alpha/2}\sqrt{\log(kT)}\right), \tag{4.10}$$

with a communication cost of $O(dT^{\alpha})$.

We present a proof sketch below and provide the detailed proof in Appendix C.2.

Proof sketch. We begin by considering a concentration inequality $P\left\{\langle \tilde{\theta}_l - \theta^*, x\rangle \geqslant W_l\right\} \leqslant 2\beta$, which indicates that in the l-th phase, the estimation error for the global reward of each action is bound by W_l *w.h.p.* Then, we show that the optimal action stays in the active set the whole time *w.h.p.* and that the regret incurred by one pull is bounded by $4W_{l-1}$ in the l-th phase. Finally, summing up the regret over rounds in all phases, we derive the regret upper bound. The analysis of the communication cost is quite straightforward. In the l-th phase, only local average reward of each chosen action in this phase is communicated.

Since the number of chosen actions is bounded by $(4d \log\log d + 16)$ according to the near-G-optimal design [85, Proposition 3.7], the communication cost is proportional to the total number of clients involved in the entire learning process. □

Remark 4.10. Theorem 4.9 gives a problem-independent regret upper bound for DPE. We can observe an obvious tradeoff between regret and communication cost, captured by the value of α. While a larger α leads to a smaller regret, it also incurs a larger communication cost. Setting $\alpha = 2/3$ gives $O(T^{2/3})$ for both regret and communication cost.

In the following, we present the performance of DP-DPE in terms of regret and communication cost under different DP models, i.e., instantiated with different Privatizers introduced in Section 4.5. We use CDP-DPE, LDP-DPE, and SDP-DPE to denote the DP-DPE algorithm in the central, local, and shuffle DP models, respectively. Let $S \triangleq 4d \log\log d + 16$ denote the upper bound of $|\text{supp}(\pi_l)|$ in every phase l.

Theorem 4.11 (CDP-DPE)**.** *Consider the Gaussian mechanism with* $\sigma_{nc} = \frac{2B\sqrt{2s_l \ln(1.25/\delta)}}{\varepsilon |U_l|}$ *in the central DP model. With* $\sigma_n = 2\sigma_{nc}\sqrt{Sd}$ *and* $\beta = 1/(kT)$*, CDP-DPE achieves the following expected regret:*

$$\begin{aligned}\mathbb{E}[R(T)] =& O(\sqrt{dT\log(kT)}) + O(\sigma T^{1-\alpha/2}\sqrt{\log(kT)}) \\ &+ O\left(\frac{Bd^{3/2}T^{1-\alpha}\sqrt{\ln(1/\delta)\log(kT)}}{\varepsilon}\right) + O\left(\frac{Bd^{5/2}\sqrt{\ln(1/\delta)\log(kT)}}{\varepsilon}\right),\end{aligned} \tag{4.11}$$

with a communication cost of $O(dT^{\alpha})$*.*

Theorem 4.12 (LDP-DPE)**.** *Consider the Gaussian mechanism with* $\sigma_{nl} = \frac{2B\sqrt{2s_l \ln(1.25/\delta)}}{\varepsilon}$ *in the local DP model. With* $\sigma_n = 2\sigma_{nl}\sqrt{\frac{Sd}{|U_l|}}$ *in the* l*-th phase and* $\beta = 1/(kT)$*, LDP-DPE*

achieves the following expected regret:

$$\begin{aligned}\mathbb{E}[R(T)] =& O(\sqrt{dT\log(kT)}) + O\left(\sigma T^{1-\alpha/2}\sqrt{\log(kT)}\right) \\ &+ O\left(\frac{Bd^{3/2}T^{1-\alpha/2}\sqrt{\ln(1/\delta)\log(kT)}}{\varepsilon}\right) + O\left(\frac{Bd^{5/2}\sqrt{\ln(1/\delta)\log(kT)}}{\varepsilon}\right),\end{aligned} \tag{4.12}$$

with a communication cost of $O(dT^{\alpha})$.

Theorem 4.13 (SDP-DPE)**.** *With* $\sigma_n = 2\sigma_{ns}\sqrt{Sd} = O\left(\frac{B\sqrt{Sds_l}\log(s_l/\delta)}{\varepsilon|U_l|}\right)$ *in the* l*-th phase and* $\beta = 1/(kT)$*, SDP-DPE achieves the following expected regret:*

$$\begin{aligned}\mathbb{E}[R(T)] =& O(\sqrt{dT\log(kT)}) + O\left(\sigma T^{1-\alpha/2}\sqrt{\log(kT)}\right) \\ &+ O\left(\frac{Bd^{3/2}\ln(d/\delta)T^{1-\alpha}\sqrt{\log(kT)}}{\varepsilon}\right) + O\left(\frac{Bd^{5/2}\ln(d/\delta)\sqrt{\log(kT)}}{\varepsilon}\right),\end{aligned} \tag{4.13}$$

and the communication cost is $O(dT^{3\alpha/2})$ *bits.*

For all Theorems 4.11, D.10, and D.11, the first term is from action-related uncertainty, the second term is due to client-related uncertainty, and the third and forth terms are introduced by privacy guarantee. We provide the detailed proofs of Theorems 4.11, D.10, and D.11 in Appendix C.2 and make the following remarks.

Remark 4.14 (Privacy "for-free")**.** Comparing the above results with Theorem 4.9 for the non-private case, we observe that the DP-DPE algorithm enables us to achieve privacy guarantees "for free" in the central and shuffle DP models, in the sense that the additional regret due to privacy protection is only a lower-order additive term. Essentially, this is because the uncertainty introduced by privacy noise is dominated by the client-related uncertainty, which can be captured by our carefully designed confidence width W_l in Eq. (4.3) and our choice of σ_n for different PRIVATIZERS. See more discussions on achieving privacy "for-free" in Section 4.7.

Remark 4.15 (Regret-privacy tradeoff under the shuffle model)**.** Consider the regret due to privacy protection. From Theorem 4.11 and D.10, we can see that while the local DP model ensures a stronger privacy guarantee compared to the central DP model, it introduces an additional regret of $O(T^{1-\alpha/2})$ compared to $O(T^{1-\alpha})$ in the central DP model. The shuffle DP model, however, leads to a much better tradeoff between regret and privacy, achieving nearly the same regret guarantee as the central DP model, yet assuming a similar trust model to the local DP model (i.e., without a trustworthy central server).

Remark 4.16 (Communication cost)**.** Both CDP-DPE and LDP-DPE consume the same amount of communication resources as the non-private DP-DPE algorithm, measured by the number of real numbers [137]. In contrast, SDP-DPE relies only on binary feedback from the clients, and thus, the communication cost is measured by the number of bits. It is worth noting that sending messages consisting of real numbers could be difficult in practice on finite computers [26, 74], and hence in this case, it is desirable to use SDP-DPE, which incurs a communication cost of $O(dT^{3\alpha/2})$ bits.

4.7 Discussion on Achieving Privacy "for Free"

4.7.1 Connection Between Bandit Online Learning and Supervised Learning

Following the remark on privacy "for-free" (Remark 4.14), in this section, we draw an interesting connection of our novel bandit online learning problem to private (distributed) supervised learning problems, through which we provide more intuition on why DP-DPE can achieve privacy "for-free". In particular, we compare our problem with differentially private stochastic convex optimization (DP-SCO) [13], where the goal is to approximately minimize

the population loss[8] over convex and Lipschitz loss functions given n *i.i.d.* d-dimensional samples from a population distribution, while protecting privacy under different trust models. More specifically, via noisy stochastic gradient descent (SGD), the excess losses[9] in DP-SCO under various trust models are roughly as follows:

$$\text{Central and Shuffle Model [13, 35]:} \quad \widetilde{O}\left(\frac{1}{\sqrt{n}} + \frac{\sqrt{d}}{n\varepsilon}\right), \tag{4.14}$$

$$\text{Local Model [49]:} \quad \widetilde{O}\left(\frac{1}{\sqrt{n}} + \frac{\sqrt{d}}{\sqrt{n}\varepsilon}\right). \tag{4.15}$$

Recall our main results in Table 4.1 as follows (ignoring all the logarithmic terms for clarity):

$$\text{Central and Shuffle Model:} \quad \widetilde{O}\left(T^{1-\alpha/2} + \frac{d^{3/2}T^{1-\alpha}}{\varepsilon}\right), \tag{4.16}$$

$$\text{Local Model:} \quad \widetilde{O}\left(T^{1-\alpha/2} + \frac{d^{3/2}T^{1-\alpha/2}}{\varepsilon}\right). \tag{4.17}$$

Now, one can easily see that in both problems, privacy protection is achieved "for free" in the central and shuffle models, in the sense that the second term (i.e., the additional privacy-dependent term) is a lower-order term (with respect to n or T) compared to the first term in both Eqs. (4.14) and (4.16). On the other hand, under the much stronger local model, in both problems, the additional privacy-dependent term is of the same order as the first term in both Eqs. (4.15) and (4.17).

We tend to believe that the above interesting connection is not a coincidence. Rather, it provides us with a sharp insight into our introduced DP-DLB formulation. In particular,

[8]The population loss for a solution w is given by $\mathcal{L}(w) \triangleq \mathbb{E}_{z\in\mathcal{D}}[l(w,z)]$, where w is the chosen solution (e.g., weights of a classifier), z is a testing sample from the population distribution $\mathcal{D}$, and l is a convex loss function of w.

[9]The excess loss measures the gap between the chosen solution and the optimal solution in terms of the population loss. That is, the excess loss of w is given by $\mathcal{L}(w) - \min_{w'\in\mathcal{W}} \mathcal{L}(w')$, where w is often the minimizer of the *Empirical Risk Minimization (ERM)* problem: $\hat{\mathcal{L}}(w) \triangleq \frac{1}{n}\sum_{i=1}^{n} l(w, z_i)$, where $\{z_i\}_{i=1}^{n}$ are *i.i.d.* samples from the population distribution. To find the minimizer of ERM, we often resort to SGD.

we know that the first term $1/\sqrt{n}$ in DP-SCO comes from standard concentration results, i.e., how independent samples approximate the true population parameter. Similarly, in our problem, the first term $\sqrt{d}T^{1-\alpha/2}$ comes from the concentration due to client sampling, which is used to approximate the true unknown population parameter θ^*. On the other hand, the second term in DP-SCO is privacy-dependent and comes from the average of noisy gradients. Similarly, in our problem, the second term is due to the average of the local reward vectors with added noise for preserving privacy.

In addition to these useful insights, we believe that this interesting connection also opens the door to a series of important future research directions, in which one can leverage recent advances in DP-SCO to improve our main results (dependence on d, communication efficiency, etc.).

4.7.2 Differentially Private Linear Bandits

Motivated by the cellular configuration problem, we consider the distributed linear bandits with partial feedback in the main content and propose the DP-DPE algorithmic framework to address the newly introduced challenges. However, we highlight that our developed techniques with minor modifications can also achieve similar results in terms of regret and privacy for the standard linear bandits, where there is no client-related uncertainty ($\sigma = 0$) and any user u can provide direct (noisy) reward observations, i.e., $\theta_u = \theta^*$ in our notations. That is, we can design differentially private linear bandits where one can also achieve privacy "for free" in the central and shuffle DP models (similar to Remarks 4.14). We list the corresponding results in Table 4.2. This might be of independent interest to the bandit learning community. We provide the detailed description of differentially private linear bandits in Appendix C.3.

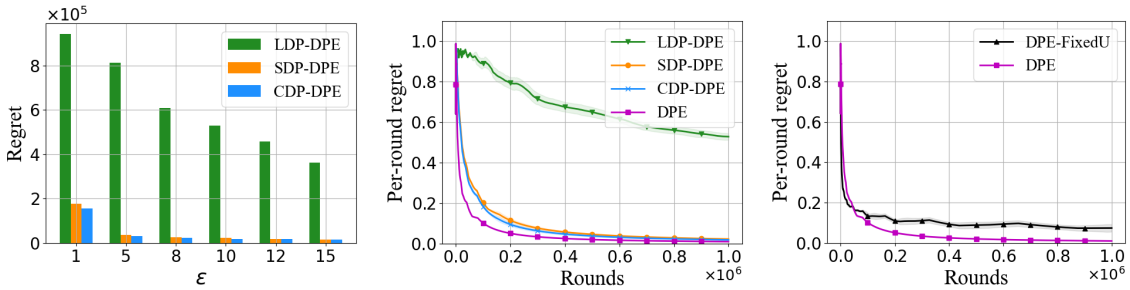
Table 4.2: Summary of the DP algorithms for the standard linear bandits

Algorithm[10]	Regret	Privacy
PE [84]	$O\left(\sqrt{dT\log(kT)}\right)$	None
CDP-PE	$O\left(\sqrt{dT\log(kT)}+\frac{Bd^{3/2}\log(T)\sqrt{\ln(1/\delta)\log(kT)}}{\varepsilon}\right)$	(ε,δ)-DP
LDP-PE	$O(\sqrt{dT\log(kT)}+O\left(\frac{Bd^{3/2}\sqrt{\ln(1/\delta)T\log(kT)}}{\varepsilon}\right)$	(ε,δ)-LDP
SDP-PE	$O\left(\sqrt{dT\log(kT)}+\frac{Bd^{3/2}\log(T)\sqrt{\ln(1/\delta)\log(kT)}}{\varepsilon}\right)$	(ε,δ)-SDP

[10]PE is the (non-private) phased elimination algorithm in [84]; CDP-PE, LDP-PE, and SDP-PE represent the designed DP algorithms in the central, local, and shuffle models, respectively, which guarantee (ε,δ)-DP, (ε,δ)-LDP, and (ε,δ)-SDP, respectively.

Remark 4.17. We can achieve the above "for-free" results because the sensitive information in linear bandits are only rewards, which is in sharp contrast to linear *contextual* bandits where both contexts and rewards need to be protected. In this case, the best known private regrets in the central, local and shuffle model are $\tilde{O}(\frac{\sqrt{T}}{\sqrt{\varepsilon}})$ [120], $\tilde{O}(\frac{T^{3/4}}{\sqrt{\varepsilon}})$ [142], and $\tilde{O}(\frac{T^{3/5}}{\varepsilon^{2/5}})$ [38], respectively.

4.8 Numerical Results



(a) (b) (c)

Figure 4.2: Performance comparisons of different algorithms. The shaded area indicates the standard deviation. (a) Final cumulative regret vs. the privacy budget ε. (b) Per-round regret vs. time with privacy parameters $\varepsilon = 10$ and $\delta = 0.25$. (c) Per-round regret vs. time for two non-private algorithms.

In this section, we conduct simulations to evaluate the performance of DP-DPE. The detailed setting of our simulations is as follows: $d = 20, k = 10^3, \sigma = 0.1, |\mathcal{U}| = 10^5, \alpha = 0.8$, and $T = 10^6$. We perform 20 independent runs for each set of simulations.

First, we study the regret performance of DP-DPE under different DP models. Recall that we use CDP-DPE, LDP-DPE, and SDP-DPE to denote DP-DPE in the central, local, and shuffle DP models, respectively. In Fig. 4.2a, we present the cumulative regret at the end of T rounds for the three algorithms under different values of privacy budget ε. We can observe an obvious tradeoff between the privacy budget and the regret performance for all the DP models: the cumulative regret decreases as the privacy requirement becomes less stringent (i.e., a larger ε). In addition, it also reflects the regret-privacy tradeoff across different DP models. That is, with the same privacy budget ε, while LDP-DPE has the largest regret yet without requiring the clients to trust anyone else (neither the server nor a third party), CDP-DPE achieves the smallest regret but relies on the assumption that the clients trust the server. Interestingly, SDP-DPE achieves a regret fairly close to that of CDP-DPE, yet without the need to trust the server. This is well aligned with our theoretical results that SDP-DPE achieves a better regret-privacy tradeoff.

In addition, we are also interested in the regret loss due to privacy protection and how efficiently DP-DPE performs the global bandit learning. Fix the privacy parameters $\varepsilon = 10$ and $\delta = 0.25$. In Fig. 4.2b, we plot how the per-round regret of the three algorithms (i.e., CDP-DPE, LDP-DPE, and SDP-DPE) varies over time compared to the non-private DP-DPE algorithm (i.e., DPE). We observe that LDP-DPE incurs the largest regret while ensuring the strongest privacy guarantee (i.e., (ε, δ)-LDP). On the other hand, the regret performance of CDP-DPE and SDP-DPE is very close to that of DPE (that does not ensure any privacy guarantee), under the assumption of a trusted central server and a trusted third party shuffler, respectively. This observation, along with our theoretical results, shows that

DP-DPE can indeed achieve privacy "for-free" under the central and shuffle models.

Finally, we show that the exponentially-increasing client-sampling plays a key role in balancing the regret and the communication cost. To this end, we compare DPE (i.e., non-private DP-DPE) with another non-private algorithm, called DPE-FixedU in Fig. 4.2c. DPE-FixedU is similar to DPE but samples only a fixed number U of participating clients in each phase (i.e., the participating clients are different, but the number of clients in each phase is fixed, in contrast to our increasing sampling schedule). For a fair comparison, we choose the value of U such that the communication cost is the same under DPE and DPE-FixedU. The results show that DPE learns much faster than DPE-FixedU while incurring the same communication cost.

4.9 Chapter Summary

In this chapter, we studied a new problem of global reward maximization with partial distributed feedback. This problem is motivated by several practical applications where the expected reward of an action represents the overall performance over a large population. In such scenarios, it is often difficult, if not impossible, to collect exact reward feedback. To that end, we proposed a differentially private distributed linear bandits formulation, where the learning agent samples clients and interacts with them by iteratively aggregating such partial distributed feedback in a privacy-preserving fashion. We then developed a unified algorithmic learning framework, called DP-DPE, which can be naturally integrated with different DP models, and systematically established the regret-communication-privacy tradeoff.

Chapter 5

Distributed Kernelized Bandits with Biased Feedback

5.1 Introduction

The preceding chapter studies the problem of global reward maximization under uncertainty with distributed biased feedback by modeling the correlation between different actions/decisions via a linear parameterized reward function. For some real-world applications, the unknown objective reward function could be non-parameterized, *non-linear*, and even *non-convex*. An important way to capture general (e.g., *non-linear* and even *non-convex*) unknown objective functions is to consider a smoothness condition specified by a small norm of a Reproducing Kernel Hilbert Space (RKHS) associated with a kernel function. This setup is often referred to as *kernelized bandits*. This chapter, motivated by cellular network configuration problem, studies the target problem in the distributed kernelized bandits setting. Recall the cellular network configuration problem in Figure 4.1. In the kernelized bandits setting, the (unknown) mean global reward (network-level performance) $f(\mathbf{x})$ of configuration $\mathbf{x}$ is assumed to live in a RKHS with a known kernel.

Thanks to the strong link between RKHS functions and Gaussian processes (GP) [37, 75, 125], an extensive line of work has exploited GP models to estimate an unknown function f given a set of (noisy) evaluations of its values $f(\mathbf{x})$ at chosen actions $\mathbf{x}$. However, in the

application of cellular network configuration, the value $f(\mathbf{x})$ represents an overall effect of action $\mathbf{x}$ (configuration) on a large population of users where it is difficult for the learning agent to make direct observations; yet, the agent could collect some partial feedback from the distributed users in the population. In addition, feedback from these users could be biased due to user heterogeneity (e.g., difference preferences). Therefore, we assume that each user u in the population is associated with a local function f_u, which is a function sampled from a GP with mean f.

To that end, we study a new kernelized bandits setting where the agent could not get direct evaluations of the unknown reward function but only distributed biased feedback. We refer to this setting as *kernelized bandits with distributed biased feedback*. This bandit problem is shared by several other practical applications, including dynamic pricing [102] and public policy making [17]. However, existing learning algorithms developed for standard kernelized/GP bandits (e.g., `GP-UCB` [37, 125]) rarely consider such partial biased feedback in a distributed setting. To solve this new problem, a learning algorithm needs to be able to learn the unknown function from such biased feedback in a *sample-efficient* manner. Moreover, two practical challenges naturally arise in our problem: *communication cost* due to distributed learning [32] and *computation complexity* due to GP update [23]. Therefore, not only need the learning algorithms be sample-efficient, but they must also be scalable in terms of both communication efficiency and computation complexity.

To that end, we propose the *learning with communication* framework where the biased feedback is communicated in phases, and design a new *distributed phase-then-batch-based elimination* algorithm that aggregates the distributed biased feedback in a communication-efficient manner and eliminates suboptimal actions in a computation-efficient manner while achieving a sublinear regret. Our main contributions are summarized as follows.

- To the best of our knowledge, this is the first work that studies a new kernelized bandits setting with distributed biased feedback, where three key challenges (user heterogeneity, communication efficiency, and computation complexity) inherently arise in the design of sample-efficient, scalable learning algorithms. While it is natural to consider phased elimination type of algorithms in such settings, the standard phased elimination algorithm relies on the so-called (near-)optimal experimental design [85], which cannot be directly applied to kernelized bandits due to the possible infinite feature dimension of RKHS functions.

- To that end, we design a new phased elimination algorithm, called *distributed phase-then-batch-based elimination (`DPBE`)*, which is carefully crafted to address all the aforementioned challenges. In particular, `DPBE` adds a *user-sampling* process to reduce the impact of bias from each individual user and selects actions according to *maximum variance reduction* within each phase. Moreover, a *batching* strategy is employed to improve both communication efficiency and computation complexity. That is, instead of selecting a new action at each round, `DPBE` plays the same action for a batch of rounds before switching to the next one. Not only does it help reduce the number of times one needs to compute the next action via GP update, but it also allows for reducing the dimensions of the vectors and matrices involved in both communication and computation.

- We show that `DPBE` achieves a sublinear regret of $\tilde{O}(T^{1-\alpha/2} + \sqrt{\gamma_T T})$[1] while incurring a communication cost of $O(\gamma_T T^{\alpha})$ and a computation complexity of $O((|\mathcal{D}|\gamma_T^3 + \gamma_T^4)\log T + \gamma_T T^{\alpha})$, where γ_T is the *maximum information gain* associated with the kernel of the unknown function f, $\mathcal{D}$ is the decision set, and $\alpha > 0$ is a user-sampling parameter that we can tune. It is worth noting that `DPBE` with $\alpha \in (0, 1)$ has a better

[1]The notation $\tilde{O}(\cdot)$ ignores polylog terms. Bounds on γ_T of different kernel functions can be found in Appendix D.1.2.

computation complexity than some variants of the state-of-the-art algorithms (originally developed for standard kernelized bandits without biased feedback); see Table 5.1. Specifically, `DPBE` achieves three significant improvements compared to the state-of-the-art algorithms: (i) user-sampling efficiency ($O(T^\alpha)$ vs. T), (ii) communication cost ($O(\gamma_T T^\alpha)$ vs. T), and (iii) computation complexity ($O(\gamma_T T^\alpha)$ vs. $O(T^3)$). Furthermore, we conduct extensive simulations to validate our theoretical results and evaluate the empirical performance in terms of regret, communication cost, and running time.

- Finally, we generalize our phase-then-batch framework to incorporate various *differential privacy* (DP) models (including the central, local, and shuffle models) into `DPBE`, which ensures privacy guarantees for users participating in the distributed learning process.

5.2 Related Work

Kernelized bandits. Since [125] studied GP in the bandit setting, kernelized bandits (also called GP bandits) have been widely adopted to address black-box function optimization over a large or infinite domain [37]. Considering different application scenarios, kernelized bandits under different settings have recently been studied, including heavy-tailed payoffs [114], model misspecification [15], and corrupted rewards [16]. As typically considered in the literature, these works also assume that direct (noisy) feedback of the unknown function at a chosen action is available to the agent. In sharp contrast, we study a new, practical setting where only distributed biased feedback can be obtained. Under this setting, not only does one need to use biased feedback in a sample-efficient manner, but one also has to consider communication efficiency, which is a common issue in distributed bandit-learning settings.

Distributed/collaborative kernelized bandits. While distributed or collaborative ker-

nelized bandits have been studied recently [40, 44, 123], we highlight the key difference between our model and theirs as follows: motivated by real-world applications, we aim to learn one (global) bandit while most of them also aim to learn every local model, which results in quite different regret definitions (their group regret vs. our standard regret defined in Section 5.3). Moreover, they assume that every party (corresponding to a user in our problem) shares the same objective function. While [40] also studies similar bandit optimization with biased feedback, they assume a fixed number of local agents and bound the regret in terms of the distance between the target function and local functions, which could be very large. In addition, [40] does not consider communication efficiency, which is a key challenge in distributed learning.

Experimental design for kernelized bandits. In [145], the authors propose to adaptively embed the feature representation of each action into a lower-dimensional space in order to apply the (near-)optimal experimental design for finite-dimensional actions. However, the intermediate regret due to the approximation error over T rounds is not considered at all because their goal is to find an ε-optimal arm at the end of T (i.e., a pure exploration problem) rather than minimizing the cumulative regret. While [25] aims at minimizing the cumulative regret, their algorithm and analysis are more complex than ours: it requires a non-standard robust estimator, obtaining an optimal distribution on the simplex, drawing samples from this distribution, and solving a second optimization problem. In contrast, we simply use the standard GP posterior mean and variance estimators, which can be computed in closed-form. Moreover, our algorithm can also be easily extended to handle infinite action sets (see Remark 5.2) rather than a finite set considered in [25].

Comparison with Chapter 4. Both Chapter 4 and this chapter study a the problem of global reward maximization problem without direct feedback and employ a phase-based elimination algorithm. The main difference is that Chapter 4 consider linear bandits by

assuming a linear reward function while this chapter studies kernelized bandits that can capture general *non-linear* and even *non-convex* functions and recover linear bandits as a special case when choosing a linear kernel. This strict generalization introduces three unique challenges: (i) different from the linearly parameterized bandits where the bias in the feedback can be quantified with a same-dimension random vector (i.e., $\xi_u = \theta_u - \theta^* \in R^d$ at each user u), it is unclear how to make an assumption of the bias in the non-parametric kernelized bandits setting in order to learn the unknown global reward function; (ii) due to the possible infinite feature dimension of functions in an RKHS, the (near-)optimal experimental design approach used in the phased-elimination algorithm for linear bandits cannot be directly adapted to kernelized bandits. Despite some recent efforts towards extending this experimental design based approach to kernelized bandits [25, 145], there still remain some key limitations (see our above discussion); (iii) since computation complexity is a critical bottleneck in kernelized bandits, a proper computation-efficient learning algorithm is desired when addressing our problem.

5.3 Preliminaries

5.3.1 Problem Setting

We introduce a new kernelized bandits problem where the unknown function represents the overall reward over a large population containing an infinite number of users. The unknown reward function $f : \mathcal{D} \to \mathbb{R}$ is assumed to be fixed over a finite set of decisions $\mathcal{D} \subseteq \mathbb{R}^d$. At round t, the agent chooses an action $\mathbf{x}_t \in \mathcal{D}$, leading to a reward with mean $f(\mathbf{x}_t)$. This reward is unknown to the agent but captures the overall effectiveness of action $\mathbf{x}_t$ over the entire population $\mathcal{U}$, thus called *global reward*. Meanwhile, each user u in the population

observes a (noisy) *local reward*: $y_{u,t} = f_u(\mathbf{x}_t) + \eta_{u,t}$ with mean $f_u(\mathbf{x}_t)$, where $\eta_{u,t}$ is the noise, and $f_u : \mathcal{D} \to \mathbb{R}$ is the local reward function, assumed to be an (unknown) realization of a random function (specified soon) with mean f. In this setting, the exact global reward corresponding to the entire population cannot be observed; only biased local reward feedback is available to the agent. We make the following assumptions about the unknown function f, the local function f_u, and the noise in the reward observations.

Assumption 1. We assume that function f is in the Reproducing Kernel Hilbert Spaces (RKHS), denoted by $\mathcal{H}_k$. Note that RKHS $\mathcal{H}_k$ is completely specified by its kernel function $k(\cdot,\cdot)$ (and vice-versa), with an inner product $\langle \cdot,\cdot \rangle_k$ obeying the reproducing property: $f(\mathbf{x}) = \langle f(\cdot), k(\mathbf{x},\cdot)\rangle_k$ for all $f \in \mathcal{H}_k$ [37]. We list the most commonly used kernel functions (such as Squared Exponential (SE) and Matérn kernels) in Appendix D.1. Moreover, we assume that function f has a bounded norm: $\|f\|_k \triangleq \sqrt{\langle f, f\rangle_k} \leqslant B$, and that the kernel function is also bounded: $k(\mathbf{x},\mathbf{x}) \leqslant \kappa^2$ for every $\mathbf{x} \in \mathcal{D}$, where both B and κ are positive constants.

Assumption 2. When the agent samples a user u to collect feedback, the local reward function f_u at u is assumed to be a function sampled from the GP with mean f and covariance[2] $k(\cdot,\cdot)$, i.e., $f_u \sim \mathcal{GP}(f(\cdot), k(\cdot,\cdot))$. In addition, we assume that each user is sampled independently for collecting feedback.

Assumption 3. We assume that the observation noise $\eta_{u,t} \sim \mathcal{N}(0,\sigma^2)$ is Gaussian with variance $\sigma > 0$ and that it is identically and independent distributed (*i.i.d.*) over time and across users.

[2]Our theoretical framework is applicable to a more general setting where the covariance of the local reward function is $v^2 k(\cdot,\cdot)$, i.e., $f_u \sim \mathcal{GP}(f(\cdot), v^2 k(\cdot,\cdot))$. This scaling parameter v^2 captures the variance of the bias in the local reward function f_u with its mean being the global reward function f. For this more general setting, our theoretical results still hold with only a slight adjustment to the posterior variance in the confidence width function (5.12).

The goal of the agent is to maximize the cumulative global reward, or equivalently, to minimize the regret defined as follows:

$$R(T) \triangleq \sum_{t=1}^{T} \left(\max_{\mathbf{x} \in \mathcal{D}} f(\mathbf{x}) - f(\mathbf{x}_t) \right). \tag{5.1}$$

5.3.2 Learning with Communication

For black-box function optimization based on noisy bandit feedback, kernelized bandit algorithms have shown strong empirical and theoretical performance. However, the agent in our problem setting does not have access to unbiased feedback of the object function f but has to collect biased feedback from distributed users from a large population. This scenario leads to the following framework of *learning with communication*.

Communication happens when some users are selected to report their feedback to the agent based on their biased local reward observations. By aggregating such biased feedback from the users, the agent improves her confidence in estimating function f and adjusts her decisions in the following rounds accordingly. To account for scalability, the agent collects distributed feedback from users periodically instead of immediately after making each decision. We call the time duration between two communications as a *phase*. Consider a particular phase l. Let $\mathcal{T}_l$ be the set of round indices in the l-th phase and U_l be the set of selected users, called *participants*, that will report their feedback. With the actions $\{\mathbf{x}_t : t \in \mathcal{T}_l\}$ chosen by the agent in this phase, each user u in U_l sends the feedback $g(\{y_{u,t}\}_{t \in \mathcal{T}_l})$ to the agent at the end of the phase, where $g(\cdot)$ is a function (e.g., the average) of the local reward observations and is assumed to be the same for all users. Then, by aggregating all feedback $\{g(\{y_{u,t}\}_{t \in \mathcal{T}_l})\}_{u \in U_l}$, the agent estimates f and decides $\mathbf{x}_t$ for round t in the next phase $\mathcal{T}_{l+1}$. This learning with communication process is repeated until the end of T, with the goal of

maximizing the cumulative (global) reward.

In this framework, we assume that the agent can employ some existing incentive mechanisms [94] in order to collect enough feedback for learning, but the cost has to be considered, e.g., the communication resources consumed for collecting feedback data. In addition, communication cost is also a critical factor in a general distributed learning system. In this work, we use the total quantity of communicated numbers (between the agent and all users) as another metric, in addition to the regret metric, to evaluate the communication efficiency of learning algorithms for our problem. Let L be the total number of phases in T rounds and $N_{u,l} \triangleq \dim\left(g(\{y_{u,t}\}_{t\in\mathcal{T}_l})\right)$ be the dimension of user u's feedback (which is the number of scalars in user u's feedback). Then, the total communication cost, denoted by $C(T)$, is as follows:

$$C(T) \triangleq \sum_{l=1}^{L} \sum_{u\in U_l} N_{u,l}. \tag{5.2}$$

In the following, we explain the learning with GP framework for standard kernelized bandits.

5.3.3 Learning with Gaussian Process

A Gaussian process (GP) over input domain $\mathcal{D}$, denoted by $\mathcal{GP}(\mu(\cdot), k(\cdot,\cdot))$, is a collection of random variables $\{f(\mathbf{x})\}_{\mathbf{x}\in\mathcal{D}}$ where every finite number of them $\{f(\mathbf{x}_i)\}_{i=1}^{n}, n \in \mathbb{N}$, is jointly Gaussian with mean $\mathbb{E}[f(\mathbf{x}_i)] = \mu(\mathbf{x}_i)$ and covariance $\mathbb{E}[(f(\mathbf{x}_i) - \mu(\mathbf{x}_i))(f(\mathbf{x}_j) - \mu(\mathbf{x}_j))] = k(\mathbf{x}_i, \mathbf{x}_j)$ for every $1 \leqslant i, j \leqslant n$. Hence, $\mathcal{GP}(\mu(\cdot), k(\cdot,\cdot))$ is specified by its mean function μ and a (bounded) covariance (or kernel) function $k : \mathcal{D} \times \mathcal{D} \to [0, \kappa^2]$. Assume that choosing action $\mathbf{x}_t$ at round t reveals a noisy observation:

$$y_t = f(\mathbf{x}_t) + \eta_t, \tag{5.3}$$

where $\eta_t \sim \mathcal{N}(0, \lambda)$ is a zero-mean Gaussian noise with variance $\lambda > 0$. Standard GP algorithms implicitly use $\mathcal{GP}(0, k(\cdot,\cdot))$ as the prior distribution over f. Then, given the observations $\mathbf{y}_t = [y_1, \dots, y_t]^\top$ corresponding to a sequence of actions $\mathbf{X}_t = [\mathbf{x}_1^\top, \dots, \mathbf{x}_t^\top]^\top$, the posterior distribution is also Gaussian with the mean and variance in the following closed-form:

$$\mu_t(\mathbf{x}) \triangleq \mathbf{k}(\mathbf{x}, \mathbf{X}_t)^\top (\mathbf{K}_{\mathbf{X}_t\mathbf{X}_t} + \lambda \mathbf{I})^{-1} \mathbf{y}_t, \tag{5.4}$$

$$\sigma_t^2(\mathbf{x}) \triangleq \mathbf{k}(\mathbf{x}, \mathbf{x}) - \mathbf{k}(\mathbf{x}, \mathbf{X}_t)^\top (\mathbf{K}_{\mathbf{X}_t\mathbf{X}_t} + \lambda \mathbf{I}^{-1})^{-1} \mathbf{k}(\mathbf{x}, \mathbf{X}_t), \tag{5.5}$$

where $\mathbf{k}(\mathbf{x}, \mathbf{X}_t) = [k(\mathbf{x}, \mathbf{x}_s)]_{s=1,\cdots,t}^\top \in \mathbb{R}^{t\times 1}$ and $\mathbf{K}_{\mathbf{X}_t\mathbf{X}_t} = [k(\mathbf{x}, \mathbf{x}')]_{\mathbf{x},\mathbf{x}'\in\mathbf{X}_t} \in \mathbb{R}^{t\times t}$ is the corresponding kernel matrix.

Next, we introduce an important kernel-dependent quantity, called *maximum information gain* [125]:

$$\gamma_t(k, \mathcal{D}) \triangleq \max_{\mathbf{X}\subseteq\mathcal{D}:|\mathbf{X}|=t} \frac{1}{2} \log\det\left(\mathbf{I} + \lambda^{-1}\mathbf{K}_{\mathbf{X}\mathbf{X}}\right), \tag{5.6}$$

which is often used to derive regret bounds. In addition, we have that $\gamma_t(k, \mathcal{D})$ scales sublinearly with t for most commonly used kernels (see Appendix D.1). For ease of notation, we often simply use γ_t to denote $\gamma_t(k, \mathcal{D})$ when the kernel function k and the dataset $\mathcal{D}$ are clear from the context.

Thanks to the strong connection between RKHS functions and GP [75] with the same kernel function k, one can use the above GP model to approximate unknown function $f \in \mathcal{H}_k$ within a reliable confidence interval with high probability.

5.4 Algorithm Design

5.4.1 New Challenges and Main Ideas

In Section 5.3, we describe the learning with communication framework, which requires the distributed biased feedback to be communicated in phases and exhibits experimental scalability. This framework naturally leads us to consider a phased elimination algorithm that gradually eliminates suboptimal actions by periodically aggregating and analyzing the local feedback from the participants. However, several new challenges arise in our setting compared to the standard phase elimination algorithm in linear bandits [84, 85].

(i) How to select actions for each phase? The standard phase elimination algorithm often relies on the so-called near-optimal experimental design (i.e., a probability distribution over the currently active set) that minimizes the worst-case variance [84]. However, due to the possible infinite feature dimension of RKHS functions, adapting this approach to kernelized bandits setting is nontrivial even with the strong assumptions, requirements, and complicated algorithm design (e.g., [145] and [25], see discussion in Section 5.2). We are wondering if there is a simple and efficient method of selecting actions in each phase for our kernelized bandits setting. (Challenge ⓐ).

(ii) How to use biased feedback? In contrast to the standard phase elimination algorithm where feedback is unbiased, in our setting the local feedback from a particular user is biased. In order to reduce the impact of bias, an efficient user-sampling scheme is needed. However, how to incorporate this idea into the phase elimination algorithm is unclear (Challenge ⓑ).

(iii) How to deal with scalability? In our setting, scalability refers to both computation complexity and communication cost. On the one hand, it is well-known that standard GP bandits suffer a poor computation complexity (e.g., $O(T^3)$) due to the matrix inverse at each

step for GP posterior update. On the other hand, due to the communication between the agent and the users, it is imperative to ensure a low communication cost (Challenge ⓒ).

Our approach. We propose a novel phase elimination algorithm that is able to simultaneously address all the above challenges. We highlight the main ideas as follows. (i) *User-sampling* for distributed biased feedback. In each phase, a well-tuned subset of users is sampled to reduce the impact of bias from each individual user. (ii) *Maximum variance reduction* for action selection. Upon selecting the next action within each phase, it simply selects the one that has the largest posterior variance. (iii) *Batching strategy* for scalability. Instead of selecting a new action at each round within a phase, it consistently plays the same action for a batch of rounds before selecting the next one, i.e., *rare-switching*. By reducing the number of times selecting a new action (which could be much smaller than the phase length), it also reduces the number of unique actions chosen within each phase, which can be utilized to improve the scalability in terms of both computation and communication through a proper design. Specifically, (a) *Computation*: via a *posterior reformulation* (specified in Section 5.4.2), we convert the dimension of the matrix in the inverse operation from the total rounds to the number of batches in each phase; (b) *Communication*: we let each participant *merge the local reward observations* in each batch before sending her feedback at the end of each phase. That is, the feedback $g(\{y_{u,t}\}_{t\in\mathcal{T}_l})$ from each participating user u in phase l is a vector, where each element corresponds to the average local reward of a batch. Then, the dimension of the feedback $g(\{y_{u,t}\}_{t\in\mathcal{T}_l}$ becomes the number of batches. For example, consider a particular phase with a total of 10 rounds. Without batching strategy, one requires to select an action for each round, i.e., 10 actions for this phase. However, the batching strategy selects an action for each batch. If each batch has size two, there are 5 batches in this phase, and the dimension of the matrix in the inverse operation is shrunk from 10 to 5, which will reduce the computation complexity about $10^3/5^3 = 8$ times for matrix inverse

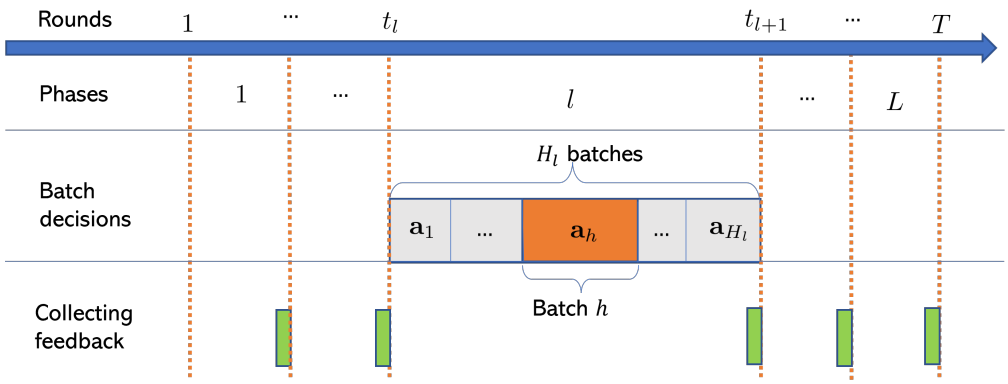


Figure 5.1: The phase-then-batch strategy: T rounds are divided into L phases; at the end of each phase, participants report their feedback, which is used for deciding actions in the next phase; within each phase l, decisions are made in a batched fashion, e.g., playing $\mathbf{a}_h$ at all the rounds in the h-th batch.

operations! In addition, by merging local observations of each unique action, only 5, instead of 10, (averaged) local rewards are communicated at each user.

5.4.2 Distributed Phase-then-Batch-based Elimination (DPBE)

Following the main ideas stated in the above section, we propose the phase-then-batch schedule strategy, shown in Figure 5.1 and design the distributed phase-then-batch-based elimination (DPBE) algorithm in Algorithm 6.

The DPBE algorithm is a phased elimination algorithm, which maintains a set $\mathcal{D}_l$ of active actions that are possible to be optimal and updates the active set after aggregating the distributed feedback.

Consider a particular phase l, DPBE has three main steps: 1) action selection (Lines 5-9); 2) distributed feedback collection (Lines 11-15); and 3) action elimination (Lines 16-20).

Before describing the details of DPBE, we explain some additional notations used in the algorithm. Throughout this chapter, we use another notation "$\mathbf{a}$" to denote the specific

chosen action under our algorithm to avoid too many subscripts or superscripts for all the batch, phase, or round indices. Consider the l-th phase. Let t_l and T_l be the time index right before the l-th phase and the length of the l-th phase, respectively. Then, the round indices in the l-th phase can be represented as $\mathcal{T}_l = \{t \in [T] : t_l + 1 \leqslant t \leqslant t_l + T_l\}$. In addition, $\mathcal{T}_l(\mathbf{a}) \triangleq \{t \in \mathcal{T}_l : \mathbf{x}_t = \mathbf{a}\}$ denotes the time indices when action $\mathbf{a}$ is selected in this phase, and H_l represents the number of batches in the l-th phase.

1) Action selection (Lines 5-9): In the l-th phase, actions are selected from the active set $\mathcal{D}_l$. As mentioned before, each selection is based on *maximum variance reduction* [133], and we employ batch schedule for scalability. Specifically, in the h-th batch, we find the action $\mathbf{a}_h$ that maximizes a reformulated posterior variance $\Sigma_{h-1}(\cdot)$ defined in Eq. (5.7) after $h-1$ batches (Eq. (5.8)). This is possible because the posterior variance can be computed without knowing any reward observations (see Eq. (5.5)). Then, play this action for $T_l(\mathbf{a}_h) \triangleq \lfloor (C^2 - 1)/\Sigma^2_{h-1}(\mathbf{a}_h) \rfloor$ rounds, which forms the h-th batch. Here, the batch size schedule is inspired by the *rare-switching* idea in [4, 23]. This batch schedule strategy enables us to merge rounds and thus shrink the dimensions of the matrix and vectors used for computing the variance in Eq. (5.5). By the end of each batch, we update the variance function by incorporating the action in the current batch. Let $\mathbf{A}_h = [\mathbf{a}_1^\top, \ldots, \mathbf{a}_h^\top]^\top \in \mathbb{R}^{h \times d}$ be the $h \times d$ matrix that contains the h chosen actions so far. We reformulate the standard posterior variance in Eq. (5.5) and update the posterior variance as follows:

$$\Sigma_h^2(\mathbf{x}) \triangleq k(\mathbf{x}, \mathbf{x}) - \mathbf{k}(\mathbf{x}, \mathbf{A}_h)^\top (\mathbf{K}_{\mathbf{A}_h \mathbf{A}_h} + \lambda \mathbf{W}_h^{-1})^{-1} \mathbf{k}(\mathbf{x}, \mathbf{A}_h), \tag{5.7}$$

where $\mathbf{W}_h \in \mathbb{R}^{h \times h}$ is a diagonal matrix with $[W_h]_{ii} = T_l(\mathbf{a}_i)$ for any $i \in [h]$, and λ is set to be the noise variance of local observations, i.e., $\lambda = \sigma^2$. Here, we reformulate the standard posterior variance in Eq. (5.5) with Eq. (5.7) in order to save computation complexity (especially for computing matrix inverse) while maintaining the same order of regret (sacrificing

only a constant multiplier).

2) Distributed feedback collection (Lines 11-15): To reduce the impact of bias from some specific user(s), the agent randomly samples a subset of users (called participants) U_l from $\mathcal{U}$ to participate in the learning process (Line 11). We let $|U_l| = \lceil 2^{\alpha l} \rceil$, where the user-sampling parameter $\alpha > 0$ is an input of the algorithm. Recall that H_l denotes the number of batches in the l-th phase. Each participant $u \in U_l$ collects their local reward observations of each chosen action $\mathbf{a} \in \mathbf{A}_{H_l}$ and send the average $y_l^u(\mathbf{a})$ for every chosen action $\mathbf{a} \in \mathbf{A}_{H_l}$ as feedback to the agent, i.e., $g(\{y_{u,t}\}_{t \in \mathcal{T}_l}) = \mathbf{y}_l^u \triangleq [y_l^u(\mathbf{a})]_{\mathbf{a} \in \mathbf{A}_{H_l}}$. Note that the dimension of the feedback depends on the number of batches, which is also the communication cost associated with each participant (Eq. (5.2)). Therefore, *by employing the idea of rare switching, we reduce both computation complexity and communication cost (ⓒ).*

3) Action elimination (Lines 16-20): Aggregate (specifically, average) the feedback from the participants for each action $\mathbf{a} \in \mathbf{A}_{H_l}$ (Line 16). Then, using the aggregated feedback (i.e., the averaged local reward $\bar{\mathbf{y}}_l = [y_l(\mathbf{a}_1), \ldots, y_l(\mathbf{a}_{H_l})]$ of the chosen actions $\mathbf{a} \in \mathbf{A}_{H_l}$), the agent can compute the posterior mean function reformulated as follows (Line 18):

$$\bar{\mu}_l(\mathbf{x}) \triangleq \mathbf{k}(\mathbf{x}, \mathbf{A}_{H_l})^\top (\mathbf{K}_{\mathbf{A}_{H_l}\mathbf{A}_{H_l}} + \lambda \mathbf{W}_{H_l}^{-1})^{-1} \bar{\mathbf{y}}_l. \tag{5.11}$$

Considering the bias in the feedback due to user heterogeneity (ⓑ), we carefully construct a confidence width $w_l(\cdot)$ that incorporates both the noise and bias as follows:

$$w_l(\mathbf{x}) \triangleq \sqrt{\frac{2k(\mathbf{x}, \mathbf{x}) \log(1/\beta)}{|U_l|}} + \sqrt{\frac{2\Sigma_{H_l}^2(\mathbf{x}) \log(1/\beta)}{|U_l|}} + B\Sigma_{H_l}(\mathbf{x}), \tag{5.12}$$

where B is the bound of f's kernel norm, and β is the confidence level from the input. Using this confidence width $w_l(\cdot)$ and the mean estimator function $\bar{\mu}_l(\cdot)$ in Eq. (5.11), we can identify suboptimal actions with high probability (*w.h.p.*). Finally, we update the set of

active actions $\mathcal{D}_{l+1}$ by eliminating the suboptimal actions from $\mathcal{D}_l$ (Line 19).

Remark 5.1 (Merge batches)**.** For implementation, we also merge different batches with the same chosen action in each phase. By doing this, we further shrink the dimension of the matrix in the inverse operation (thus reducing the time complexity) and also the dimension of local feedback (thus reducing the communication cost).

Remark 5.2 (General decision set)**.** Following the techniques used in [93], `DPBE` can also be extended from a finite domain to a continuous domain (e.g., $\mathcal{D} = [0,1]^d$) via a simple discretization trick and Lipschitz continuity of functions under commonly used kernels.

5.5 Main Results

In this section, we present the performance of our proposed `DPBE` algorithm in terms of regret, computation complexity, and communication cost, respectively.

First, we analyze the regret performance of `DPBE` and present the upper bound in Theorem 5.3. While the `DPBE` algorithm uses GP tools to define and manage the uncertainty in estimating the unknown function f, the analysis of `DPBE` algorithm does not rely on any *Bayesian* assumption about f being drawn from the prior $\mathcal{GP}(0, k(\cdot,\cdot))$, and it only requires f to be bounded in the kernel norm associated with the RKHS $\mathcal{H}_k$.

Theorem 5.3 (Regret)**.** *Let $\beta = \frac{1}{|\mathcal{D}|T}$. Under Assumptions 1, 2 and 3, the* `DPBE` *algorithm achieves the following expected regret:*

$$\mathbb{E}[R(T)] = O(T^{1-\alpha/2}\sqrt{\log(|\mathcal{D}|T)}) + O(\sqrt{\gamma_T T}) + O(\sqrt{\gamma_T T^{1-\alpha}\log(|\mathcal{D}|T)}). \tag{5.13}$$

We provide the detailed proof of Theorem 5.3 in Appendix D.3. Bounds for γ_T of different

kernels can be found in Appendix D.1.2. In the following, we make two remarks about the above result.

Remark 5.4. In the above regret upper bound, the first term, $O(T^{1-\alpha/2}\sqrt{\log(|\mathcal{D}|T)})$, is due to the bias in the feedback at heterogeneous participants, and the last two terms, $O(\sqrt{\gamma_T T})$ + $O(\sqrt{\gamma_T T^{1-\alpha}\log(|\mathcal{D}|T)})$, are from the noisy feedback of each action as in the standard kernelized bandits (cf. [125]). Note that the first term (i.e., the regret caused by the bias) can be improved if one increases the number of sampled users in the learning process (i.e., choosing a larger value of α). However, this would also result in a larger communication cost.

Remark 5.5 (Maximum Uncertainty Reduction)**.** Recall that `DPBE` selects actions that have maximum variance for each batch (Eq. (5.5)). Intuitively, variance at action $\mathbf{x}$ indicates the uncertainty about $f(\mathbf{x})$, and thus, maximum-variance selection leads to maximum uncertainty reduction, which promotes exploration.

Remark 5.6 ((Sub-)optimality)**.** We first note that one natural lower bound for our setting is the one for the standard setting of kernelized bandits, where the agent receives unbiased feedback after taking an action. In this setting, the state-of-the-art lower bounds under two commonly-used kernel functions (SE and Matérn)[3] are summarized in Table D.1 (see Appendix D.1.2), which can also serve as valid lower bounds for the setting we consider. Recall that $\alpha > 0$ is the user-sampling parameter that one can choose. We discuss the (sub-)optimality of our upper bounds in two cases: $\alpha \geq 1$ (i.e., the high-communication regime) and $\alpha \in (0, 1)$ (i.e., the low-communication regime). (i) In the high-communication regime, the upper bound in (5.13) now becomes $O(\sqrt{\gamma_T T})$, which is *near-optimal* under both SE and Matérn kernels. In particular, if one plugs the best-known bounds on γ_T for

[3]Note that even for the standard setting of kernelized bandits, there only exist lower bounds for these specific kernel functions rather than a general one in terms of the maximum information gain γ_T.

SE and Matérn kernels (as listed in the first column in Table D.1; also see [134]) into the regret upper bound $O(\sqrt{\gamma_T T})$, one can now have explicit regret upper bounds (as listed in the third column in Table D.1), which match the corresponding lower bounds, up to only a logarithmic factor. (ii) In the low-communication regime, the first term in the regret upper bound (see Eq. (5.13) in Theorem 5.3) that depends on α may be dominant and cannot be ignored. On the other hand, the existing lower bounds do not depend on α since they are derived under the standard setting of kernelized bandits, where user sampling is irrelevant. Therefore, an important open problem is to close the gap by deriving tighter lower and/or upper bounds that capture the effect of user sampling in the new setting with distributed biased feedback we consider. We leave it as our future work.

As a critical bottleneck of kernelized bandits algorithms, the computation complexity of `DPBE` algorithm is analyzed in the following Theorem 5.7.

Theorem 5.7 (Computation complexity)**.** *The computation complexity of* `DPBE` *is at most* $O(\gamma_T T^\alpha + (|\mathcal{D}|\gamma_T^3 + \gamma_T^4)\log T)$.

Proof. Recall that H_l is the number of batches in the l-th phase. Then, the computation complexity of the central agent in the l-th phase is upper bounded by the following:

$$O(H_l \cdot H_l^3 + H_l \cdot |\mathcal{D}_l| H_l^2 + |U_l| H_l + |\mathcal{D}_l| H_l^2).$$

Specifically, for each $h \in [H_l]$ within phase l, the agent would compute the matrix inverse in (5.7), which takes at most $O(h^3) \leq O(H_l^3)$. With this matrix inverse result ready, the agent can solve the maximum-variance problem in Eq. (5.8) with at most $O(|\mathcal{D}_l| H_l^2)$ for each batch and determine the batch length $\mathcal{T}_l(\mathbf{a}_h)$ with $O(1)$ after we have the posterior variance. Since there is a total of H_l batches for phase l, the total complexity up to this stage is $O(H_l \cdot H_l^3 + H_l \cdot |\mathcal{D}_l| H_l^2)$. Finally, in the elimination stage for phase l, the agent first

Table 5.1: Comparison of computation complexity under DPBE and three state-of-the-art algorithms.

Algorithms	Complexity
GP-UCB [37]	$O(\lvert\mathcal{D}\rvert T^3)$
BBKB [22]	$O(\lvert\mathcal{D}\rvert T\gamma_T^2)$
mini-GP-Opt [23]	$O(T + \lvert\mathcal{D}\rvert\gamma_T^3 + \gamma_T^4)$
DPBE (this work)	$O(\gamma_T T^\alpha + (\lvert\mathcal{D}\rvert\gamma_T^3 + \gamma_T^4)\log T)$

loads/aggregates all the feedbacks with $O(|U_l|H_l)$ and can again reuse the matrix inverse result so that only $O(|\mathcal{D}_l|H_l^2)$ is required to eliminate all the bad arms.

Putting the two stages together, we have the above result. Thus, it remains to bound the number of batches H_l within each phase l. Fortunately, inspired by [23], we are able to show that H_l can be upper bounded by the maximum information gain. We state this result in Lemma 5.8 and provide the proof in Appendix D.4.

Lemma 5.8 (Bound on H_l)**.** *For any phase l, the number of batches H_l is at most $\frac{4\sigma^2C^2}{C^2-1}\gamma_T$.*

We can get that the total number of phases is $O(\log T)$ and the total number of participants satisfies $O(T^\alpha)$. Armed with all the above results, we arrive at our final computation complexity. □

Remark 5.9 (Complexity comparison)**.** For comparison, we list the computation complexity of the state-of-the-art algorithms for standard kernelized bandits in Table 5.1. As we already know, GP-UCB has a computation complexity of $O(|\mathcal{D}|T^3)$, because it requires computing the posterior mean and variance using $O(T^2)$ and then finds the action that maximizes the UCB function per step. Recently, BBKB in [22] improves the time complexity to $(|\mathcal{D}|T\gamma_T^2)$, and later mini-GP-Opt in [23] further reduces computation complexity to $O(T + |\mathcal{D}|\gamma_T^3 + \gamma_T^4)$, which is currently the fastest no-regret algorithm. Although more feedback is needed to address the additional bias in our setting, our algorithm can still achieve an improvement

with the highest order term being $O(\gamma_T T^\alpha)$. This improvement comes from the fact that the participants help preprocess local reward observations before sending them out.

Meanwhile, the bound on H_l also allows us to achieve a meaningful communication cost.

Theorem 5.10 (Communication cost)**.** *DPBE incurs at most $O(\gamma_T T^\alpha)$ communication cost.*

The proof for Theorem 5.10 is also provided in Appendix D.4.

Remark 5.11 (Communication cost when merging batches)**.** By further merging batches according to Remark 5.1, the `DPBE` algorithm incurs $O(\min\{\gamma_T, |\mathcal{D}|\}T^\alpha)$ communication cost; We highlight that the batch schedule strategy plays a key role in obtaining the above bounds. Otherwise, even merging rounds as Remark 5.1 with the reformulated representation in Eqs. (5.7) and (5.11), the dimension of the local feedback at each participant is $O(\min\{T_l, |\mathcal{D}_l|\})$ in order to distinguish different actions, which leads to $O(\min\{T, |\mathcal{D}|\}T^\alpha)$ (vs. ours $O(\min\{\gamma_T, |\mathcal{D}|\}T^\alpha)$).

5.6 Differential Private `DPBE`

As privacy is also an important factor in distributed learning, it is critical to protect users' sensitive data when collecting and aggregating their feedback. For example, in the dynamic pricing application, it is required that an adversary cannot infer a customer's private information (e.g., purchase or not) by observing the pricing mechanism set by the company. Moreover, users may require more stringent privacy protection in some applications — users are not willing to share their perceived Quality-of-Experience (QoE) directly with the central controller in the cellular network configuration problem; citizens are not willing to reveal the information about their preference for a certain policy to the government. Formally, we

adopt the concept of *differential privacy* (DP) [50] as the privacy metric. Thanks to the phase-then-batch schedule strategy in our algorithm, different DP trust models (e.g., central [50], local [144], and shuffle [34]) can be applied through proper designs. In this section, we describe how to ensure DP under DPBE with a trusted agent (the central DP model) and also analyze the regret under such a DP model. Extensions of the differentially private DPBE algorithms in other DP models (e.g., the stronger local DP model) are presented in Appendix D.5.

5.6.1 DP Definition and Algorithm

In the central DP model, we assume that each participating user trusts the agent, and hence, the agent can collect their raw data (i.e., the local reward $\mathbf{y}_l^u$ in our case). The privacy guarantee is that any adversary with arbitrary auxiliary information cannot infer a particular user's data by observing the decisions of the agent. To achieve this privacy protection, the central DP model requires that the decisions of the agent on two neighboring sets of users (differing in only one user) are indistinguishable [51]. Formally, we have the following definition.

Definition 5.12. (Differential Privacy (DP)). For any $\varepsilon \geqslant 0$ and $\delta \in [0, 1]$, a randomized algorithm $\mathcal{M}$ is (ε, δ)-differentially private (or (ε, δ)-DP) if for every pair of $U, U' \subseteq \mathcal{U}$ differing on a single participant and for any subset of output actions $\mathbf{Z} = [\mathbf{z}_1^\top, \ldots, \mathbf{z}_T^\top]^\top$, we have

$$\mathbb{P}[\mathcal{M}(U) = \mathbf{Z}] \leqslant e^{\varepsilon}\mathbb{P}[\mathcal{M}(U) = \mathbf{Z}] + \delta. \tag{5.14}$$

The parameters ε and δ indicate how private $\mathcal{M}$ is; the smaller, the more private. According to the post-processing property of DP (cf. Proposition 2.1 in Dwork and Roth [50]), it suffices to guarantee that the aggregator (Line 16 in Algorithm 6) is (ε, δ)-DP. To achieve this, the

standard Gaussian mechanism can be applied by adding Gaussian noise to the aggregated distributed feedback. Then, the *private* aggregated feedback for the chosen actions in the l-th phase becomes

$$\tilde{\mathbf{y}}_l = \bar{\mathbf{y}}_l + (\rho_1, \dots, \rho_{H_l}), \tag{5.15}$$

where $\rho_j \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma_{nc}^2)$, and the variance σ_{nc}^2 is based on the ℓ_2 sensitivity of the average vector $\bar{\mathbf{y}}_l$. In addition, we replace $\bar{\mathbf{y}}_l$ with $\tilde{\mathbf{y}}_l$ in Eq. (5.15) to obtain the private mean estimator $\tilde{\mu}_l(\cdot)$:

$$\tilde{\mu}_l(\mathbf{x}) \triangleq \mathbf{k}(\mathbf{x}, \mathbf{A}_{H_l})^\top (\mathbf{K}_{\mathbf{A}_{H_l}\mathbf{A}_{H_l}} + \lambda \mathbf{W}_{H_l}^{-1})^{-1} \tilde{\mathbf{y}}_l. \tag{5.16}$$

The confidence width function is also updated by counting the uncertainty introduced by privacy noise as follows:

$$\tilde{w}_l(\mathbf{x}) \triangleq \sqrt{\frac{2k(\mathbf{x},\mathbf{x})\log(1/\beta)}{|U_l|}} + \sqrt{\frac{2\Sigma_{H_l}^2(\mathbf{x})\log(1/\beta)}{|U_l|}} + B\Sigma_{H_l}(\mathbf{x}) + \sqrt{2\sigma_n^2 \log(1/\beta)}, \tag{5.17}$$

where σ_n is related to the overall privacy noise and will be specified with our theoretical results. The details of the differentially private `DPBE` algorithm, called `DP-DPBE`, are presented in Algorithm 10 in Appendix D.5.

5.6.2 Performance Guarantees

In the following, we provide the main results of the `DP-DPBE` algorithm in terms of privacy guarantee and regret. We start by stating an additional assumption in Assumption 4. This one-time participation assumption is commonly used in private bandits (see, e.g., [45, 101, 117, 129]). To handle multiple-times participation, one can use (adaptive) composition theorem of differential privacy [50].

Assumption 4. Each sampled user only participates in one phase of the learning process.

Then, we present the privacy guarantee in Theorem 5.13 and provide the proof in Appendix D.5.2.

Theorem 5.13 (Privacy Guarantee)**.** *Under Assumptions 1, 2, 3, and 4, the* `DP-DPBE` *algorithm in Algorithm 10 with* $\sigma_{nc} = \frac{2\sqrt{2(\kappa^2+\sigma^2)H_l \log(2H_l/\delta_1)\ln(1.25/\delta_2))}}{\varepsilon|U_l|}$ *in the central model guarantees* (ε, δ)*-DP where* $\delta = \delta_1 + \delta_2$*.*

As an additional Gaussian noise is injected to protect privacy, `DP-DPBE` suffers additional regret cost. We present its regret upper bound in Theorem 5.14.

Theorem 5.14 (Regret of `DP-DPBE`)**.** *Let* $\sigma_{nc} = \frac{2\sqrt{2(\kappa^2+\sigma^2)H_l \log(2H_l/\delta_1)\ln(1.25/\delta_2))}}{\varepsilon|U_l|}$*. Under Assumptions 1, 2, and 3, the* `DP-DPBE` *algorithm with* $\beta = \frac{1}{|\mathcal{D}|T}$ *and* $\sigma_n = \sigma_{nc}\sqrt{2C^2\gamma_T}$ *achieves the following expected regret*

$$\mathbb{E}[R(T)] = O(T^{1-\alpha/2}\sqrt{\log(|\mathcal{D}|T)}) + O\left(\frac{\ln(1/\delta)\gamma_T T^{1-\alpha}\sqrt{\log(|\mathcal{D}|T)}}{\varepsilon}\right). \tag{5.18}$$

The full proof of Theorem 5.14 is provided in Appendix D.5.3. Regarding this regret result, we make the following remark.

Remark 5.15 (Privacy "for free")**.** Comparing Theorem 5.14 with Theorem 5.3, we see that the additional regret cost introduced by privacy noise is $\tilde{O}\left(\frac{\ln(1/\delta)\gamma_T T^{1-\alpha}}{\varepsilon}\right)$, which is a lower order term compared to the first non-private term. This implies that our `DP-DPBE` algorithm enables us to achieve a privacy guarantee "for free" in the kernelized bandits setting. The same observation of achieving privacy "for free" is also observed in Chapter 4 for distributed linear bandits setting. Here, our results show that it holds for general functions and recovers the result in Chapter 4 when considering a linear kernel.

Remark 5.16 (Other DP models)**.** In the cases where the users do not trust the agent, users' data privacy has to be protected by the users themselves as in a local DP model or

by resorting to a third party, e.g., the shuffler in the shuffle DP model. In Appendix D.5, we make extensions of `DP-DPBE` by considering these two trust models. In the local model, a local randomizer is equipped with each participant so that the feedback from each user is private. While the local model ensures a stronger privacy guarantee compared to the central DP, it always incurs a larger additional regret cost. Meanwhile, thanks to the phase-based strategy in `DPBE`, DP-DPE can also be easily extended to the shuffle model,which achieves better regret-privacy tradeoff as in [89], i.e., achieving nearly the same regret as the central model, yet without the need to assume a trustworthy agent.

5.7 Numerical Experiments

We now evaluate our proposed approach empirically on three types of functions: 1) synthetic functions in the RKHS with an SE kernel, 2) standard benchmark functions (with an unknown RKHS norm) [127] , and 3) functions from a real-world dataset. We implement the algorithms in `python` and run the numerical experiments on a Dell desktop (Processor: Intel®Core i7 CPU, 8 cores; Memory: 32GB).

5.7.1 Synthetic Function

We follow [70] to construct the global function f from the RKHS by sampling $m = 30d$ independent points, $\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_m$, uniformly on $[0,1]^d$, and $\hat{a}_1, \ldots, \hat{a}_m$, uniformly on $[-1,1]$, and defining $f(\mathbf{x}) = \sum_{i=1}^m \hat{a}_i k(\hat{\mathbf{x}}_i, \mathbf{x})$ for all $\mathbf{x} \in \mathcal{D}$, where k is SE kernel with length-scale $l_{SE} = 0.2$. The RKHS norm is $\|f\|_k^2 = \sum_{i=1}^m \sum_{j=1}^m \hat{a}_i \hat{a}_j k(\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j)$, which is assumed to be known. Each local reward function f_u, a random function sampled from a given Gaussian process, is generated by following Algorithm 1 in [75]. In the simulations, we evaluate the

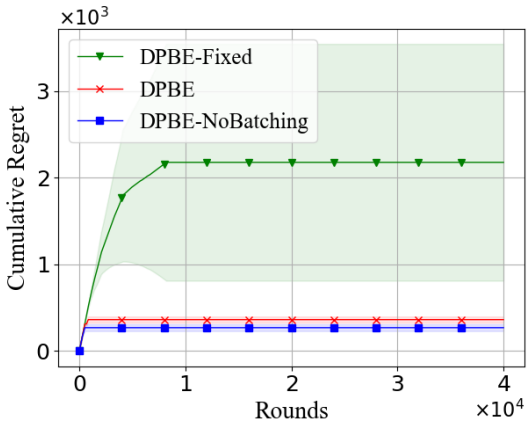


Figure 5.2: Comparison of regret performance on a synthetic function. The shaded area represents the standard deviation

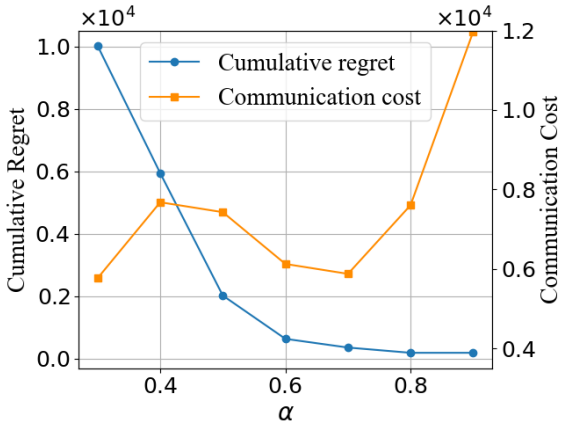


Figure 5.3: The regret and communication cost under DPBE with different values of α.

algorithms in a more general setting with $f_u \sim \mathcal{GP}(f(\cdot), v^2 k(\cdot,\cdot))$, where v^2 is a scaling parameter that can be used to set a reasonable level of local bias (see Footnote 2).

Ablation Studies and Analysis.

First, we show that the DPBE algorithm that selects actions according to maximum variance reduction achieves sublinear regret, as shown in Figure 5.2. Then, we perform numerous ablation studies to confirm the efficacy of other two key components in our algorithm: user-sampling and batching strategy. To this end, we consider the corresponding variants of our algorithm. In this simulation, we perform 20 runs for each algorithm by setting $|\mathcal{D}| = 100, d = 3, C = 1.6$, $\sigma = 0.01$, $v = 0.1$, $T = 40000$, $\alpha = 0.7$, $\beta = 1/(|\mathcal{D}|T)$ and $\lambda = \sigma^2/v^2$ and present the regret performance in Figure 5.2 and communication cost and runtime in Table 5.2.

1) Importance of (exponentially-increasing) user-sampling. To this end, we consider the first variation of DPBE with a fixed number of participants, called DPBE-Fixed, where the

Table 5.2: Comparisons of communication cost and running time under DPBE, DPBE-Fixed, and DPBE-NoBatching on a synthetic function.

Algorithms	Communication cost	Running time (seconds)
DPBE	5.87×10^3	0.12
DPBE-Fixed	5.87×10^3	0.19
DPBE-NoBatching	1.81×10^4	0.61

number of participants in each phase is fixed at $|U| = \lfloor \frac{\sum_{l=1}^{L} |U_l| * N_{u,l}}{\sum_{l=1}^{L} N_{u,l}} \rfloor$ so as to have the same communication cost as DPBE. From Figure 5.2, we observe that DPBE with exponentially-increasing user-sampling over phases performs much better than DPBE-Fixed with the same communication cost. It demonstrates that the exponentially-increasing user-sampling mechanism in DPBE is critical to striking a balance between regret and communication cost. From Table 5.2, we observe that DPBE-Fixed takes a little longer time than DPBE. This is mainly because DPBE-Fixed needs more phases to find the optimal action (i.e., L is larger when $|\mathcal{D}_L| = 1$).

2) Benefits of batching strategy. To illustrate the impact of batching schedule strategy, we consider another variant of DPBE that does not employ batching strategy, called DPBE-NoBatching. In particular, it selects an action according to Eq. (5.8) for each round in any phase. Without batching strategy, DPBE-NoBatching communicates local observations directly without merging, and computes the posterior mean and variance according to standard update formula: Eq. (5.4) and Eq. (5.5) respectively; From Figure 5.2, we observe that DPBE, similar to other *rare-switching* algorithms [4], achieves a slightly worse regret performance than DPBE-NoBatching. However, as shown in Table 5.2, it significantly saves communication cost ($\sim 3\times$) by merging local observations in batches and reduces computation time ($\sim 5\times$) by shrinking the dimension of posterior reformulations.

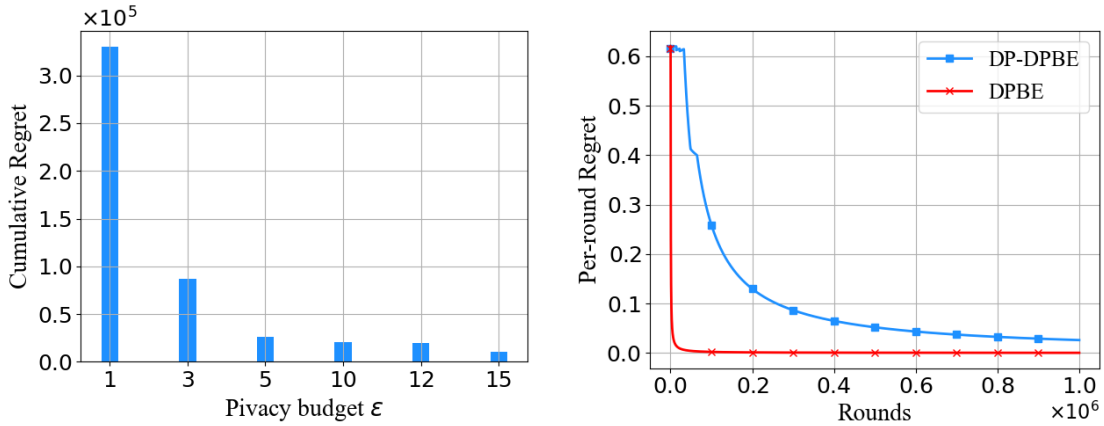
Regret-communication Tradeoff.

We now turn to investigate the regret-communication tradeoff captured by the user-sampling parameter α, as shown in Theorem 5.3.

Consider $\alpha \in \{0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$. The cumulative regret and total communication cost of `DPBE` with different values of α are presented in Figure 5.3. As expected, while a larger value of α yields a lower regret, it generally results in a higher communication cost. Notice that `DPBE` incurs slightly higher communication cost when $\alpha = \{0.4, 0.5, 0.6\}$ compared to $\alpha = 0.7$, this is mainly because `DPBE` with a smaller value of α needs more phases to find the optimal action (i.e., L is larger when $|\mathcal{D}_L| = 1$). One can tune the user-sampling parameter α to achieve a better regret-communication cost accordingly, e.g., $\alpha = 0.7$ for this synthetic function setting.

Regret-privacy Tradeoff.

Finally, we evaluate the performance of the differentially private `DPBE`, i.e., `DP-DPBE`, and present the result in Figure 5.4. Figure 5.4a shows how the cumulative regret at the end of $T = 10^6$ rounds varies with different values of the privacy parameter ε, which reveals an obvious tradeoff between regret and the privacy parameter ε. Figure 5.4b shows the regret performance of `DPBE` and `DP-DPBE` with privacy parameters $\varepsilon = 5$ and $\delta = 0.1$. We observe that although `DP-DPBE` adds extra noise to protect privacy, it can still achieve no-regret (i.e., $\lim_{T\to\infty} \frac{R(T)}{T} \to 0$). Indeed, to protect privacy, `DP-DPBE` requires much more time to find the optimal action, which is the typical regret-privacy tradeoff. However, for a large T, the gap compared to the non-private one is small, which also validates the privacy "for-free" result.



(a) (b)

Figure 5.4: Performance of `DP-DPBE`. (a) Final cumulative regret vs. the privacy budget ε with $\delta = 0.1$; (b) Per-round regret vs. time with parameters $\varepsilon = 5$ and $\delta = 0.1$.

5.7.2 Standard Benchmark Functions

In addition, we study the performance of `DPBE` on standard optimization benchmark functions. This corresponds to a more realistic setting where the RKHS norm of the target function is unknown in advance. In particular, we use three common functions in global optimization problems [127]: (a) Sphere function, (b) Six-hump Camel function, and (c) Michalewicz function, and provide the performance comparison of `DPBE-Fixed`, `DPBE`, and `DPBE-NoBatching` in Figure 5.5 and Table 5.3. In the simulations, we scale the range of the function values to $[-1, 1]$ and use RKHS norm $B = 1$ in the algorithms as in [70]. Without knowing the exact kernel of the target function, each local reward function f_u is constructed by sampling a function from the GP $\mathcal{GP}(f(\cdot), v^2 k(\cdot, \cdot))$, where we choose $v^2 = 0.001$ and use the SE kernel with $l_{SE} = 0.2$. In addition, we set $T = 4 \times 10^4$ and $|\mathcal{D}| = 100$ and run each algorithm on each function for 20 times.

From Figure 5.5 and Table 5.3, we observe similar results to those of the synthetic function with the same kernel. First, compared to `DPBE-Fixed` that incurs the same communication cost, `DPBE` might perform slightly worse at the very beginning (e.g., Figure 5.5a) but eventu-

Table 5.3: Communication cost and running time under `DPBE`, `DPBE-Fixed`, and `DPBE-NoBatching`

Function	Algorithm	Communication cost	Running time (seconds)
Sphere	`DPBE`	1.49×10^3	0.07
	`DPBE-Fixed`	1.49×10^3	0.12
	`DPBE-NoBatching`	6.16×10^3	0.69
Six-Hump Camel	`DPBE`	1.26×10^3	0.03
	`DPBE-Fixed`	1.26×10^3	0.12
	`DPBE-NoBatching`	1.45×10^4	0.17
Michalewicz	`DPBE`	2.06×10^3	0.06
	`DPBE-Fixed`	2.06×10^3	0.14
	`DPBE-NoBatching`	2.73×10^4	0.49
Light Sensor Data	`DPBE`	5.17×10^3	0.22
	`DPBE-Fixed`	5.17×10^3	0.28
	`DPBE-NoBatching`	2.73×10^4	5.20

ally achieves a much smaller regret. Note that `DPBE-Fixed` may not be able to find the optimal action by the end of T (e.g., Figure 5.5b). This phenomenon strengthens our argument on the exponentially-increasing user-sampling mechanism in `DPBE`. While `DPBE-NoBatching` has slightly better regret performance than `DPBE`, it incurs much higher communication cost ($5 \sim 13\times$) and requires a much longer time ($6 \sim 23\times$, see running time column in Table 5.3), which demonstrates the key benefits of the batching strategy in improving communication efficiency and computation complexity.

In addition, we also evaluate the regret-privacy tradeoff under `DP-DPBE`. Due to space limitations, we present the numerical results in Appendix D.6 (see Figures D.1 and D.2).

5.7.3 Functions from Real-World Data

We also evaluate the performance of `DPBE` on a function from a real-world dataset, where there is no explicit closed-form expression.

Light Sensor Data. We use the light sensor data collected from the CMU Intelligent Workplace in November 2005, which is available online [118]. It contains locations of 41 sensors, 601 training samples, and 192 testing samples. Following [37, 125, 143], we compute the empirical covariance matrix of the training samples and use it as the kernel matrix in the algorithm. Here, for each location $\mathbf{x}$, we let $f(\mathbf{x})$ be the average of the normalized sample readings at $\mathbf{x}$ and set $B = \max_{\mathbf{x}} f(\mathbf{x})$ in the algorithm. For this function (from real data), we construct each local function f_u by sampling a function from a Gaussian process with mean f and the kernel constructed above, and set the noise in the local feedback as $\sigma = 0.01$ and the bias in each local feedback as $v = 0.1$. We run `DPBE` with input parameters $\alpha = 0.7, \beta = 1/(|\mathcal{D}|T)$, and $\lambda = \sigma^2/v^2$, and present the regret performance in Figure 5.5d and communication cost and running time in Table 5.3. The observations are qualitatively similar to those made in simulations on other functions: `DPBE` outperforms `DPBE-Fixed` in regret given the same communication cost and achieves a regret close to `DPBE-NoBatching`, which has much longer running time. Besides, we also run `DP-DPBE` on this real-world dataset and present the results in Appendix D.6 (see Figures D.1d and D.2d), which validates the regret-privacy tradeoff.

5.8 Comparison with the State-of-the-Arts

5.8.1 Discussion

We now consider an alternative way of addressing kernelized bandits with distributed biased feedback. One may incorporate the local bias as another level of noise added to the noise in the rewards as a new noisy measurement of the global function f with a larger variance. In this case, the state-of-the-art algorithms for the traditional kernelized bandits [37, 125] may

be adapted to our setting. However, they have some key limitations.

Consider two representative state-of-the-art algorithms: `GP-UCB` [37] and `BPE` [93]. `GP-UCB` is one of the most commonly used algorithms for standard kernelized bandits, It was proposed in [125] and improved in [37]. By resorting to the Gaussian process surrogate model (see Section 5.3.3), `GP-UCB` adaptively selects the action with the maximal *upper confidence bound* in each round based on historical observations up to the current round. `BPE` is a batch-based algorithm that eliminates suboptimal actions batch by batch, and within each batch, actions are chosen independently from reward observations. In the following, we compare our proposed `DPBE` algorithm with `GP-UCB` and `BPE` (adapted to our setting) and show their limitations in user-sampling, communication cost, and computation complexity.

First, both `GP-UCB` and `BPE` require to collect feedback from one user per step, which results in T users involved in the learning process. In practice, even though there is a large population, not all users are willing to send their feedback. Hence, it may not be feasible to collect feedback from too many users. In our algorithm, instead of sampling more users to reduce the overall uncertainty, we ask each sampled user (who is more willing to participate) to participate in more rounds and send their feedback. In this way, we alleviate the user-sampling burden by letting the participating users collect more reward samples of the chosen actions. However, due to the bias in the feedback of each user, we could not just sample one user and then let her report the feedback during the entire horizon. We need to balance the tradeoff between sampling more users and letting the users participate in more rounds.

Second, by collecting feedback in each round, both `GP-UCB` and `BPE` incur a very high communication cost of T. Instead, we employ a phase-based communication protocol where feedback corresponding to any particular action at each participant is averaged and only communicated at the end of each phase. Then, the total communication cost depends on the number of phases, the number of distinct actions in each phase, and the number of sampled

users. The smaller each of these three factors, the smaller the communication cost. By carefully designing the algorithm, we can reduce the communication cost to $O(\min\{\gamma_T, |\mathcal{D}|\}T^\alpha)$, where $\alpha \in (0, 1)$ is the user-sampling parameter one can choose.

Finally, at each round t, `GP-UCB` finds the decision action $\mathbf{x}_t$ that maximizes an acquisition function (specifically, the UCB index, which is the sum of the posterior mean and variance). Note that obtaining the posterior mean and variance requires computing matrix inverse (see Eqs. (5.4) and (5.5)), which still has a computation complexity of $O(t^2)$ even using rank-one recursive updates [37, Appendix 7]. Hence, the overall computation complexity of `GP-UCB` is $O(T^3)$. Similarly, `BPE` may also compute the posterior variance using the rank-one recursive update within each batch, and then the total computation complexity depends on the batch size and the number of batches. As in [93], the batch size is updated as $N_i = \sqrt{T\sqrt{N_{i-1}}}$, initialized with $N_0 = 1$, which results in $\lceil \log\log(T) \rceil$ batches in total. Therefore, the computation complexity of `BPE` is $O(T^3)$. In our design, we employ the batch schedule strategy and reformulate the posterior mean and variance as Eqs. (5.11) and (5.7), where the dimension of the matrix becomes much smaller. This leads to a much smaller overall computation complexity of $O(\gamma_T T^\alpha + (|\mathcal{D}|\gamma_T^3 + \gamma_T^4)\log T)$.

5.8.2 Empirical Performance

In this subsection, we evaluate the empirical performance of `DPBE` with different values of user-sampling parameter α compared to `GP-UCB` and `BPE`.

The simulations are run on the same three types of functions as in the preceding section: the synthetic function in Section 5.7.1, the standard benchmark functions in Section 5.7.2, and the function from light sensor data in Section 5.7.3. Due to space limitation, we only present the results of synthetic function here and put the results of the latter two types of

Table 5.4: Comparison of running time (seconds) under GP-UCB, BPE, and DPBE with different values of α.

Algorithms	DPBE $\alpha = 0.4$	DPBE $\alpha = 0.5$	DPBE $\alpha = 0.6$	DPBE $\alpha = 0.7$	DPBE $\alpha = 0.8$	DPBE $\alpha = 0.9$	GP-UCB	BPE
Running time	0.24	0.19	0.14	0.12	0.13	0.17	5.32	27.49

functions in Appendix D.6.

Consider[4] $\alpha \in \{0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$ for DPBE. We show the empirical regret performance of all algorithms in Figure 5.6 and the running time in Table 5.4. From Figure 5.6, we observe that the empirical regret performance of DPBE can be fairly close to or even better than that of GP-UCB and BPE via properly choosing parameter α. However, it consumes much less time for DPBE with each $\alpha \in \{0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$ than both GP-UCB and BPE. For example, while DPBE takes about 0.15 second in most scenarios, GP-UCB takes more than 5 seconds, which is more than 30 times slower. BPE takes around 27 seconds, which is even slower.

Recall the empirical communication cost of DPBE with different values of α shown in Figure 5.3. While the communication cost of GP-UCB and BPE is 4×10^4 (specifically, one feedback per round), DPBE incurs a much smaller communication cost even when $\alpha = 0.9$ (4×10^4 vs. 1.19×10^4).

In summary, the comparison of empirical performance under DPBE with GP-UCB and BPE demonstrates the significant improvements of DPBE in terms of communication cost and computation complexity, although little regret performance is sacrificed when α is not big enough.

[4]Note that the smaller the value of α, the larger the cumulative regret. In Figure 5.6, we omit the regret performance when $\alpha < 0.4$ since they are much larger than others.

5.9 Conclusion

In this chapter, we studied a new kernelized bandits problem with distributed biased feedback, where the feedback of the unknown objective function is biased due to user heterogeneity. To learn and optimize the unknown function using distributed biased feedback, we proposed the learning with communication framework. Considering the communication cost for collecting feedback and the computational bottleneck of kernelized bandits, we carefully designed the distributed phase-then-batch-based elimination (`DPBE`) algorithm to address all the new challenges. Specifically, `DPBE` selects actions according to maximum variance reduction, reduces bias via user-sampling, and improves communication efficiency and computation complexity via the batching strategy. Furthermore, we showed that `DPBE` achieves a sublinear regret while being scalable in terms of communication efficiency and computation complexity. Finally, we generalized `DPBE` to incorporate various differential privacy models to ensure privacy guarantees for participating users.

Algorithm 6 Distributed Phase-then-Batch-based Elimination (`DPBE`)

Input: $\mathcal{D} \subseteq \mathbb{R}^d$, parameters $\alpha > 0$, $\beta \in (0,1)$, C, and local noise σ^2
Initialization: $l = 1$, $\mathcal{D}_1 = \mathcal{D}$, $t_1 = 0$, and $T_1 = 1$
while $t_l < T$ **do**
  Set $\tau = 1$, $h = 1$, $\tau_1 = 0$ and $\Sigma_0^2(\mathbf{x}) = k(\mathbf{x}, \mathbf{x})$, for all $\mathbf{x} \in \mathcal{D}_l$
  while $\tau \leqslant T_l$ **do**
    Choose action

$$\mathbf{a}_h \in \operatorname*{argmax}_{\mathbf{x} \in \mathcal{D}_l} \Sigma_{h-1}^2(\mathbf{x}) \tag{5.8}$$

    Play action $\mathbf{a}_h$ for $T_l(\mathbf{a}_h) \triangleq \lfloor (C^2 - 1)/\Sigma_{h-1}^2(\mathbf{a}_h) \rfloor$ rounds if not reaching $\min\{T, t_l + T_l\}$
    Update $h = h + 1$, $\tau = \tau + T_l(\mathbf{a}_h)$, and incorporate $\mathbf{a}_h$ into $\Sigma_h^2(\cdot)$ in Eq. (5.7)
  end while
  Let $H_l = h$ denote the total number of batches in this phase
  Randomly select $\lceil 2^{\alpha l} \rceil$ participants U_l
  # Operations at each participant
  for each participant $u \in U_l$ **do**
    Collect and compute local average reward for every chosen action $\mathbf{a} \in \mathbf{A}_{H_l}$:

$$y_l^u(\mathbf{a}) = \frac{1}{T_l(\mathbf{a})} \sum_{t \in \mathcal{T}_l(\mathbf{a})} y_{u,t}$$

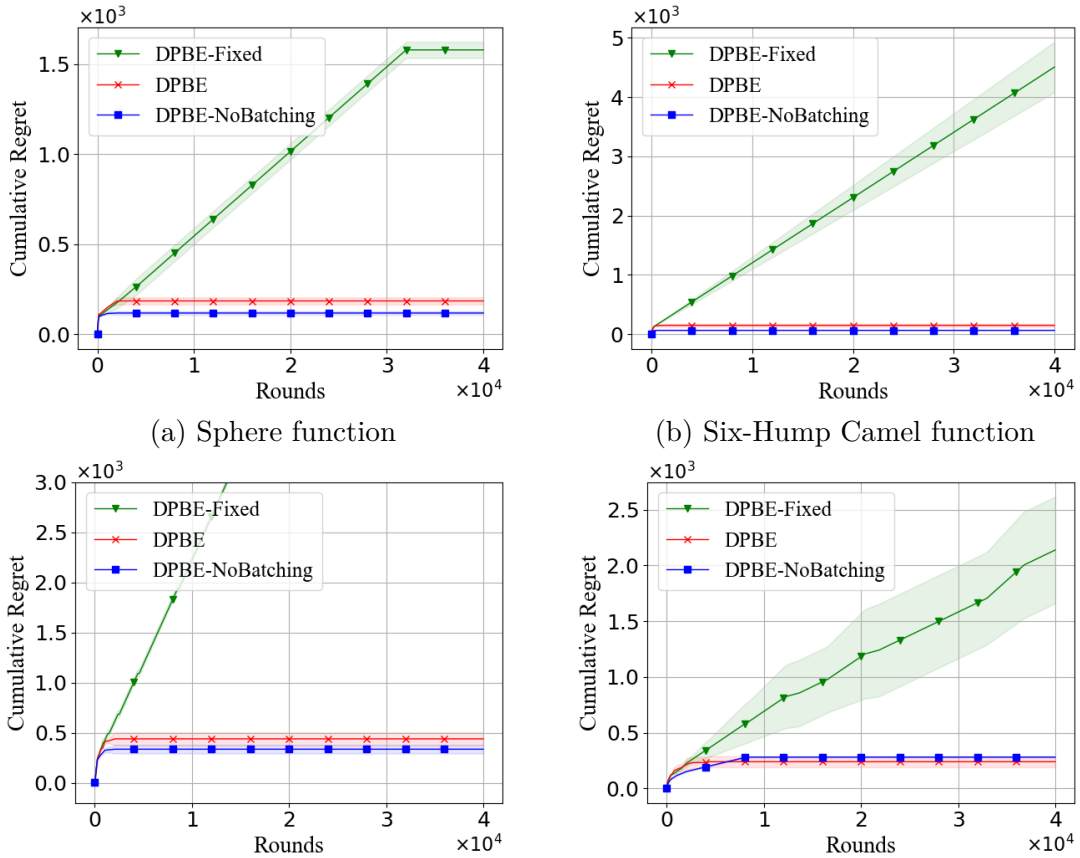
    Send the average reward for each chosen action $\mathbf{y}_l^u \triangleq [y_l^u(\mathbf{a})]_{\mathbf{a} \in \mathbf{A}_{H_l}}$ to the agent
  end for
  Aggregate local observations for each chosen action $\mathbf{a} \in \mathbf{A}_{H_l}$:

$$y_l(\mathbf{a}) = \frac{1}{|U_l|} \sum_{u \in U_l} y_l^u(\mathbf{a}) \tag{5.9}$$

  Let $\bar{\mathbf{y}}_l = [y_l(\mathbf{a}_1), \ldots, y_l(\mathbf{a}_{H_l})]$
  Update $\bar{\mu}_l(\cdot)$ according to Eq. (5.11)
  Eliminate low-rewarding actions from $\mathcal{D}_l$ based on the confidence width $w_l(\cdot)$ in Eq. (5.12):

$$\mathcal{D}_{l+1} = \left\{ \mathbf{x} \in \mathcal{D}_l : \bar{\mu}_l(\mathbf{x}) + w_l(\mathbf{x}) \geqslant \max_{\mathbf{b} \in \mathcal{D}_l} (\bar{\mu}_l(\mathbf{b}) - w_l(\mathbf{b})) \right\} \tag{5.10}$$

  $T_{l+1} = 2T_l$, $t = t + T_l$, $l = l + 1$
end while



(a) Sphere function

(b) Six-Hump Camel function

(c) Michalewicz function

(d) Light sensor data

Figure 5.5: Comparison of regret performance under `DPBE`, `DPBE-Fixed`, and `DPBE-NoBatching` on four functions. (a) Sphere function. Settings: $d = 3, C = 1.6, \sigma = 0.01, \lambda = \sigma^2/v^2, \alpha = 0.7$; (b) Six-Hump Camel function. Settings: $d = 2, C = 1.6, \sigma = 0.01, \lambda = \sigma^2/v^2, \alpha = 0.7$; (c) Michalewicz function. Settings: $d = 2, C = 1.6, \sigma = 0.1, \lambda = \sigma^2/v^2, \alpha = 0.6$; (d) Function from light sensor data. Settings: $d = 2, C = 1.42, \sigma = 0.01, \lambda = \sigma^2/v^2, \alpha = 0.8$.

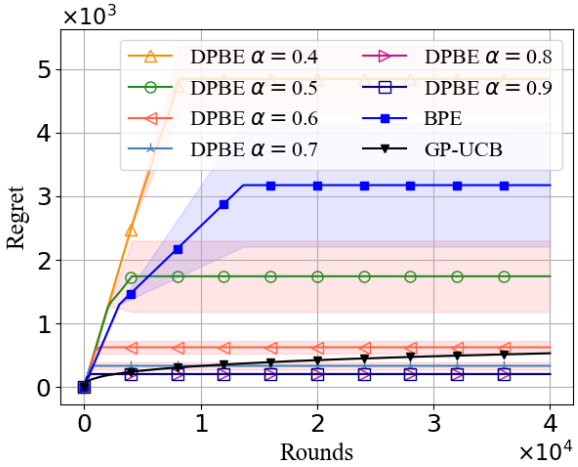


Figure 5.6: Regret performance comparison of `GP-UCB`, `BPE`, and `DPBE`.

Chapter 6

Summary and Future Work

6.1 Summary

This dissertation studied two novel online learning problems that are motivated by autonomous resource management in Next-Generation (NextG) networks under the uncertainty. Specifically, we consider two network resource allocation problems: 1) wireless scheduling with fairness or QoS constraints, and 2) cellular network configuration. Motivated by these problems, the objective of this dissertation is three-fold: i) exploring scheduling with fairness constraints, ii) studying online learning with distributed biased feedback, and iii) investigating the scalability of a design.

Arising from the above two resource allocation problems, this dissertation studies two types of optimization problem and formulates four online (learning) models. First, we study reward maximization under uncertainty with fairness constraints, which is motivated by wireless scheduling with QoS constraints under uncertainty. This dissertation formulates a combinatorial bandits with fairness constraints, and extends it to a nonlinear combinatorial model of submodular maximization with fairness constraints. The second type of problem is global reward maximization under uncertainty with distributed biased feedback, which is motivated by the problem of cellular network configuration for optimizing network-level performance. For the second type of problem, we start with considering linear-parameterized (unknown) reward functions and then extend to studying non-parametric nonlinear global reward func-

tions. Specifically, which are modeled as distributed linear bandits and distributed kernelized bandits, respectively

Among the four main chapters in this dissertation, Chapters 2 and 3 are devoted to the first type of problems, while Chapters 4 and 5 are studying the second type of problems. The following is a brief summary of these four chapters.

- In Chapter 2, we proposed a unified CSMAB-F framework that integrates several critical factors (i.e., combinatorial actions, availability of actions, and fairness) of the system in many real-world applications. In particular, no prior work has studied MAB problems with fairness constraints on a minimum selection fraction for each individual arm. To address the new challenges introduced by modeling these factors, we developed a new LFG algorithm that achieves a provable regret upper bound while effectively providing fairness guarantees.

- Chapter 3 studies submodular maximization with fairness constraints and formulates a multi-round monotone submodular maximization with cardinality and fairness constraints (MMSM-F). To address this new problem, we proposed three carefully designed algorithms (i.e., FairCG1, FairCG2, and FairDG) and presented both theoretical and simulation results to demonstrate the effectiveness of our proposed algorithms.

- Chapter 4, motivated by the cellular network configuration, investigated a new problem of global reward maximization under uncertainty with distributed biased feedback. By assuming a linear parameterized reward function, this chapter modeled the problem as a (differentially private) distributed linear bandits formulation, where the learning agent samples clients and interacts with them by iteratively aggregating such distributed biased feedback in a privacy-preserving fashion. Then, a unified algorithmic learning framework, called DP-DPE, is developed, which can be naturally integrated

with different DP models, and systematically established the regret-communication-privacy tradeoff.

- Chapter 5 extended Chapter 4 to non-linear or even non-convex reward functions and studied the same problem in the distributed kernelized bandits setting. Considering the communication cost for collecting feedback and the computational bottleneck of kernelized bandits, this chapter carefully designed the distributed phase-then-batch-based elimination (`DPBE`) algorithm to address all the new challenges. We showed that `DPBE` achieves a sublinear regret while being scalable in terms of communication efficiency and computation complexity. Finally, similar to DP-DPE in Chapter 4, `DPBE` is flexible to incorporate various differential privacy models to ensure privacy guarantees for participating users.

6.2 Future Work

As this dissertation studied two key problems of resource allocation in wireless networks, there are a number of interesting research directions that I have identified during my research investigation. These research directions are listed below.

Distributed bandits with heterogeneous DP requirements. In the second part of this dissertation, we study distributed bandits with heterogeneous users in the sense of different local reward functions. In practical, heterogeneous users may have heterogeneous DP requirements. One may satisfy this by adding different amount of noise at mentioned in Chapter 4. What if we need to pay for the private data? That is, the learning agent needs to pay to the clients first before being able to collecting users' feedback. In this case, I am also wondering how to learn while saving cost. We may choose the users who do not care much about privacy but users whose private data costs a lost may have to be selected in

order to balance the bias in the feedback.

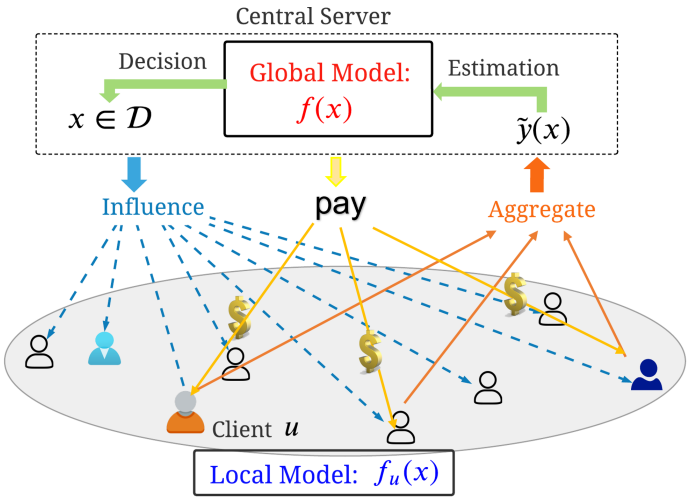


Figure 6.1: Paying for Heterogeneous DP

Unknown submodular functions. In Chapter 3, we study submodular maximization with fairness constraints by assuming access to a value oracle as the problem is NP-hard and remains unexplored in the literature. If the submodular function under consideration is unknown in advance, it is highly interesting to investigate the joint learning and selection problem of multi-round submodular optimization as well.

Regret-communication-(computation)-privacy tradeoff. While we proposed `DP-DPE` for distributed linear bandits setting and `DPBE` for kernelized bandit setting to address the new challenges that arise in the problem setup, it would be worthwhile to explore other batch-based algorithms and investigate whether one can further improve the tradeoff among regret, communication efficiency, computation complexity, and privacy.

(Sub-)optimality analysis. In Chapter 2, 4, and 5, we study different variants of bandits problems and provide regret upper bound in the main results. Note that the lower bounds derived for the standard bandits are also valid lower bounds for the problems we study since our problems can degenerate to the standard cases (e.g., no fairness constraints, or no bias

in the feedback). We can show that our algorithms can achieve the lower bounds in some cases. In general, however, it is an important open problem to close the gap by deriving tighter lower and/or upper bounds that capture the effect of specific factors (e.g., fairness constraints in Chapter 2 or bias due to insufficient participants in Chapter 4 and 5. We leave exploring lower bounds for all the bandits setting in this dissertation as our future work.

Other resource allocation problems in wireless networks. In Chapter 2, we consider the QoS of delivery ratio requirement in wireless scheduling, which is formulated as a fairness constraint. In addition to this temporal fairness criteria, we are also interested in exploring more general fairness criteria as well as other QoS requirements in wireless networks. In addition, while the main problems studied in this dissertation are motivated by two specific network optimization problems in cellular wireless networks, it is also interesting to explore other problems, e.g., power control and congestion control, and build online learning models due to the inherit uncertainty in wireless networks.

Appendices

Appendix A

Proofs for Chapter 2

A.1 Proof of Theorem 2.2

Proof. Consider the LFG algorithm. To prove feasibility optimality, we want to show that for any vector $\mathbf{r}$ strictly inside the maximal feasibility region $\mathcal{C}$, the minimum selection fraction requirements (i.e., Eq. (3.1)) are satisfied. Note that the requirements of (3.1) are satisfied as long as the virtual queue system defined in (2.7) is *mean rate stable* [104, pp. 56-57], i.e., $\lim_{T\to\infty} \frac{\mathbb{E}[\sum_{i=1}^{N} Q_i(T)]}{T} = 0$. In our virtual queue system, mean rate stability is implied by a stronger notion called *strong stability*, i.e., $\limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[\sum_{i=1}^{N} Q_i(t)] < \infty$. Therefore, it is sufficient to show that the virtual queue system is strongly stable for any vector $\mathbf{r}$ strictly inside $\mathcal{C}$.

We proceed the proof using the Lyapunov-drift analysis [104]. Let $\mathbf{Q}(t) = (Q_1(t), \ldots, Q_N(t))$ be the queue-length vector in round t. Consider the following Lyapunov function:

$$L(\mathbf{Q}(t)) \triangleq \frac{1}{2} \sum_{i=1}^{N} Q_i^2(t). \tag{A.1}$$

The drift of the Lyapunov function is given by

$$
\begin{aligned}
&L(\mathbf{Q}(t+1)) - L(\mathbf{Q}(t)) \\
&= \frac{1}{2}\sum_{i=1}^{N} Q_i^2(t+1) - \frac{1}{2}\sum_{i=1}^{N} Q_i^2(t) \\
&\overset{(a)}{\leqslant} \frac{1}{2}\sum_{i=1}^{N} (Q_i(t) + r_i - d_i(t))^2 - \frac{1}{2}\sum_{i=1}^{N} Q_i^2(t) \\
&= \frac{1}{2}\sum_{i=1}^{N} (r_i - d_i(t))^2 + \sum_{i=1}^{N} (r_i - d_i(t))Q_i(t) \\
&\overset{(b)}{\leqslant} \frac{N}{2} + \sum_{i=1}^{N} r_i Q_i(t) - \sum_{i=1}^{N} d_i(t) Q_i(t),
\end{aligned}
\tag{A.2}
$$

where (a) is from the queue-length evolution (2.7) and (b) holds because both r_i and $d_i(t)$

are within $[0,1]$. Taking conditional expectation of both sides of the above gives

$$
\begin{aligned}
&\mathbb{E}[L(\mathbf{Q}(t+1)) - L(\mathbf{Q}(t))|\mathbf{Q}(t)] \\
&\leq \frac{N}{2} + \sum_{i=1}^{N} r_i Q_i(t) - \mathbb{E}\left[\sum_{i=1}^{N} d_i(t) Q_i(t)|\mathbf{Q}(t)\right] \\
&= \frac{N}{2} + \sum_{i=1}^{N} r_i Q_i(t) - \mathbb{E}\left[\sum_{i\in S(t)} d_i(t) Q_i(t)|\mathbf{Q}(t)\right] \\
&= \frac{N}{2} + \sum_{i=1}^{N} r_i Q_i(t) + \mathbb{E}\left[\sum_{i\in S(t)} \eta w_i \bar{\mu}_i(t)|\mathbf{Q}(t)\right] \\
&\quad - \mathbb{E}\left[\sum_{i\in S(t)} (Q_i(t) + \eta w_i \bar{\mu}_i(t))|\mathbf{Q}(t)\right] \\
&\leqslant \frac{N}{2} + \sum_{i=1}^{N} r_i Q_i(t) + \eta m w_{\max} \\
&\quad - \mathbb{E}\left[\sum_{i\in S(t)} (Q_i(t) + \eta w_i \bar{\mu}_i(t))|\mathbf{Q}(t)\right] \\
&= B + \sum_{i=1}^{N} r_i Q_i(t) \\
&- \mathbb{E}\left[\sum_{i\in S(t)} (Q_i(t) + \eta w_i \bar{\mu}_i(t))|\mathbf{Q}(t)\right],
\end{aligned}
\tag{A.3}
$$

where the last inequality holds because $w_i \leqslant w_{\max}$, $\bar{\mu}_i(t) \leqslant 1$, and $|S(t)| \leqslant m$, and $B \triangleq \frac{N}{2} + \eta m w_{\max}$ is a constant.

Recall that $\mathbf{r}$ is strictly inside $\mathcal{C}$. Then, there must exist some $\varepsilon > 0$ such that $\mathbf{r} + \varepsilon\mathbf{1}$ is also strictly inside $\mathcal{C}$, where $\mathbf{1}$ denotes the N-dimensional all-ones vector. By Lemma 2.1, there exists an A-only policy α that can support vector $\mathbf{r} + \varepsilon\mathbf{1}$. That is,

$$
\sum_{Z\in\mathcal{P}(\mathcal{N})} P_{\mathbf{A}}(Z) \sum_{S\in\mathcal{S}(Z):i\in S} q_S^{\alpha}(Z) \geqslant r_i + \varepsilon, \ \forall i \in \mathcal{N}, \tag{A.4}
$$

where $\mathbf{q}^\alpha = [q_S^\alpha(Z), \forall S \in \mathcal{S}(Z), \forall Z \in \mathcal{P}(\mathcal{N})]$ is the group of probability distributions associated with policy α. Recall that in each round t, policy α observes available arms $A(t)$ and chooses a super arm $S^\alpha(t) \in \mathcal{S}(A(t))$ independent of $\mathbf{Q}(t)$. Then, the last term of the right-hand side of (A.3) satisfies

$$\begin{aligned}
&\mathbb{E}\left[\sum_{i\in S(t)} (Q_i(t) + \eta w_i \bar{\mu}_i(t)) | \mathbf{Q}(t)\right] \\
&= \mathbb{E}\left[\mathbb{E}\left[\sum_{i\in S(t)} (Q_i(t) + \eta w_i \bar{\mu}_i(t)) | \mathbf{Q}(t), A(t)\right]\right] \\
&\overset{(a)}{\geqslant} \mathbb{E}\left[\mathbb{E}\left[\sum_{i\in S^\alpha(t)} (Q_i(t) + \eta w_i \bar{\mu}_i(t)) | \mathbf{Q}(t), A(t)\right]\right] \\
&\geqslant \mathbb{E}\left[\mathbb{E}\left[\sum_{i\in S^\alpha(t)} Q_i(t) | \mathbf{Q}(t), A(t)\right]\right] \\
&\overset{(b)}{=} \mathbb{E}\left[\mathbb{E}\left[\sum_{i\in S^\alpha(t)} Q_i(t) | A(t)\right]\right] \\
&\overset{(c)}{=} \mathbb{E}\left[\sum_{S\in\mathcal{S}(A(t))} q_S^\alpha(A(t)) \sum_{i\in S} Q_i(t)\right] \\
&= \sum_{Z\in\mathcal{P}(\mathcal{N})} P_{\mathbf{A}}(Z) \sum_{S\in\mathcal{S}(Z)} q_S^\alpha(Z) \sum_{i\in S} Q_i(t) \\
&= \sum_{i=1}^{N} Q_i(t) \sum_{Z\in\mathcal{P}(\mathcal{N})} P_{\mathbf{A}}(Z) \sum_{S\in\mathcal{S}(Z):i\in S} q_S^\alpha(Z),
\end{aligned} \tag{A.5}$$

where (a) is due to the operations of LFG (specifically, (2.8)), (b) holds because policy α's decision is independent of $\mathbf{Q}(t)$, and (c) is due to the operations of policy α.

Substituting (A.5) into (A.3) and applying (A.4) give

$$\begin{aligned}
&\mathbb{E}[L(\mathbf{Q}(t+1)) - L(\mathbf{Q}(t))|\mathbf{Q}(t)] \\
&\leqslant B + \sum_{i=1}^{N} r_i Q_i(t) \\
&\quad - \sum_{i=1}^{N} Q_i(t) \sum_{Z \in \mathcal{P}(\mathcal{N})} P_{\mathbf{A}}(Z) \sum_{S \in \mathcal{S}(Z): i \in S} q_S^{\alpha}(Z) \\
&\leqslant B + \sum_{i=1}^{N} r_i Q_i(t) - \sum_{i=1}^{N} Q_i(t)(r_i + \varepsilon) \\
&= B - \varepsilon \sum_{i=1}^{N} Q_i(t).
\end{aligned} \tag{A.6}$$

Finally, invoking the Lyapunov Drift Theorem [104, Theorem 4.1] gives (2.10), which completes the proof. □

A.2 Proof of Theorem 2.3

Proof. Consider an optimal A-only policy α^* and its associated probability distributions $\mathbf{q}^* = [q_S^*(Z), \forall S \in \mathcal{S}(Z), \forall Z \in \mathcal{P}(\mathcal{N})]$. Let $S^*(t)$ be the super arm selected by policy α^* in round t. Vector $\mathbf{d}^*(t) = (d_1^*(t), \ldots, d_N^*(t))$ is the corresponding action vector. Due to (2.4), we have

$$\begin{aligned}
R^* &= \sum_{Z \in \mathcal{P}(\mathcal{N})} P_{\mathbf{A}}(Z) \sum_{S \in \mathcal{S}(Z)} q_S^*(Z) \sum_{i \in S} w_i \mu_i \\
&= \mathbb{E}\left[\sum_{i \in S^*(t)} w_i \mu_i \right].
\end{aligned} \tag{A.7}$$

Plugging (A.7) into (2.5), we can rewrite the regret of LFG as

$$\begin{aligned} R_{\text{LFG}}(T) &= R^* - \frac{1}{T}\mathbb{E}\left[\sum_{t=0}^{T-1}\sum_{i\in S(t)} w_i X_i(t)\right] \\ &= \frac{1}{T}\sum_{t=0}^{T-1}\left\{R^* - \mathbb{E}\left[\sum_{i\in S(t)} w_i\mu_i\right]\right\} \\ &= \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\underbrace{\sum_{i\in S^*(t)} w_i\mu_i - \sum_{i\in S(t)} w_i\mu_i}_{\Delta R(t)}\right]. \end{aligned} \tag{A.8}$$

We define the following quantity:

$$\begin{aligned} \Delta R(t) &\triangleq \sum_{i\in S^*(t)} w_i\mu_i - \sum_{i\in S(t)} w_i\mu_i \\ &= \sum_{i=1}^{N} w_i\mu_i d_i^*(t) - \sum_{i=1}^{N} w_i\mu_i d_i(t), \end{aligned} \tag{A.9}$$

which captures the gap between the expected rewards achieved by policy α^* and LFG in round t. Adding $\Delta R(t)$ scaled by η to the drift of the Lyapunov function (i.e., (A.2)) gives the drift-plus-regret:

$$\begin{aligned} &L(\mathbf{Q}(t+1)) - L(\mathbf{Q}(t)) + \eta\Delta R(t) \\ &\leqslant \frac{N}{2} + \sum_{i=1}^{N} r_i Q_i(t) - \sum_{i=1}^{N} d_i(t) Q_i(t) \\ &\quad + \eta\sum_{i=1}^{N} w_i\mu_i d_i^*(t) - \eta\sum_{i=1}^{N} w_i\mu_i d_i(t) \\ &= \frac{N}{2} + \sum_{i=1}^{N}(Q_i(t) + \eta w_i\mu_i)(d_i^*(t) - d_i(t)) \\ &\quad + \sum_{i=1}^{N} Q_i(t)(r_i - d_i^*(t)). \end{aligned} \tag{A.10}$$

We can bound the expected drift-plus-regret as

$$\begin{aligned}
&\mathbb{E}[L(\mathbf{Q}(t+1)) - L(\mathbf{Q}(t)) + \eta \Delta R(t)] \\
&\leqslant \frac{N}{2} + \sum_{i=1}^{N} \mathbb{E}[(Q_i(t) + \eta w_i \mu_i)(d_i^*(t) - d_i(t))] \\
&\quad + \sum_{i=1}^{N} \mathbb{E}[Q_i(t)(r_i - d_i^*(t))] \\
&\leqslant \frac{N}{2} + \mathbb{E}\left[\underbrace{\sum_{i=1}^{N} (Q_i(t) + \eta w_i \mu_i)(d_i^*(t) - d_i(t))}_{C_1(t)} \right],
\end{aligned} \tag{A.11}$$

where the last step follows from $\mathbb{E}[Q_i(t) d_i^*(t)] = \mathbb{E}[Q_i(t)]\mathbb{E}[d_i^*(t)]$ (due to the decision of policy α^* being independent of the queue length $Q_i(t)$) and $\mathbb{E}[d_i^*(t)] \geqslant r_i$ (because policy α^* is stationary and feasible). Define $C_1(t) \triangleq \sum_{i=1}^{N} (Q_i(t) + \eta w_i \mu_i)(d_i^*(t) - d_i(t))$. Summing (A.11) for all $t \in \{0, \ldots, T-1\}$, using the trick of telescoping sum, and dividing both sides of the inequality by $T\eta$, we obtain

$$\begin{aligned}
&\frac{1}{T\eta} \mathbb{E}[L(\mathbf{Q}(T)) - L(\mathbf{Q}(0))] + \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[\Delta R(t)] \\
&\leqslant \frac{N}{2\eta} + \frac{1}{T\eta} \sum_{t=0}^{T-1} \mathbb{E}[C_1(t)].
\end{aligned} \tag{A.12}$$

Since $L(\mathbf{Q}(T)) \geqslant 0$ and $L(\mathbf{Q}(0)) = 0$, we have

$$\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[\Delta R(t)] \leqslant \frac{N}{2\eta} + \frac{1}{T\eta} \sum_{t=0}^{T-1} \mathbb{E}\left[C_1(t)\right]. \tag{A.13}$$

In Appendix A.3, we will show the following bound:

$$\begin{aligned}&\frac{1}{T\eta}\sum_{t=0}^{T-1}\mathbb{E}[C_1(t)]\\&\leqslant \frac{w_{\max}}{T}\left(2\sqrt{6mNT\log T}+(1+\frac{5\pi^2}{12})N\right).\end{aligned} \tag{A.14}$$

Finally, plugging (A.14) into (A.13) and combining it with (A.8) yield (2.11). This completes the proof of Theorem 2.3. □

A.3 Bounding $C_1(t)$

In this section, we want to show (A.14).

Consider a policy π', which, in each round t, chooses a super arm $S'(t)$ in the following manner:

$$S'(t) \in \operatorname*{argmax}_{S\in\mathcal{S}(A(t))} \sum_{i\in S}(Q_i(t)+\eta w_i\mu_i(t)). \tag{A.15}$$

Recall that in each round t, the LFG algorithm chooses a super arm $S(t)$ according to (2.8). Therefore, we have

$$\sum_{i\in S(t)}(Q_i(t)+\eta w_i\bar{\mu}_i(t)) \geqslant \sum_{i\in S'(t)}(Q_i(t)+\eta w_i\bar{\mu}_i(t)). \tag{A.16}$$

Next, we derive an upper bound on $C_1(t)$:

$$
\begin{aligned}
C_1(t) &= \sum_{i=1}^{N} (Q_i(t) + \eta w_i \mu_i)(d_i^*(t) - d_i(t)) \\
&= \sum_{i \in S^*(t)} (Q_i(t) + \eta w_i \mu_i) - \sum_{i \in S(t)} (Q_i(t) + \eta w_i \mu_i) \\
&\overset{(a)}{\leqslant} \sum_{i \in S'(t)} (Q_i(t) + \eta w_i \mu_i) - \sum_{i \in S(t)} (Q_i(t) + \eta w_i \mu_i) \\
&\overset{(b)}{\leqslant} \sum_{i \in S'(t)} (Q_i(t) + \eta w_i \mu_i) - \sum_{i \in S(t)} (Q_i(t) + \eta w_i \mu_i) \\
&\quad + \sum_{i \in S(t)} (Q_i(t) + \eta w_i \bar{\mu}_i(t)) \\
&\quad - \sum_{i \in S'(t)} (Q_i(t) + \eta w_i \bar{\mu}_i(t)) \\
&= \eta \left(\underbrace{\sum_{i \in S(t)} w_i(\bar{\mu}_i(t) - \mu_i)}_{C_2(t)} + \underbrace{\sum_{i \in S'(t)} w_i(\mu_i - \bar{\mu}_i(t))}_{C_3(t)} \right),
\end{aligned}
\tag{A.17}
$$

where (a) is from (A.15) and (b) is from (A.16). Define $C_2(t) \triangleq \sum_{i \in S(t)} w_i(\bar{\mu}_i(t) - \mu_i)$ and $C_3(t) \triangleq \sum_{i \in S'(t)} w_i(\mu_i - \bar{\mu}_i(t))$. In Appendices A.4 and **??**, we will show the following two bounds, respectively:

$$
\sum_{t=0}^{T-1} \mathbb{E}[C_2(t)] \leqslant w_{\max} \left(2\sqrt{6mNT \log T} + (1 + \frac{\pi^2}{4})N \right) \tag{A.18}
$$

$$
\sum_{t=0}^{T-1} \mathbb{E}[C_3(t)] \leqslant \frac{\pi^2}{6} w_{\max} N. \tag{A.19}
$$

Finally, summing (A.17) for all $t \in \{0, \ldots, T-1\}$, dividing both sides of the resulting inequality by $T\eta$, and plugging (A.18) and (A.19) into it yield (A.14).

Remark: The bound in (A.18) consists of two terms: the first term is of the order $O(\sqrt{T \log T})$,

which corresponds to the notion of regret in typical MAB problems and is attributed to the cost that needs to be paid in the learning/exploration process; the second term is a constant, which is from applying the Chernoff-Hoeffding bound (see, e.g., [8]) to a "bad" event $\{\hat{\mu}_i(t-1) - \mu_i > \sqrt{\frac{3\log t}{2h_i(t-1)}}\}$. Similarly, the bound in (A.19) is from applying the Chernoff-Hoeffding bound to another "bad" event $\{\hat{\mu}_i(t-1) - \mu_i < -\sqrt{\frac{3\log t}{2h_i(t-1)}}\}$.

A.4 Bounding $C_2(t)$

In this section, we want to show (A.18).

Consider an arbitrary arm i in $\mathcal{N}$ and an arbitrary round $t = 0, 1, \ldots, T-1$. Let t_a^i be the round in which arm i is played for the a-th time. Recall that $h_i(t)$ is the number of times arm i has been played by the end of round t. Clearly, we have $d_i(t_a^i) = 1$, $h_i(t_a^i) = a$, and $h_i(t_a^i - 1) = a - 1$ for all $a \in \{1, 2, \ldots, h_i(T-1)\}$. In addition, we also have

$$0 \leqslant t_1^i < t_2^i < \cdots < t_{h_i(T-1)}^i < T. \tag{A.20}$$

Define the following event:

$$U_i(t) \triangleq \{\bar{\mu}_i(t) < \mu_i\}. \tag{A.21}$$

Let E^c be the complement of an event E, and let $\mathbb{1}_{\{\cdot\}}$ denote the indicator function. We

bound the expectation of $C_2(t)$ as

$$\begin{aligned}
\mathbb{E}[C_2(t)] &= \mathbb{E}\left[\sum_{i=1}^{N} w_i(\bar{\mu}_i(t)-\mu_i)d_i(t)\right] \\
&= \mathbb{E}\left[\sum_{i=1}^{N} w_i(\bar{\mu}_i(t)-\mu_i)d_i(t)\mathbb{1}_{\{U_i(t)\}}\right] \\
&\quad + \mathbb{E}\left[\sum_{i=1}^{N} w_i(\bar{\mu}_i(t)-\mu_i)d_i(t)\mathbb{1}_{\{U_i^c(t)\}}\right] \\
&\overset{(a)}{\leqslant} \mathbb{E}\left[\sum_{i=1}^{N} w_i(\bar{\mu}_i(t)-\mu_i)d_i(t)\mathbb{1}_{\{U_i^c(t)\}}\right] \\
&\overset{(b)}{\leqslant} w_{\max}\sum_{i=1}^{N}\mathbb{E}[\underbrace{(\bar{\mu}_i(t)-\mu_i)d_i(t)\mathbb{1}_{\{U_i^c(t)\}}}_{J_1(t)}],
\end{aligned} \tag{A.22}$$

where (a) is due to $\bar{\mu}_i(t) < \mu_i$ when event $U_i(t)$ happens and (b) is due to $\bar{\mu}_i(t) \geq \mu_i$ when event $U_i^c(t)$ happens. Define $J_1(t) \triangleq (\bar{\mu}_i(t)-\mu_i)d_i(t)\mathbb{1}_{\{U_i^c(t)\}}$. Also, define another event:

$$F_i(t) \triangleq \left\{\hat{\mu}_i(t-1)-\mu_i \leqslant \sqrt{\frac{3\log t}{2h_i(t-1)}}\right\}. \tag{A.23}$$

Then, summing $J_1(t)$ for all $t \in \{0, \ldots, T-1\}$ gives

$$\begin{aligned}
&\sum_{t=0}^{T-1} J_1(t) \\
&\overset{(a)}{=} \sum_{a=1}^{h_i(T-1)} (\bar{\mu}_i(t_a^i) - \mu_i)\mathbb{1}_{\{U_i^c(t_a^i)\}} \\
&\overset{(b)}{\leqslant} 1 + \sum_{a=2}^{h_i(T-1)} (\bar{\mu}_i(t_a^i) - \mu_i)\mathbb{1}_{\{U_i^c(t_a^i)\}} \\
&= 1 + \sum_{a=2}^{h_i(T-1)} (\bar{\mu}_i(t_a^i) - \mu_i)\mathbb{1}_{\{U_i^c(t_a^i)\}}\big(\mathbb{1}_{\{F_i(t_a^i)\}} + \mathbb{1}_{\{F_i^c(t_a^i)\}}\big) \\
&\overset{(c)}{\leqslant} 1 + \sum_{a=2}^{h_i(T-1)} \big(\underbrace{(\bar{\mu}_i(t_a^i) - \mu_i)\mathbb{1}_{\{U_i^c(t_a^i)\cap F_i(t_a^i)\}}}_{J_2(t_a^i)} + \mathbb{1}_{\{F_i^c(t_a^i)\}}\big),
\end{aligned} \tag{A.24}$$

where (a) is due to $d_i(t_a^i) = 1$ for all $a \in \{1, 2, \ldots, h_i(T-1)\}$ and $d_i(t) = 0$ for all other t, (b) is due to $\bar{\mu}_i(t_1^i) - \mu_i \leq 1$, and (c) is due to $(\bar{\mu}_i(t_a^i) - \mu_i)\mathbb{1}_{\{U_i^c(t_a^i)\}} \leqslant 1$. We define $J_2(t_a^i) \triangleq (\bar{\mu}_i(t_a^i) - \mu_i)\mathbb{1}_{\{U_i^c(t_a^i)\cap F_i(t_a^i)\}}$ and want to bound both $\sum_{a=2}^{h_i(T-1)} \mathbb{E}[J_2(t_a^i)]$ and $\sum_{a=2}^{h_i(T-1)} \mathbb{E}[\mathbb{1}_{\{F_i^c(t_a^i)\}}]$.

First, we want to bound $\sum_{a=2}^{h_i(T-1)} \mathbb{E}[J_2(t_a^i)]$. Consider t_a^i for all $a \in \{2, \ldots, h_i(T-1)\}$. Suppose event $F_i(t_a^i)$ happens. Then, we have

$$\hat{\mu}_i(t_a^i - 1) - \mu_i \leqslant \sqrt{\frac{3 \log t_a^i}{2h_i(t_a^i - 1)}}. \tag{A.25}$$

From (2.6), we also have

$$\bar{\mu}_i(t_a^i) \leqslant \hat{\mu}_i(t_a^i - 1) + \sqrt{\frac{3 \log t_a^i}{2h_i(t_a^i - 1)}}. \tag{A.26}$$

Combining (A.25) and (A.26) gives

$$\bar{\mu}_i(t_a^i) - \mu_i \leqslant 2\sqrt{\frac{3\log t_a^i}{2h_i(t_a^i - 1)}}, \tag{A.27}$$

which implies that for all $a \in \{2, \ldots, h_i(T-1)\}$, we have

$$\begin{aligned} J_2(t_a^i) &= (\bar{\mu}_i(t_a^i) - \mu_i)\mathbb{1}_{\{U_i^c(t_a^i) \cap F_i(t_a^i)\}} \\ &\leqslant 2\sqrt{\frac{3\log t_a^i}{2h_i(t_a^i - 1)}}. \end{aligned} \tag{A.28}$$

Then, summing $J_2(t_a^i)$ for all $a \in \{2, \ldots, h_i(T-1)\}$ gives

$$\begin{aligned} \sum_{a=2}^{h_i(T-1)} J_2(t_a^i) &\overset{(a)}{\leqslant} \sum_{a=2}^{h_i(T-1)} 2\sqrt{\frac{3\log t_a^i}{2h_i(t_a^i - 1)}} \\ &\overset{(b)}{\leqslant} \sqrt{6\log T} \sum_{a=2}^{h_i(T-1)} \frac{1}{\sqrt{a-1}} \\ &\overset{(c)}{\leqslant} \sqrt{6\log T}\left(1 + \int_1^{h_i(T-1)} \frac{1}{\sqrt{x}}\, dx\right) \\ &\leqslant 2\sqrt{6h_i(T-1)\log T}, \end{aligned} \tag{A.29}$$

where (a) is from (A.28), (b) is due to $t_a^i \leqslant T$ (from (A.20)) and $h_i(t_a^i - 1) = a - 1$ for all $a \in \{2, \ldots, h_i(T-1)\}$, and (c) is due to a basic relationship between the considered summation and integral. Therefore, we have

$$\sum_{a=2}^{h_i(T-1)} \mathbb{E}[J_2(t_a^i)] \leqslant 2\sqrt{6\log T}\,\mathbb{E}[\sqrt{h_i(T-1)}]. \tag{A.30}$$

Next, we want to bound $\sum_{a=2}^{h_i(T-1)} \mathbb{E}[\mathbb{1}_{\{F_i^c(t_a^i)\}}]$. According to the definition of t_a^i, we have $h_i(t_a^i - 1) = a - 1$, and thus $\hat{\mu}_i(t_a^i - 1)$ is the sample mean of $(a-1)$ *i.i.d.* random variables

$X_i(t_1^i), \cdots, X_i(t_{a-1}^i)$ with mean μ_i. Further, we know t_a^i must satisfy $a-1 \leqslant t_a^i \leqslant T-1$. Hence,

$$\begin{aligned} F_i^c(t_a^i) &= \left\{ \hat{\mu}_i(t_a^i - 1) - \mu_i > \sqrt{\frac{3\log t_a^i}{2h_i(t_a^i - 1)}} \right\} \\ &\subseteq \cup_{n=a-1}^{T-1} \left\{ \hat{\mu}_i(n-1) - \mu_i > \sqrt{\frac{3\log n}{2(a-1)}} \right\}. \end{aligned} \tag{A.31}$$

By applying the union bound and the Chernoff-Hoeffding bound (see, e.g., [8]), we have

$$\begin{aligned} &\mathbb{E}[\mathbb{1}_{\{F_i^c(t_a^i)\}}] = \mathbb{P}\left\{F_i^c(t_a^i)\right\} \\ &\leqslant \sum_{\tau=a-1}^{T-1} \mathbb{P}\left\{ \hat{\mu}_i(\tau-1) - \mu_i > \sqrt{\frac{3\log \tau}{2(a-1)}} \right\} \\ &\leqslant \sum_{\tau=a-1}^{T-1} \frac{1}{\tau^3} \leqslant \frac{1}{(a-1)^3} + \int_{a-1}^{\infty} \frac{1}{x^3} dx \leqslant \frac{3}{2(a-1)^2}. \end{aligned}$$

Hence, we derive

$$\sum_{a=2}^{h_i(T-1)} \mathbb{E}[\mathbb{1}_{\{F_i^c(t_a^i)\}}] \leqslant \sum_{a=2}^{h_i(T-1)} \frac{3}{2(a-1)^2} \leqslant \sum_{a=1}^{\infty} \frac{3}{2a^2} = \frac{\pi^2}{4}. \tag{A.32}$$

Taking expectation of both sides of (A.24) and plugging (A.30) and (A.32) into it yield

$$\sum_{t=0}^{T-1} \mathbb{E}[J_1(t)] \leqslant 2\sqrt{6\log T}\,\mathbb{E}[\sqrt{h_i(T-1)}] + 1 + \frac{\pi^2}{4}. \tag{A.33}$$

Finally, summing (A.22) for all $t \in \{0, \ldots, T-1\}$ and plugging (A.33) into it yield (A.18):

$$\begin{aligned}
&\sum_{t=0}^{T-1} \mathbb{E}[C_2(t)] \\
&\leqslant w_{\max} \sum_{i=1}^{N} \left(2\sqrt{6 \log T} \mathbb{E}[\sqrt{h_i(T-1)}] + 1 + \frac{\pi^2}{4} \right) \\
&\leqslant w_{\max} \left(2\sqrt{6mNT \log T} + (1 + \frac{\pi^2}{4})N \right),
\end{aligned} \tag{A.34}$$

where the last step follows from $\frac{1}{N}\sum_{i=1}^{N} \sqrt{h_i(T-1)} \leqslant \sqrt{\frac{1}{N}\sum_{i=1}^{N} h_i(T-1)}$ (due to Jensen's inequality) and $\sum_{i=1}^{N} h_i(T-1) \leqslant Tm$ (due to the fact that at most m arms can be selected in each round).

Appendix B

Appendix for Chapter 3

B.1 Applications

In this section, we discuss three additional examples of real-world applications in detail to better motivate the proposed MMSM-CF problem. These examples include sensor scheduling in wireless sensor networks [63, 71, 119, 132], task assignment in crowdsourcing platforms [59, 130, 140], and data subset selection in machine learning [10, 79, 138].

Sensor Scheduling in Wireless Sensor Networks: We consider sensor scheduling in wireless sensor networks, where a set of inexpensive sensors are deployed to sense the environment state process. After the measurements from the sensors are transmitted to a sink node, they will be fused to estimate the environment state process. Obviously, more measurements (collected from distinct sensors) will result in a more accurate estimation of the environment state. However, only a subset of sensors could transmit their measurements simultaneously (e.g., due to wireless interference) [63, 119]. Let $f(S)$ be the aggregate sensing quality of the measurements collected from the scheduled sensors S. Due to the spatial correlation among the sensor measurements, the aggregate sensing quality $f(S)$ often exhibits a diminishing returns property [63, 119]. Moreover, one usually needs to repeatedly collect the measurements (i.e., in multiple rounds) to continuously monitor the environment and aims to maximize the overall sensing quality over time. In addition, to obtain a holistic view of the environment that is being monitored, one often does not want to miss out on too much data from any

single sensor. This imposes a minimum delivery ratio requirement of each sensor, which can be modeled as the fairness requirement in our model. Hence, the goal is to schedule a sequence of subsets of sensors so as to maximize the overall sensing quality over time while guaranteeing a minimum delivery ratio of each sensor.

Task Assignment in Crowdsourcing Platforms: Crowdsourcing offers an efficient method for task distribution and completion. Consider a spatial crowdsourcing application (e.g., traffic speed estimation) [130, 140]. When a spatial task arrives at the crowdsourcing platform, it will be assigned to a group of workers on the platform (e.g., no more than k workers due to the budget limit). A certain amount of utility is generated after this particular task is completed. The utility could represent the informative data gathered for the crowdsourced sensing task. Due to the similarity of responses from different workers [59, 130, 140], such as the possible overlapping sensing data from different workers [141], the utility could be described as a submodular function with regard to the set of assigned workers [140]. The participation of more workers usually leads to more informative data and thus a larger utility (i.e., monotone). As in [130], we assume that the sequential tasks are of the same type and that all the workers are qualified to perform the tasks. This can be captured by the multi-round nature of our model. The goal here is to maximize the time-average utility by determining an optimal worker assignment for multiple tasks. In addition, the platform has to take fairness towards workers into account through a minimum assignment ratio guarantee for each worker. This helps maintain a healthy and sustained platform with improved satisfaction among the workers and thus encourage more participation.

Data Subset Selection in Machine Learning: The data subset selection problem in machine learning has been extensively studied in the literature [10, 79, 138]. This is motivated by both limited computational resources and redundant information in a massive amount of data. For training, one prefers to select a subset of data sources that is informative or

representative of the entire dataset as modeled by the corresponding objective function. It has been shown that some highly relevant objective functions (used to measure the informativeness or the representativeness, or the combination of the two) are submodular with regard to the selected data sources because of a diminishing returns property it exhibits. Consider a multi-round training process. Let the total utility be a simple additive sum of these objective values corresponding to the sequentially selected data subsets. The goal here is to maximize the total utility. Moreover, in order to ensure enough data for post-training data analysis, a minimum selection fraction requirement for each data source must be taken into consideration. We should select not only the most informative or representative data sources but also those less informative or representative ones for a certain amount of times. This can be naturally modeled as a fairness requirement using the MMSM-CF framework. In this case, our goal is to sequentially select a subset of data sources that maximize the total utility while guaranteeing a minimum selection fraction for each data source.

B.2 Proof of Theorem 3.3 and Theorem 3.5

Proof. First, for any feasible $\mathbf{r}$, P_f is non-empty. By the end of Step 1 (i.e., $\tau = 1$) of the fair continuous greedy algorithms, we have

i) FairCG1

$$\begin{aligned}\mathbf{y}(1) &= \int_0^1 \frac{d\mathbf{y}(\tau)}{d\tau} d\tau + \mathbf{y}(0) \\ &= \int_0^1 \mathbf{x}(\tau) d\tau + \mathbf{0} = \int_0^1 \mathbf{x}(\tau) d\tau,\end{aligned} \tag{B.1}$$

ii) FairCG2

$$\begin{aligned}\mathbf{y}(1) &= \int_0^1 \frac{d\mathbf{y}(\tau)}{d\tau} d\tau + \mathbf{y}(0) \\ &= \int_0^1 (\mathbf{x}(\tau) - \mathbf{r}) d\tau + \mathbf{r} = \int_0^1 \mathbf{x}(\tau) d\tau,\end{aligned} \tag{B.2}$$

i.e., $\mathbf{y}(1)$ is a convex linear combination of $\mathbf{x}(\tau)$'s, and thus, $\mathbf{y}(1)$ must be in P_f since $\mathbf{x}(\tau) \in P_f$ and P_f is a convex set. Besides, it is not difficult to show $\mathbf{y}(1)^\mathsf{T}\mathbf{1} = k$ since every $\mathbf{x}(\tau)$ is a vertex of the convex body P_f satisfying $\mathbf{x}^\mathsf{T}\mathbf{1} = k$. Therefore, by the end of this step, we derive a fractional solution $\mathbf{y}(1)$ that satisfies $\mathbf{r} \leqslant \mathbf{y}(1) \leqslant \mathbf{1}$ and $\mathbf{y}(1)^\mathsf{T}\mathbf{1} = k$.

Then, in each round t, the selected set S_t is derived by performing randomized dependent rounding, i.e., DEPROUNDING, on $\mathbf{y}(1)$. Note that $|S_t| = k$ because of the loop invariant $\mathbf{y}^\mathsf{T}\mathbf{1} = k$ in DEPROUNDING. Let a random variable $Y_u(t) \in \{0, 1\}$ represent whether element u is selected or not in round t, i.e., $Y_u(t) = \mathbb{I}_{\{u \in S_t\}}$. We have $\mathbb{P}\{Y_u(t) = 1\} = y_u(1)$ and $\sum_{u \in \mathcal{N}} Y_u(t) = \sum_{u \in \mathcal{N}} y_u(1) = k$ according to the *marginal distribution* property and the *degree-preservation* property of dependent rounding scheme respectively in [58]. Eventually, for the stationary randomized algorithm, we have $\liminf_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}\left[\mathbb{I}_{\{u \in S_t\}}\right] = \mathbb{E}\left[\mathbb{I}_{\{u \in S_t\}}\right] = \mathbb{P}\{Y_u(t) = 1\} = y_u(1) \geqslant r_u$ for every element u. The fairness requirement in Eq. (3.1) is satisfied.

Consider $T > 0$ rounds. Besides, for every element u, $Y_u(1), Y_u(2), \ldots, Y_u(T)$ are *i.i.d.* Bernuoulli variables with mean $y_u(1)$. According to the Hoeffding Inequality [65], the empirical mean of these variables satisfies, for any $\delta > 0$,

$$\mathbb{P}\left\{\frac{1}{T}\sum_{t=1}^{T} Y_u(t) - y_u(1) \leqslant -\delta\right\} \leqslant e^{-2T\delta^2}, \tag{B.3}$$

which further indicates the inequality in Eq. (3.8) since $Y_u(t) = \mathbb{I}_{u \in S_t}$ and $y_u(1) \geqslant r_u$ for each element u. $\square$

B.3 Proof of Theorem 3.4

Before proving Theorem 3.4, we first show an important inequality in the following Lemma B.1.

Lemma B.1. *Consider any time point τ in Step 1 of FairCG. Combining the definitions of $\mathbf{w}(\tau)$ and $\mathbf{x}(\tau)$, we have the following inequality,*

$$\mathbf{x}(\tau)^{\mathsf{T}}\mathbf{w}(\tau) \geqslant U_{\text{opt}} - F(\mathbf{y}(\tau)).$$

The proof for Lemma B.1 is shown in Appendix B.4.

Proof of Theorem 3.4. To derive the result in Theorem 3.4, we first show that the fractional vector $\mathbf{y}(1)$ satisfies $F(\mathbf{y}(1)) \geqslant (1-1/e)U_{\text{opt}}$ (①) and then prove $\mathbb{E}[f(S_t)] \geqslant F(\mathbf{y}(1))$ in each round $t \in \{1, 2, \cdots\}$ (②). Combining the conditions ① and ②, we derive the result in Eq. (3.9).

First, we show ①: $F(\mathbf{y}(1)) \geqslant (1-1/e)U_{\text{opt}}$.

The error due to discretization could be made (polynomially) small [136], and thus, we here only give an analysis for the continuous version. Starting with $\mathbf{y}(0) = \mathbf{0}$ and $F(\mathbf{y}(0)) = F(\mathbf{0}) = 0$, we want to see how much $F(\cdot)$ increases during each discretized time interval

$[\tau, \tau + d\tau)$ in the `while` loop (Lines 2-9). Applying the chain rule yields:

$$\begin{aligned}
\frac{dF(\mathbf{y}(\tau))}{d\tau} &= \sum_{u \in \mathcal{N}} \left(\frac{dy_u(\tau)}{d\tau} \cdot \frac{\partial F(\mathbf{y})}{\partial y_u} \bigg|_{\mathbf{y}=\mathbf{y}(\tau)} \right) \\
&= \sum_{u \in \mathcal{N}} \left(x_u(\tau) \cdot \frac{\partial F(\mathbf{y})}{\partial y_u} \bigg|_{\mathbf{y}=\mathbf{y}(\tau)} \right) \\
&= \sum_{u \in \mathcal{N}} \left(x_u(\tau) \cdot \frac{F(\mathbf{y}(\tau) \vee \mathbf{1}_u) - F(\mathbf{y}(\tau))}{1 - y_u(\tau)} \right) \\
&\geqslant \sum_{u \in \mathcal{N}} x_u(\tau) \cdot w_u(\tau) \\
&= \mathbf{x}(\tau)^{\mathsf{T}} \mathbf{w}(\tau).
\end{aligned} \tag{B.4}$$

Combining the result in Lemma B.1, we have the differential inequality with respect to the function of $F(\mathbf{y}(\tau))$:

$$\frac{dF(\mathbf{y}(\tau))}{d\tau} \geqslant U_{\text{opt}} - F(\mathbf{y}(\tau)).$$

Solving the above differential inequality with the initial condition $F(\mathbf{y}(0)) = 0$ under FairCG1, we have the fractional vector $\mathbf{y}(1)$ satisfying

$$F(\mathbf{y}(1)) \geqslant (1 - 1/e) U_{\text{opt}}. \tag{B.5}$$

Then, we show $\mathbb{E}[f(S_t)] \geqslant F(y(1))$ by employing the property of dependent rounding and convexity of the multilinear extension $F(\cdot)$ in any direction $\mathbf{d} = \mathbf{1}_u - \mathbf{1}_v$ for any pair of distinct elements $u, v \in \mathcal{N}$. Here, the expectation is taken over the randomness of selecting set with FairCG.

The randomized dependent rounding process proceeds as follows. It starts with an n-dimensional fractional vector $\mathbf{y}$ and rounds at least one floating element $y_u \in (0, 1)$ in each iteration of the `While` loop. Hence, the `while` loop takes at most n iterations.

Let $\mathbf{z}_m$ be the random variable denoting the value of $\mathbf{y}$ at the beginning of iteration m, and let n_0 be the last iteration after which all the elements of $\mathbf{y}$ are integers ($n_0 \leqslant n$). We have $\mathbf{z}_1 = \mathbf{y}(1)$ and the returned set $S = \{u \in \mathcal{N} : (Z_{n_0+1})_u = 1)\}$. In the following, we will first show that

$$\forall m, \mathbb{E}[F(\mathbf{z}_{m+1})] \geqslant \mathbb{E}[F(\mathbf{z}_m)], \tag{B.6}$$

where the expectation is taking over the randomness of the updating step in Line 16 in Algorithm 2. Then, we have

$$\mathbb{E}[f(S)] = \mathbb{E}[F(\mathbf{z}_{n_0+1})] \geqslant \mathbb{E}[F(\mathbf{z}_1)] = F(\mathbf{y}(1)).$$

We now prove the inequality Eq. (B.6) for a fixed iteration m. Suppose that elements u and v with $y_u, y_v \in (0,1)$ are the two elements found in current iteration (Line 14 of Algorithm 2. Let a_m and b_m be the values of a and b respectively in iteration m, and $\mathbf{d}_m = \mathbf{1}_u - \mathbf{1}_v$. Then, we have $\mathbf{z}_{m+1}$ equal to $\mathbf{z}_m + a_m\mathbf{d}_m$ with probability $\frac{b_m}{a_m+b_m}$ and equal to $\mathbf{z}_m - b_m\mathbf{d}_m$ with probability $\frac{a_m}{a_m+b_m}$, which indicates the conditional expectation given $F(\mathbf{z}_m) = h$,

$$\begin{aligned} &\mathbb{E}[F(\mathbf{z}_{m+1})|F(\mathbf{z}_m) = h] \\ =&\frac{b_m}{a_m+b_m}F(\mathbf{z}_m + a_m\mathbf{d}_m) + \frac{a_m}{a_m+b_m}F(\mathbf{z}_m - b_m\mathbf{d}_m). \end{aligned} \tag{B.7}$$

Let $g_{\mathbf{z}}(\xi) \triangleq F(\mathbf{z}_m + \xi\mathbf{d}_m)$. Now $g_{\mathbf{z}}(\xi)$ is convex in ξ due to the convexity of $F(\cdot)$ in the direction $\mathbf{d}_m = \mathbf{1}_u - \mathbf{1}_v$. It indicates: $\frac{b_m}{a_m+b_m}g_{\mathbf{z}}(a_m) + \frac{a_m}{a_m+b_m}g_{\mathbf{z}}(-b_m) \geqslant g_{\mathbf{z}}(0)$, i.e., $\frac{b_m}{a_m+b_m}F(\mathbf{z}_m + a_m\mathbf{d}_m) + \frac{a_m}{a_m+b_m}F(\mathbf{z}_m - b_m\mathbf{d}_m) \geqslant F(\mathbf{z}_m)$. Hence, we have

$$\mathbb{E}[F(\mathbf{z}_{m+1})|F(\mathbf{z}_m) = h] \geqslant F(\mathbf{z}_m) = h.$$

Let H be the set of all possible values of h. We have the following inequalities:

$$\begin{aligned}
&\mathbb{E}[F(\mathbf{z}_{m+1})] \\
=&\sum_{h\in H}\mathbb{E}[F(\mathbf{z}_{m+1})|F(\mathbf{z}_m)=h]\cdot\mathbb{P}\{F(\mathbf{z}_m)=h\} \\
\geqslant&\sum_{h\in H} h\cdot\mathbb{P}\{F(\mathbf{z}_m)=h\} \\
=&\mathbb{E}[F(\mathbf{z}_m)],
\end{aligned} \tag{B.8}$$

which is exactly Eq. (B.6). This completes our proof for $\mathbb{E}[f(S)] \geqslant F(\mathbf{y}(1))$ and then the results in Eq. (3.10) by combining the result in Eq. (B.11). □

B.4 Proof of Lemma B.1

Proof. Let $\mathbf{1}_S$ represent the characteristic vector of subset S and $\mathbf{q}^* = [q_S^*]_{S\in\mathcal{N}_k}$ denote an optimal solution to Problem (3.5). Define an n-dimensional vector $\mathbf{y}^* \triangleq \sum_{S\in\mathcal{N}_k} q_S^*\mathbf{1}_S$ with the coordinate corresponding to u being $y_u^* = \sum_{S\in\mathcal{N}_k:u\in S} q_S^*$ for each u in $\mathcal{N}$. Then, we have $\mathbf{r} \leqslant \mathbf{y}^* \leqslant \mathbf{1}$ (constrains in Problem (3.5)) and $\mathbf{y}^{*\mathsf{T}}\mathbf{1} \leqslant k$ since any set $S \in \mathcal{N}_k$ satisfies $\mathbf{1}_S^\mathsf{T}\mathbf{1} \leqslant k$, which implies $\mathbf{y}^* \in P_f$. Based on the definition of $\mathbf{x}(\tau)$, we have

$$\begin{aligned}
&\mathbf{x}(\tau)^\mathsf{T}\mathbf{w}(\tau) \geqslant \mathbf{y}^{*\mathsf{T}}\mathbf{w}(\tau) \\
=&\sum_{u\in\mathcal{N}} y_u^*\left(F(\mathbf{y}(\tau)\vee\mathbf{1}_u)-F(\mathbf{y}(\tau))\right) \\
=&\sum_{u\in\mathcal{N}}\sum_{S\in\mathcal{N}_k:u\in S} q_S^*\left(F(\mathbf{y}(\tau)\vee\mathbf{1}_u)-F(\mathbf{y}(\tau))\right) \\
=&\sum_{S\in\mathcal{N}_k} q_S^*\sum_{u\in S}\left(F(\mathbf{y}(\tau)\vee\mathbf{1}_u)-F(\mathbf{y}(\tau))\right).
\end{aligned}$$

Recall that the multilinear extension $F(\mathbf{y})$ represents the expected value of the submodular function $f(R(\mathbf{y}))$ where $R(\mathbf{y})$ is a random set with each element u being independently selected with probability y_u. To distinguish from other randomness, we denote the multilinear extension as $F(\mathbf{y}) = \mathbb{E}_{R\sim\mathbf{y}}[f(R(\mathbf{y}))]$. We have

$$\begin{aligned}
&\sum_{S\in\mathcal{N}_k} q_S^* \sum_{u\in S} (F(\mathbf{y}(\tau)\vee \mathbf{1}_u) - F(\mathbf{y}(\tau))) \\
=&\sum_{S\in\mathcal{N}_k} q_S^* \sum_{u\in S} \mathbb{E}_{R\sim\mathbf{y}}[f(R(\mathbf{y}(\tau)\vee \mathbf{1}_u)) - f(R(\mathbf{y}(\tau)))] \\
\overset{(a)}{\geqslant}&\sum_{S\in\mathcal{N}_k} q_S^* \mathbb{E}_{R\sim\mathbf{y}}[f(R(\mathbf{y}(\tau)\vee \mathbf{1}_S)) - f(R(\mathbf{y}(\tau)))] \\
\overset{(b)}{\geqslant}&\sum_{S\in\mathcal{N}_k} q_S^* \mathbb{E}_{R\sim\mathbf{y}}[f(R(\mathbf{1}_S)) - f(R(\mathbf{y}(\tau)))] \\
=&\sum_{S\in\mathcal{N}_k} q_S^* (f(S) - F(\mathbf{y}(\tau))) \\
=&\sum_{S\in\mathcal{N}_k} q_S^* f(S) - \sum_{S\in\mathcal{N}_k} q_S^* F(\mathbf{y}(\tau))) \\
=&U_{\text{opt}} - F(\mathbf{y}(\tau))),
\end{aligned}$$

where inequality (a) holds due to the submodularity of f and (b) due to its monotonicity.

Then, we derive the result Eq. (B.4) in Lemma B.1. □

B.5 Proof of Theorem 3.6

Proof. Similar to the proof of Theorem 3.4, we first show that the fractional vector $\mathbf{y}(1)$ satisfies $F(\mathbf{y}(1)) \geqslant (1 - 1/e^{c_\mathbf{r}})U_{\text{opt}} + F(\mathbf{r})/e^{c_\mathbf{r}}$ (①) and then that $\mathbb{E}[f(S_t)] \geqslant F(\mathbf{y}(1))$ holds in each round $t = \{1, 2, \cdots\}$, (②), which is exactly the same as Theorem 3.4. Combining them, we derive the result in Eq. (3.10).

In this proof, we will show $F(\mathbf{y}(1)) \geqslant (1 - 1/e^{c_{\mathbf{r}}})U_{\text{opt}} + F(\mathbf{r})/e^{c_{\mathbf{r}}}$ under FairCG2.

FairCG2 starts with $\mathbf{y}(0) = \mathbf{r}$ and updates $\mathbf{y}(\tau)$ with a rate $\mathbf{x}(\tau) - \mathbf{r}$. As the previous proof, we will see how much $F(\cdot)$ increases during each discretized time interval $[\tau, \tau + d\tau)$ in the `while` loop (Lines 2-9). By aplying the chain rule, we have

$$\begin{aligned}
\frac{dF(\mathbf{y}(\tau))}{d\tau} &= \sum_{u \in \mathcal{N}} \left(\frac{dy_u(\tau)}{d\tau} \cdot \frac{\partial F(\mathbf{y})}{\partial y_u}\bigg|_{\mathbf{y}=\mathbf{y}(\tau)} \right) \\
&= \sum_{u \in \mathcal{N}} \left((x_u(\tau) - r_u) \cdot \frac{\partial F(\mathbf{y})}{\partial y_u}\bigg|_{\mathbf{y}=\mathbf{y}(\tau)} \right) \\
&= \sum_{u \in \mathcal{N}} \left((x_u(\tau) - r_u) \cdot \frac{F(\mathbf{y}(\tau) \vee \mathbf{1}_u) - F(\mathbf{y}(\tau))}{1 - y_u(\tau)} \right) \\
&\geqslant \sum_{u \in \mathcal{N}} (x_u(\tau) - r_u) \cdot w_u(\tau) = (\mathbf{x}(\tau) - \mathbf{r})^{\mathsf{T}} \mathbf{w}(\tau).
\end{aligned} \tag{B.9}$$

Let $\mathbf{y}'(\tau)$ be a point in the convex polytope P_f, such that $\mathbf{y}'(\tau) - \mathbf{r} = c_\tau \mathbf{x}(\tau)$, where c_τ is a constant. Then, $\mathbf{y}'(\tau)$ satisfies the following two constraints:

$$\begin{cases} y'_u(\tau) \leqslant 1, \forall u \\ \sum_{u \in \mathcal{N}} y'_u(\tau) \leqslant k, \end{cases}$$

i.e.,

$$\begin{cases} r_u + c_\tau x_u(\tau) \leqslant 1, \forall u \\ \sum_{u \in \mathcal{N}} r_u + c_\tau k \leqslant k, \end{cases} \tag{B.10}$$

where the second inequality of Eq. (B.10) is from $\sum_{u \in \mathcal{N}} x_u(\tau) = k$ due to the fact that $\mathbf{x}(\tau)$ must be a vertex of the convex polytope P_f. Let $D_\tau = \{u \in \mathcal{N} : x_u(\tau) \neq 0\}$. Combining the constraints in Eq. (B.10), we set $c_\tau = \min\{\min_{u \in D_\tau} \frac{1 - r_u}{x_u(\tau)}, 1 - \frac{\sum_{u \in \mathcal{N}} r_u}{k}\}$ and let $c_{\mathbf{r}} = 1 - \max\{\max_u r_u, \frac{\sum_u r_u}{k}\}$. Obviously, we have $c_\tau \geqslant c_{\mathbf{r}}$ and thus $\mathbf{y}'(\tau) - \mathbf{r} \geqslant c_{\mathbf{r}} \mathbf{x}(\tau)$. Then,

the inequality in Eq. (B.9) becomes

$$\begin{aligned}&\frac{dF(\mathbf{y}(\tau))}{d\tau}\\ \geqslant&(\mathbf{x}(\tau)-\mathbf{r})^{\mathsf{T}}\mathbf{w}(\tau)\\ \geqslant&(\mathbf{y}'(\tau)-\mathbf{r})^{\mathsf{T}}\mathbf{w}(\tau)\\ \geqslant&c_{\mathbf{r}}\mathbf{x}(\tau)^{\mathsf{T}}\mathbf{w}(\tau).\end{aligned}$$

where the second inequality is from the definition of $\mathbf{x}(\tau)$.

Combining Lemma B.1, we have the differential inequality with respect to the function of $F(\mathbf{y}(\tau))$:

$$\frac{dF(\mathbf{y}(\tau))}{d\tau} \geqslant c_{\mathbf{r}}(U_{\text{opt}} - F(\mathbf{y}(\tau))).$$

Solving the above differential inequality with the initial condition $F(\mathbf{y}(0)) = F(\mathbf{r})$ under FairCG2, we have the fractional vector $\mathbf{y}(1)$ satisfying

$$F(\mathbf{y}(1)) \geqslant (1 - 1/e^{c_{\mathbf{r}}})U_{\text{opt}} + F(\mathbf{r})/e^{c_{\mathbf{r}}}. \tag{B.11}$$

Combining the result $\mathbb{E}[f(S_t)] \geqslant F(\mathbf{y}(1))$ in each round t, we derive the result in Eq. (3.10).

□

B.5.1 Proof of Theorem 3.7

Before proving Theorem 3.7, we first present a sufficient and necessary condition for a fairness requirement vector $\mathbf{r}$ being feasible. The proof is shown in Appendix B.5.2.

Lemma B.2. *A fairness requirement vector* $\mathbf{r}$ *is feasible if and only if* $\mathbf{r}^{\mathsf{T}}\mathbf{1} \leqslant k$.

The proof of Theorem 3.7 is inspired by [108]. We show that the fairness "debt" for each

element u by the end of round t is less than one, i.e., $rt - N_{u,t} < 1$ for all u in $\mathcal{N}$ and each t. However, our proof is more involved because of the combinatorial nature in each round. Specifically, in [108], only one element could be selected in each round, and a feasible requirement $\mathbf{r}$ satisfies $nr \leqslant 1$. They divide the interval $(-\infty, nr)$ into $n+1$ partitions by the points in $\{0, r, 2r, \cdots, (n-1)r\}$ and show that the fairness "debt" of each element must lie in one of the partitions in every round, which further implies $rt - N_{u,t} < nr \leqslant 1$. During this process, it is not difficult to identify the new partition to which each element belongs after the selection in each round. However, for the MMSM-CF problem we consider, a fairness requirement $\mathbf{r}$ is feasible only if $nr \leqslant k$ (Lemma B.2). We cannot do the partitions in a similar manner since n/k is not necessarily an integer. Moreover, it sometimes becomes difficult to identify the new partition to which each element belongs after the selection. In our proof, we divide the time horizon into small periods and group the elements in $\mathcal{N}$ according to their selection times. Then, we show by induction that two useful conditions about the groups hold in every period, which further implies the short-term fairness guarantee.

Proof. We prove Theorem 3.7 by showing that $\frac{1}{T}\sum_{t=1}^{T} \mathbb{I}_{\{u \in S_t\}} > r - \frac{1}{T}$ holds for every element u and any $T \in \{1, 2, \dots\}$. That is, $rT - N_{u,T} < 1$ holds for every element u and any $T \in \{1, 2, \dots\}$. We call the value $d_{u,T} \triangleq rT - N_{u,T}$ of element u as the fairness debt of element u by the end of round T. In the proof, we show that this debt is less than one for every element u and any $T = 1, 2, \cdots$.

Note that the fairness debt of each element u in round T is determined by its being selected times $N_{u,T}$ (because we assume the same fairness requirement r for every u). We focus on finding the potential trend in terms of the selection times of all elements with FairDG. In round T, we partition the ground set $\mathcal{N}$ according to the selected times of each element $N_{u,T}$. Define $J_{T,m} \triangleq \{u : N_{u,T} = m\}$. Then, we have $\mathcal{N} = J_{T,0} \cup J_{T,1} \cup \cdots \cup J_{T,T}$ for any

$T \in \{1, 2, \cdots\}$. Moreover, for simplicity, we define the following sets:

$$\bar{J}_{T,m} \triangleq \{u : N_{u,T} > m\},$$
$$\underline{J}_{T,m} \triangleq \{u : N_{u,T} < m\}.$$

Then the following lemma is the key of the proof, implying the short-term fairness in Theorem 3.7.

Lemma B.3. *For any integer $m \in \{0, 1, 2, \cdots\}$, we have*

1. $J_{\lceil \frac{m}{r} \rceil, m} \cup \bar{J}_{\lceil \frac{m}{r} \rceil, m} = \mathcal{N}$,
2. $|J_{\lceil \frac{m}{r} \rceil, m}| \leqslant \left(\frac{m+1}{r} - \lceil \frac{m}{r} \rceil\right) k$.

We provide the proof of Lemma B.3 in Appendix B.5.3.

Condition 1 in Lemma B.3 ensures that each element is selected for at least m times by the end of round $\lceil \frac{m}{r} \rceil$.

Consider an arbitrary integer $m \geqslant 0$. For any round T such that $\lceil \frac{m}{r} \rceil \leqslant T \leqslant \lceil \frac{m+1}{r} \rceil - 1$, and any element u, we have $N_{u,T} \geqslant m$ and the fairness debt of u satisfying

$$\begin{aligned} d_{u,T} =& rT - N_{u,T} \\ \leqslant& r\left(\left\lceil \frac{m+1}{r} \right\rceil - 1\right) - m \\ <& r\left(\frac{m+1}{r}\right) - m = 1. \end{aligned} \tag{B.12}$$

Therefore, we have $d_{u,T} < 1$ for every $T \in \{1, 2, \cdots\}$, implying FairDG being 1-fair. □

B.5.2 Proof of Lemma B.2

Proof. First, we prove that the condition, $\mathbf{r}^\top \mathbf{1} \leqslant k$, is necessary: if $\mathbf{r}$ is feasible, we have $\mathbf{r}^\top \mathbf{1} \leqslant k$.

If the requirement vector $\mathbf{r}$ is feasible, there exists a policy that schedules a sequence of sets $\mathcal{S} = (S_1, S_2, \dots)$ satisfying the fairness requirement in Eq. (3.1), which implies

$$\begin{aligned}\sum_{u\in\mathcal{N}} r_u &\leqslant \sum_{u\in\mathcal{N}} \liminf_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\left[\mathbb{I}_{\{u\in S_t\}}\right] \\ &\leqslant \liminf_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\left[\sum_{u\in\mathcal{N}} \mathbb{I}_{\{u\in S_t\}}\right] \\ &\overset{(a)}{\leqslant} \liminf_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} k = k,\end{aligned} \tag{B.13}$$

where (a) is from the cardinality constraint. Hence, we have $\sum_{u\in\mathcal{N}} r_u \leqslant k$, i.e., $\mathbf{r}^\top \mathbf{1} \leqslant k$.

Then, we show that the condition $\mathbf{r}^\top \mathbf{1} \leqslant k$ is also sufficient. That is, we can always find a policy satisfying the fairness requirement in Eq. (3.1) as long as $\mathbf{r}^\top \mathbf{1} \leqslant k$. Consider $T > 0$ rounds. As described in the model, in each round we can select k elements. Thus, we can treat these T rounds as kT slots, where we can select one element in each slot. Denote the kT slots as $(1^{(1)}, \cdots, T^{(1)}, 1^{(2)}, \cdots, T^{(2)}, \cdots, 1^{(k)}, \cdots, T^{(k)})$ and the ground set as $\mathcal{N} = \{u_1, u_2, \cdots, u_n\}$. Consider a policy π' that, starting from slot $1^{(1)}$, assigns each element u_i for consecutive slots one by one until the $\lceil (\sum_{j=1}^{i} r_{u_j})T \rceil$-th slot. Specifically, assign the first element u_1 in the first $\lceil r_{u_1} T \rceil$ slots, the second element u_2 in the following $\lceil (r_{u_2} + r_{u_1})T \rceil - \lceil r_{u_1} T \rceil$ slots, and so on. Since $\mathbf{r}^\top \mathbf{1} \leqslant k$, i.e., $\lceil \sum_{u_i\in\mathcal{N}} r_{u_i} T \rceil \leqslant \lceil kT \rceil = kT$, policy π' completes the above assignment for the last element u_n, and each element u_i is assigned in $\lceil (\sum_{j=1}^{i} r_{u_j})T \rceil - \lceil (\sum_{j=1}^{i-1} r_{u_j})T \rceil$ slots. Then, for any $t \leqslant T$, select the elements that are assigned in the slots, $t^{(1)}, t^{(2)}, \cdots, t^{(k)}$, in round t. For those rounds with less than k

elements, add any element that has not been selected in that round. Note that each element u_i will be scheduled in distinct rounds due to $\lceil(\sum_{j=1}^{i} r_{u_j})T\rceil - \lceil(\sum_{j=1}^{i-1} r_{u_j})T\rceil \leqslant \lceil r_{u_i}T\rceil \leqslant T$. Taking limits yields that the selection fraction of each element u_i satisfies

$$\begin{aligned}
&\liminf_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\left[\mathbb{I}_{\{u_i\in S_t\}}\right] \\
\geqslant &\liminf_{T\to\infty} \frac{\left\lceil(\sum_{j=1}^{i} r_{u_j})T\right\rceil - \left\lceil(\sum_{j=1}^{i-1} r_{u_j})T\right\rceil}{T} \\
\geqslant &\liminf_{T\to\infty} \frac{\lceil r_{u_i}T\rceil - 1}{T} = r_u,
\end{aligned} \tag{B.14}$$

Therefore, with the condition $\mathbf{r}^\top\mathbf{1} \leqslant k$, policy π' satisfies the fairness requirements Eq. (3.1).

□

B.5.3 Proof of Lemma B.3

We prove Lemma B.3 by induction. In the inductive step, we assume that the two conditions hold for any $m \in \{0, 1, 2, \cdots\}$ and then show that they still hold for $m + 1$. First, we present two important results based on the the assumption and then directly use them in the inductive step.

We divide the time horizon into small periods at $\lceil\frac{m}{r}\rceil$, where $m = 0, 1, 2, \cdots$. Denote the m-th period as $\mathcal{T}_m$, i.e., $\mathcal{T}_m = \{\lceil\frac{m-1}{r}\rceil + 1, \cdots, \lceil\frac{m}{r}\rceil\}$ for $m = 1, 2, \cdots$.

Assume that the two conditions hold for m. We have $N_{u,\lceil\frac{m}{r}\rceil} \geqslant m$ for every element u and the following three observations.

Observation 1: For any element u with $N_{u,t-1} = m$, i.e., $u \in J_{t-1,m}$, we have the following observations.

1. if $u \in S_t$ (selected), then $u \in J_{t,m+1}$,

2. if $u \notin S_t$ (not selected), then $u \in J_{t,m}$.

Observation 2: Assume $N_{u,\lceil \frac{m}{r} \rceil} \geqslant m$ for every u in $\mathcal{N}$. For $t \in \mathcal{T}_{m+1} \backslash \{\lceil \frac{m+1}{r} \rceil\}$, we have the following results:

1. $A_t = J_{t-1,m}$.

2. $J_{t,m} \subseteq J_{t-1,m}$

3. If $|A_t| \geqslant k$, then $|J_{t,m}| = |J_{t-1,m}| - k$, and $|J_{t,m+1}| = |J_{t-1,m+1}| + k$.

4. If $|A_t| < k$, then $|J_{t,m}| = 0$, and $|J_{t,m+1}| \leqslant |J_{t-1,m}| + |J_{t-1,m+1}|$.

Proof. For any $t = \lceil \frac{m}{r} \rceil + 1, \cdots, \lceil \frac{m+1}{r} \rceil - 1$, we first have $N_{u,t-1} \geqslant m$ for any u in $\mathcal{N}$, i.e., $J_{t-1,m} \cup \bar{J}_{t-1,m} = \mathcal{N}$.

1. i) $\forall u \in J_{t-1,m}$,

$$rt - N_{u,t-1} > r\lceil \frac{m}{r} \rceil - m \geqslant 0$$

$$\Rightarrow u \in A_t.$$

ii) $\forall u \in \bar{J}_{t-1,m}$, $N_{u,t-1} \geqslant m+1$ and

$$\begin{aligned} rt - N_{u,t-1} &\leqslant r\left(\lceil \frac{m+1}{r} \rceil - 1\right) - (m+1) \\ &< r \cdot \frac{m+1}{r} - (m+1) = 0 \end{aligned}$$

$$\Rightarrow u \notin A_t.$$

Then, we have the result 1.

2. Notice that $J_{t-1,m} \cup \bar{J}_{t-1,m} = \mathcal{N}$.

i) $\forall u \in J_{t-1,m}$,

$$u \in \begin{cases} J_{t,m+1}, & \text{if } u \in S_t, \\ J_{t,m}, & \text{if } u \notin S_t. \end{cases} \tag{B.15}$$

i.e.,

$$u \in \begin{cases} \bar{J}_{t,m}, & \text{if } u \in S_t, \\ J_{t,m}, & \text{if } u \notin S_t. \end{cases} \tag{B.16}$$

ii) $\forall u \in \bar{J}_{t-1,m}$, we have

$$u \in \bar{J}_{t,m}, \tag{B.17}$$

whenever u is selected or not.

From above, we know that every element u in $J_{t,m}$ is from $J_{t-1,m}$ and then the result holds.

3. if $|A_t| \geqslant k$, then $S_t \subseteq A_t$ according to FairDG, indicating that k elements from $J_{t-1,m}$ are selected as S_t. Hence, the selection times of element u is

$$N_{u,t} = \begin{cases} m+1, & \forall u \in S_t, \\ m, & \forall u \in J_{t-1,m} \backslash S_t, \\ N_{u,t-1} \geqslant m+1, & \forall u \in \bar{J}_{t-1,m}. \end{cases} \tag{B.18}$$

We have $J_{t,m} = J_{t-1,m} \backslash S_t$ and then $|J_{t,m}| = |J_{t-1,m}| - k$. Similarly, $J_{t,m+1} = J_{t-1,m+1} \cup S_t$ and then $|J_{t,m+1}| = |J_{t-1,m+1}| + k$.

4. If $|A_t| < k$, then every element in A_t is selected in round t, i.e., $A_t \subseteq S_t$. Hence, the

selection times of element u is

$$N_{u,t} = \begin{cases} m+1, & \forall u \in J_{t-1,m}, \\ N_{u,t-1}+1 \geqslant m+2, & \forall u \in S_t \backslash J_{t-1,m}, \\ N_{u,t-1} \geqslant m+1, & \forall u \in \bar{J}_{t-1,m} \backslash S_t. \end{cases} \tag{B.19}$$

Then, we have $J_{t,m} = \emptyset$ and $J_{t,m+1} = J_{t-1,m} \cup J_{t-1,m+1} \backslash (S_t \cap J_{t-1,m+1})$. Obviously, the result holds. □

Observation 3: Assume $N_{u,\lceil \frac{m}{r} \rceil} \geqslant m$ for every u in $\mathcal{N}$. In round $t = \lceil \frac{m+1}{r} \rceil$, we have:

$$A_t = J_{t-1,m} \cup J_{t-1,m+1}.$$

Proof. Obviously, we have $N_{u,\lceil \frac{m+1}{r} \rceil -1} \geqslant m$ for any u in $\mathcal{N}$ and thus, $J_{\lceil \frac{m+1}{r} \rceil -1,m} \cup J_{\lceil \frac{m+1}{r} \rceil -1,m+1} \cup \bar{J}_{\lceil \frac{m+1}{r} \rceil -1,m+1} = \mathcal{N}$.

i) $\forall u \in J_{\lceil \frac{m+1}{r} \rceil -1,m}$,

$$rt - N_{u,t-1} = r\lceil \frac{m+1}{r} \rceil - m > 0$$

$$\Rightarrow u \in A_t.$$

ii) $\forall u \in J_{\lceil \frac{m+1}{r} \rceil -1,m+1}$,

$$rt - N_{u,t-1} = r\lceil \frac{m+1}{r} \rceil - (m+1) \geqslant 0$$

$$\Rightarrow u \in A_t.$$

iii) $\forall u \in \bar{J}_{\lceil \frac{m+1}{r} \rceil - 1, m+1}$, $N_{u, \lceil \frac{m+1}{r} \rceil - 1} \geqslant m+2$ and

$$\begin{aligned} rt - N_{u,t-1} &= r\lceil \frac{m+1}{r} \rceil - (m+2) \\ &\leqslant r\left(\lceil \frac{m+2}{r} \rceil - 1\right) - (m+2) \\ &< r \cdot \frac{m+2}{r} - (m+2) = 0 \end{aligned}$$

$$\Rightarrow u \notin A_t.$$

□

Proof of Lemma B.3. We prove the lemma by induction.

Induction base case ($m = 0$): At the very beginning, we have $N_{u,0} = 0$ for any u in $\mathcal{N}$ and thus, $\mathcal{N} = J_{0,0} \cup \bar{J}_{0,0}$ trivially. Moreover, we have $|J_{0,0}| = n \leqslant \frac{1}{r} \cdot k$. That is, results 1. and 2. in Lemma B.3 hold for $m = 0$ trivially.

Inductive Step: Assume that both the results hold for m. Combining the above observations, we show the results hold for $m+1$ in the following.

Consider three cases:

i) when $\lceil \frac{m+1}{r} \rceil > \lceil \frac{m}{r} \rceil + 1$, there exists a round $\tau \in \{\lceil \frac{m}{r} \rceil + 1, \cdots, \lceil \frac{m+1}{r} \rceil - 1\}$, such that $|A_\tau| < k$;

ii) when $\lceil \frac{m+1}{r} \rceil > \lceil \frac{m}{r} \rceil + 1$, for every $\tau \in \{\lceil \frac{m}{r} \rceil + 1, \cdots, \lceil \frac{m+1}{r} \rceil - 1\}$, $|A_\tau| \geqslant k$ holds;

iii) $\lceil \frac{m+1}{r} \rceil = \lceil \frac{m}{r} \rceil + 1$.

Case i): There exists $\tau \in \mathcal{T}_{m+1} \backslash \{\lceil \frac{m+1}{r} \rceil\}$ such that $|A_\tau| < k$.

Due to $A_\tau = J_{\tau-1,m}$ and $\mathcal{N} = J_{\tau-1,m} \cup \bar{J}_{\tau-1,m}$, we have

$$\forall u \in J_{\tau-1,m} \Rightarrow u \in S_\tau \Rightarrow u \in J_{\tau,m+1}$$

$$\forall v \in \bar{J}_{\tau-1,m} \Rightarrow \begin{cases} v \in \bar{J}_{\tau,m+1} & \text{if } v \in S_\tau \\ v \in \bar{J}_{\tau,m} & \text{if } v \notin S_\tau \end{cases} \tag{B.20}$$

$$\Rightarrow v \in \bar{J}_{\tau,m} = J_{\tau,m+1} \cup \bar{J}_{\tau,m+1}$$

Hence, we derive $\mathcal{N} = J_{\tau,m+1} \cup \bar{J}_{\tau,m+1}$. For each element u, we have $N_{u,\lceil \frac{m+1}{r} \rceil} \geqslant N_{u,\tau} \geqslant m+1$, and thus, $u \in J_{\lceil \frac{m+1}{r} \rceil,m} \cup \bar{J}_{\lceil \frac{m+1}{r} \rceil,m+1}$. Therefore, we obtain the first result $\mathcal{N} = J_{\lceil \frac{m+1}{r} \rceil,m} \cup \bar{J}_{\lceil \frac{m+1}{r} \rceil,m+1}$ in Lemma B.3.

Besides, we have $J_{\lceil \frac{m+1}{r} \rceil-1,m} \subseteq J_{\tau,m} = \emptyset$ and then $A_{\lceil \frac{m+1}{r} \rceil} = J_{\lceil \frac{m+1}{r} \rceil-1,m+1}$ and $\mathcal{N} = J_{\lceil \frac{m+1}{r} \rceil-1,m+1} \cup \bar{J}_{\lceil \frac{m+1}{r} \rceil-1,m+1}$.

1) If $|A_{\lceil \frac{m+1}{r} \rceil}| \leqslant k$, then all elements with $N_{u,\lceil \frac{m+1}{r} \rceil-1} = m+1$ are selected. We have

$\forall u \in J_{\lceil \frac{m+1}{r} \rceil-1,m+1} \Rightarrow u \in S_{\lceil \frac{m+1}{r} \rceil}$, and then, $u \in J_{\lceil \frac{m+1}{r} \rceil,m+2}$

$\Rightarrow |J_{\lceil \frac{m+1}{r} \rceil,m+1}| = 0$ satisfying the result trivially.

2) If $|A_{\lceil \frac{m+1}{r} \rceil}| > k$, then k elements from $J_{\lceil \frac{m+1}{r} \rceil-1,m+1}$ are selected in this round. Then, we

have

$$
\begin{aligned}
&|J_{\lceil \frac{m+1}{r} \rceil, m+1}| \\
&= |J_{\lceil \frac{m+1}{r} \rceil - 1, m+1}| - k \\
&\leqslant n - k \\
&\leqslant n - k + \left(\frac{m+1}{r} + 1 - \lceil \frac{m+1}{r} \rceil \right) k \\
&\leqslant \frac{1}{r} \cdot k + \frac{m+1}{r} \cdot k - \lceil \frac{m+1}{r} \rceil \cdot k \\
&\leqslant \left(\frac{m+2}{r} - \lceil \frac{m+1}{r} \rceil \right) k.
\end{aligned}
$$

Hence, the second statement in Lemma B.3 holds for $m+1$.

Case ii): When $\lceil \frac{m+1}{r} \rceil > \lceil \frac{m}{r} \rceil + 1$, for every $\tau \in \{\lceil \frac{m}{r} \rceil + 1, \cdots, \lceil \frac{m+1}{r} \rceil - 1\}$, $|A_\tau| \geqslant k$ holds.

According to Observation 2, we have

$$|J_{\lceil \frac{m}{r} \rceil + 1, m}| = |J_{\lceil \frac{m}{r} \rceil, m}| - k$$

$$|J_{\lceil \frac{m}{r} \rceil + 2, m}| = |J_{\lceil \frac{m}{r} \rceil + 1, m}| - k$$

$$\cdots$$

$$|J_{\lceil \frac{m+1}{r} \rceil - 1, m}| = |J_{\lceil \frac{m+1}{r} \rceil - 2, m}| - k$$

$$
\begin{aligned}
\Rightarrow |J_{\lceil \frac{m+1}{r} \rceil - 1, m}| &= |J_{\lceil \frac{m}{r} \rceil, m}| - \left(\lceil \frac{m+1}{r} \rceil - 1 - \lceil \frac{m}{r} \rceil \right) k \\
&\leqslant \left(\frac{m+1}{r} - \lceil \frac{m}{r} \rceil \right) k - \left(\lceil \frac{m+1}{r} \rceil - 1 - \lceil \frac{m}{r} \rceil \right) k \\
&= \left(\frac{m+1}{r} + 1 - \lceil \frac{m+1}{r} \rceil \right) k.
\end{aligned} \tag{B.21}
$$

In round $\lceil \frac{m+1}{r} \rceil$, we have $A_{\lceil \frac{m+1}{r} \rceil} = J_{\lceil \frac{m+1}{r} \rceil - 1, m} \cup J_{\lceil \frac{m+1}{r} \rceil - 1, m+1}$ according to Observation 3.

1) If $|A_{\lceil \frac{m+1}{r} \rceil}| \leqslant k$,

$\forall u \in J_{\lceil \frac{m+1}{r} \rceil - 1, m}$ will be selected, i.e., $u \in S_{\lceil \frac{m+1}{r} \rceil}$, and then, $u \in J_{\lceil \frac{m+1}{r} \rceil, m+1}$;

$\forall u \in J_{\lceil \frac{m+1}{r} \rceil - 1, m+1}$ will be selected, i.e., $u \in S_{\lceil \frac{m+1}{r} \rceil}$, and then, $u \in J_{\lceil \frac{m+1}{r} \rceil, m+2}$. That is, the number of selection times for each element will be

$$
\begin{aligned}
& N_{u, \lceil \frac{m+1}{r} \rceil} \\
= & \begin{cases}
m+1, & \forall u \in J_{\lceil \frac{m+1}{r} \rceil - 1, m}, \\
m+2, & \forall u \in J_{\lceil \frac{m+1}{r} \rceil - 1, m+1}, \\
N_{u, \lceil \frac{m+1}{r} \rceil - 1} + 1 \geqslant m+3, & \forall u \in S_{\lceil \frac{m+1}{r} \rceil} \backslash A_{\lceil \frac{m+1}{r} \rceil}, \\
N_{u, \lceil \frac{m+1}{r} \rceil - 1} \geqslant m+2, & \forall u \in \mathcal{N} \backslash S_{\lceil \frac{m+1}{r} \rceil}.
\end{cases}
\end{aligned}
\tag{B.22}
$$

Then, we have $J_{\lceil \frac{m+1}{r} \rceil, m} = \emptyset$, indicating $\mathcal{N} = J_{\lceil \frac{m+1}{r} \rceil, m+1} \cup \bar{J}_{\lceil \frac{m+1}{r} \rceil, m+1}$. Moreover, by Eq. (B.21) we have

$$
\begin{aligned}
|J_{\lceil \frac{m+1}{r} \rceil, m+1}| &= |J_{\lceil \frac{m+1}{r} \rceil - 1, m}| \\
&\leqslant \left(\frac{m+1}{r} + 1 - \lceil \frac{m+1}{r} \rceil \right) k \\
&\leqslant \left(\frac{m+2}{r} - \lceil \frac{m+1}{r} \rceil \right) k,
\end{aligned}
$$

where the last step is from $r \leqslant 1$. That is, the second result in Lemma B.3 holds for $m+1$ when $|A_{\lceil \frac{m+1}{r} \rceil}| \leqslant k$.

2) Consider the case when $|A_{\lceil \frac{m+1}{r} \rceil}| > k$.

Note that every element u in $J_{\lceil \frac{m+1}{r} \rceil - 1, m}$ with a larger fairness debt (smaller $N_{u, \lceil \frac{m+1}{r} \rceil - 1}$) have a higher priority of being selected than any element in $J_{\lceil \frac{m+1}{r} \rceil - 1, m+1}$.

Since $|J_{\lceil \frac{m+1}{r} \rceil - 1, m}| \leqslant \left(\frac{m+1}{r} + 1 - \lceil \frac{m+1}{r} \rceil \right) k \leqslant k$, every element in $J_{\lceil \frac{m+1}{r} \rceil - 1, m}$ will be selected. That is, for $\forall u \in J_{\lceil \frac{m+1}{r} \rceil - 1, m}$, we have $u \in S_{\lceil \frac{m+1}{r} \rceil}$, and then, $u \in J_{\lceil \frac{m+1}{r} \rceil, m+1}$.

Besides, $k - |J_{\lceil \frac{m+1}{r} \rceil - 1, m}|$ elements from $J_{\lceil \frac{m+1}{r} \rceil - 1, m+1}$ are selected.

Then, we have $J_{\lceil \frac{m+1}{r} \rceil, m} = \emptyset$, indicating $\mathcal{N} = J_{\lceil \frac{m+1}{r} \rceil, m+1} \cup \bar{J}_{\lceil \frac{m+1}{r} \rceil, m+1}$.

For $\forall u \in S_{\lceil \frac{m+1}{r} \rceil} \backslash J_{\lceil \frac{m+1}{r} \rceil - 1, m}$, we have $u \in J_{\lceil \frac{m+1}{r} \rceil - 1, m+1}$ ($|A_{\lceil \frac{m+1}{r} \rceil}| > k$), and then, $u \in J_{\lceil \frac{m+1}{r} \rceil, m+2}$. In details, the number of selection times of each element will be

$$
\begin{aligned}
& N_{u, \lceil \frac{m+1}{r} \rceil} \\
= & \begin{cases} m+1, & \forall u \in J_{\lceil \frac{m+1}{r} \rceil - 1, m}, \\ m+2, & \forall u \in J_{\lceil \frac{m+1}{r} \rceil - 1, m+1} \cap S_{\lceil \frac{m+1}{r} \rceil}, \\ m+1, & \forall u \in J_{\lceil \frac{m+1}{r} \rceil - 1, m+1} \backslash S_{\lceil \frac{m+1}{r} \rceil}, \\ N_{u, \lceil \frac{m+1}{r} \rceil - 1} \geqslant m+2, & \text{otherwise} \end{cases}
\end{aligned}
\tag{B.23}
$$

Finally, we have

$$
\begin{aligned}
& |J_{\lceil \frac{m+1}{r} \rceil, m+1}| \\
& = |J_{\lceil \frac{m+1}{r} \rceil - 1, m}| + |J_{\lceil \frac{m+1}{r} \rceil - 1, m+1}| - \left(k - |J_{\lceil \frac{m+1}{r} \rceil - 1, m}|\right) \\
& \leqslant n - k + |J_{\lceil \frac{m+1}{r} \rceil - 1, m}| \\
& \leqslant n - k + \left(\frac{m+1}{r} - \lceil \frac{m}{r} \rceil\right) k \\
& \leqslant \frac{1}{r} \cdot k - k + \frac{m+1}{r} \cdot k - \lceil \frac{m}{r} \rceil \cdot k \\
& = \frac{m+2}{r} \cdot k - (1 + \lceil \frac{m}{r} \rceil) k \\
& \leqslant \left(\frac{m+2}{r} - \lceil \frac{m+1}{r} \rceil\right) k.
\end{aligned}
$$

That is, the second result in Lemma B.3 holds for $m+1$ when $|A_{\lceil \frac{m+1}{r} \rceil}| > k$.

Case iii): $\lceil \frac{m+1}{r} \rceil = \lceil \frac{m}{r} \rceil + 1$.

In round $\lceil \frac{m+1}{r} \rceil$, we have $A_{\lceil \frac{m+1}{r} \rceil} = J_{\lceil \frac{m}{r} \rceil, m} \cup J_{\lceil \frac{m}{r} \rceil, m+1}$ according to Observation 3. We

consider the two cases 1) $|A_{\lceil \frac{m+1}{r} \rceil}| \leqslant k$ and 2) $|A_{\lceil \frac{m+1}{r} \rceil}| > k$ as in the other two cases.

1) If $|A_{\lceil \frac{m+1}{r} \rceil}| \leqslant k$,

$\forall u \in J_{\lceil \frac{m}{r} \rceil, m}$ will be selected, i.e., $u \in S_{\lceil \frac{m+1}{r} \rceil}$, and then, $u \in J_{\lceil \frac{m+1}{r} \rceil, m+1}$;

$\forall u \in J_{\lceil \frac{m}{r} \rceil, m+1}$ will be selected, i.e., $u \in S_{\lceil \frac{m+1}{r} \rceil}$, and then, $u \in J_{\lceil \frac{m+1}{r} \rceil, m+2}$. That is, the number of selection times for each element will be

$$
\begin{aligned}
& N_{u, \lceil \frac{m+1}{r} \rceil} \\
& = \begin{cases} m+1, & \forall u \in J_{\lceil \frac{m}{r} \rceil, m}, \\ m+2, & \forall u \in J_{\lceil \frac{m}{r} \rceil, m+1}, \\ N_{u, \lceil \frac{m}{r} \rceil} + 1 \geqslant m+3, & \forall u \in S_{\lceil \frac{m+1}{r} \rceil} \backslash A_{\lceil \frac{m+1}{r} \rceil}, \\ N_{u, \lceil \frac{m}{r} \rceil} \geqslant m+2, & \forall u \in \mathcal{N} \backslash S_{\lceil \frac{m+1}{r} \rceil}. \end{cases}
\end{aligned} \tag{B.24}
$$

Then, we have $J_{\lceil \frac{m+1}{r} \rceil, m} = \emptyset$, indicating $\mathcal{N} = J_{\lceil \frac{m+1}{r} \rceil, m+1} \cup \bar{J}_{\lceil \frac{m+1}{r} \rceil, m+1}$. Moreover, we have

$$
\begin{aligned}
|J_{\lceil \frac{m+1}{r} \rceil, m+1}| &= |J_{\lceil \frac{m}{r} \rceil, m}| \\
&\leqslant \left(\frac{m+1}{r} - \lceil \frac{m}{r} \rceil \right) k \\
&= \left(\frac{m+1}{r} + 1 - \lceil \frac{m+1}{r} \rceil \right) k \\
&\leqslant \left(\frac{m+2}{r} - \lceil \frac{m+1}{r} \rceil \right) k,
\end{aligned}
$$

where the last step is from $r \leqslant 1$. That is, the second result in Lemma B.3 holds for $m+1$ when $|A_{\lceil \frac{m+1}{r} \rceil}| < k$.

2) Consider the case when $|A_{\lceil \frac{m+1}{r} \rceil}| > k$.

Note that every element u in $J_{\lceil \frac{m}{r} \rceil, m}$ with a larger fairness debt (smaller $N_{u, \lceil \frac{m}{r} \rceil}$) have a higher priority of being selected than any element in $J_{\lceil \frac{m}{r} \rceil, m+1}$.

Since $|J_{\lceil \frac{m}{r} \rceil,m}| \leqslant \left(\frac{m+1}{r} + 1 - \lceil \frac{m+1}{r} \rceil\right) k \leqslant k$, every element in $J_{\lceil \frac{m}{r} \rceil,m}$ will be selected. That is, for $\forall u \in J_{\lceil \frac{m}{r} \rceil,m}$, we have $u \in S_{\lceil \frac{m+1}{r} \rceil}$, and then, $u \in J_{\lceil \frac{m+1}{r} \rceil,m+1}$.

Besides, $k - |J_{\lceil \frac{m}{r} \rceil,m}|$ elements from $J_{\lceil \frac{m}{r} \rceil,m+1}$ are selected.

Then, we have $J_{\lceil \frac{m+1}{r} \rceil,m} = \emptyset$, indicating $\mathcal{N} = J_{\lceil \frac{m+1}{r} \rceil,m+1} \cup \bar{J}_{\lceil \frac{m+1}{r} \rceil,m+1}$.

For $\forall u \in S_{\lceil \frac{m+1}{r} \rceil} \backslash J_{\lceil \frac{m}{r} \rceil,m}$, we have $u \in J_{\lceil \frac{m}{r} \rceil,m+1}$ ($|A_{\lceil \frac{m+1}{r} \rceil}| > k$), and then, $u \in J_{\lceil \frac{m+1}{r} \rceil,m+2}$. In details, the number of selection times of each element will be

$$
\begin{aligned}
& N_{u,\lceil \frac{m+1}{r} \rceil} \\
= & \begin{cases} m+1, & \forall u \in J_{\lceil \frac{m}{r} \rceil,m}, \\ m+2, & \forall u \in J_{\lceil \frac{m}{r} \rceil,m+1} \cap S_{\lceil \frac{m+1}{r} \rceil}, \\ m+1, & \forall u \in J_{\lceil \frac{m}{r} \rceil,m+1} \backslash S_{\lceil \frac{m+1}{r} \rceil}, \\ N_{u,\lceil \frac{m}{r} \rceil} \geqslant m+2, & \forall u \in \mathcal{N} \backslash J_{\lceil \frac{m}{r} \rceil,m} \backslash J_{\lceil \frac{m}{r} \rceil,m+1}. \end{cases}
\end{aligned} \tag{B.25}
$$

Finally, we have

$$
\begin{aligned}
& |J_{\lceil \frac{m+1}{r} \rceil,m+1}| \\
& = |J_{\lceil \frac{m+1}{r} \rceil-1,m}| + |J_{\lceil \frac{m+1}{r} \rceil-1,m}| - \left(k - |J_{\lceil \frac{m+1}{r} \rceil-1,m}|\right) \\
& \leqslant n - k + |J_{\lceil \frac{m+1}{r} \rceil-1,m}| \\
& \leqslant n - k + \left(\frac{m+1}{r} - \lceil \frac{m}{r} \rceil\right) k \\
& \leqslant \frac{1}{r} \cdot k - k + \frac{m+1}{r} \cdot k - \lceil \frac{m}{r} \rceil \cdot k \\
& = \frac{m+2}{r} \cdot k - (1 + \lceil \frac{m}{r} \rceil) k \\
& \leqslant \left(\frac{m+2}{r} - \lceil \frac{m+1}{r} \rceil\right) k.
\end{aligned}
$$

That is, the second result in Lemma B.3 holds for $m+1$ when $|A_{\lceil \frac{m+1}{r} \rceil}| \leqslant k$. Therefore, the

two results in Lemma B.3 hold for any $m \geqslant 0$. We complete the proof. $\square$

Appendix C

Proofs for Chapter 4

C.1 Proofs and Supplementary Materials for Section 4.5

In this section, we provide the proof of the theorems in Section 4.5 to show that DP-DPE is differentially private under different models. Before considering different models, we describe the Gaussian Mechanism in the differential privacy literature.

Let $f : \mathcal{U} \to \mathbb{R}^s$ be a vector-valued function operating on databases. The ℓ_2-sensitivity of f, denoted $\Delta_2 f$ is the maximum over all pairs $\mathcal{U}, \mathcal{U}'$ of neighboring datasets of $\|f(\mathcal{U}) - f(\mathcal{U}')\|$. The Gaussian mechanism adds independent noise drawn from a Gaussian with mean zero and standard deviation slightly greater than $\Delta_2 f \sqrt{\ln(1/\delta)}/\varepsilon$ to each element of its output [53].

Theorem C.1. *(Gaussian Mechanism [50]). Given any vector-valued function $f : \mathcal{U}^* \to \mathbb{R}^s$, define $\Delta_2 \triangleq \max_{\mathcal{U},\mathcal{U}' \in \mathcal{U}^*} \|f(\mathcal{U}) - f(\mathcal{U}')\|_2$. Let $\sigma = \Delta_2 \sqrt{2\ln(1.25/\delta)}/\varepsilon$. The Gaussian mechanism, which adds independently drawn random noise from $\mathcal{N}(0, \sigma^2)$ to each output of $f(\cdot)$, i.e. returning $f(\mathcal{U}) + (\gamma_1, \ldots, \gamma_s)$ with $\gamma_j \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma^2)$, ensures (ε, δ)-DP.*

C.1.1 The Central Model

The Privatizer $\mathcal{P}$ in the central model adds Gaussian noise to the averaged local performance of each action directly, i.e., at analyzer $\mathcal{A}$ while doing identity mapping with $\mathcal{R}$ and $\mathcal{S}$. That

is,

$$\tilde{y}_l = \mathcal{P}\left(\{\vec{y}_l^u\}_{u\in U_l}\right) = \mathcal{A}\left(\{\vec{y}_l^u\}_{u\in U_l}\right) = \frac{1}{|U_l|}\sum_{u\in U_l}\vec{y}_l^u + (\gamma_1,\ldots,\gamma_{s_l}), \tag{C.1}$$

where $\gamma_j \overset{i.i.d.}{\sim} \mathcal{N}(0,\sigma_{nc}^2)$.

Proof of Theorem 4.4. Recall that $\mathcal{U}_T = (U_l)_{l=1}^L \subseteq \mathcal{U}$ represents the sequence of participating clients in the total L phases. Let $\mathcal{U}_T, \mathcal{U}_T' \subseteq \mathcal{U}$ be the sequence of participating clients differing on a single client and U_l, U_l' be the sets of participating clients in l-th phase corresponding to $\mathcal{U}_T$ and $\mathcal{U}_T'$ respectively. Note that the ℓ_2 sensitivity of the average $\frac{1}{|U_l|}\sum_{u\in U_l}\vec{y}_l^u$ is

$$\begin{aligned}
\Delta_2 &= \max_{\mathcal{U}_T,\mathcal{U}_T'}\left\|\frac{1}{|U_l|}\sum_{u\in U_l}\vec{y}_l^u - \frac{1}{|U_l|}\sum_{u\in U_l'}\vec{y}_l^u\right\|_2 \\
&\leqslant \frac{1}{|U_l|}\max_{u,u'\in\mathcal{U}}\|\vec{y}_l^u - \vec{y}_l^{u'}\|_2 \\
&\leqslant \frac{2}{|U_l|}\max_{u\in\mathcal{U}}\|\vec{y}_l^u\|_2 \leqslant \frac{2B\sqrt{s_l}}{|U_l|},
\end{aligned} \tag{C.2}$$

where the last step is because s_l is the dimension of y_l^u and then $\|\vec{y}_l^u\|_2^2 \leqslant s_l\|y_l^u(\mathbf{x})\|_1^2 \leqslant s_l B^2$. Let $\sigma_{nc} = \frac{2B\sqrt{2s_l\ln(1.25/\delta)}}{\varepsilon|U_l|}$. According to Theorem D.13, we have that the output $\tilde{y}_l$ of the PRIVATIZER in the central model is (ε,δ)-DP with respect to $\mathcal{U}_T$.

Combining the result of Proposition 2.1 in [50], we derive that DP-DPE with a PRIVATIZER in the central model is (ε,δ)-DP. □

C.1.2 The Local Model

The PRIVATIZER $\mathcal{P}$ in the local model applies Gaussian mechanism to the local average performance of each action ($\vec{y}_l^u$) with $\mathcal{R}$ while doing identity mapping with $\mathcal{S}$ and pure averaging

with $\mathcal{A}$. That is,

$$\tilde{y}_l = \mathcal{P}\left(\{\vec{y}_l^u\}_{u\in U_l}\right) = \frac{1}{|U_l|}\sum_{u\in U_l}\mathcal{R}(\vec{y}_l^u) = \frac{1}{|U_l|}\sum_{u\in U_l}\left(\vec{y}_l^u + \gamma_u\right),$$

where $\gamma_u \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma_{nl}^2 I_{s_l})$.

Proof of Theorem 4.6. An algorithm is (ε, δ)-LDP if the output of the local randomizer $\mathcal{R}$ is (ε, δ)-DP. In the local model of Privatizer, we have

$$\mathcal{R}(\vec{y}_l^u) = \vec{y}_l^u + \gamma_u. \tag{C.3}$$

For any input of local parameter estimator $\vec{y}_l$, the ℓ_2 sensitivity is

$$\Delta_2 = \max_{\vec{y}_l, \vec{y}_l'} \|\vec{y}_l - \vec{y}_l'\|_2 \leqslant 2\max\|\vec{y}_l^u\|_2 \leqslant 2B\sqrt{s_l}. \tag{C.4}$$

Let $\sigma_{nl} = \frac{2B\sqrt{2s_l\ln(1.25/\delta)}}{\varepsilon}$. According to Theorem D.13, we have that the output of the local randomizer $\mathcal{R}$ of the Privatizer in the local model is (ε, δ)-DP. That is, the DP-DPE algorithm instantiated with the Privatizer in the local model is (ε, δ)-LDP. □

C.1.3 The Shuffle Model

In the shuffle model, the Privatizer $\mathcal{P}$ operates under the cooperation of the local randomizer $\mathcal{R}$ at each client, the shuffler $\mathcal{S}$, and the analyzer $\mathcal{A}$ at the central server. First, we present the implementation of each component of the Privatizer $\mathcal{P}$, including $\mathcal{R}$, $\mathcal{S}$, and $\mathcal{A}$, in the shuffle model in Algorithm 7.

The Privatizer $\mathcal{P}$ in the shuffle model adds binary bits to the local average performance

of each played action (after being converted to binary representation) at each client, i.e., at local randomizer $\mathcal{R}$, and then shuffles all bits reported from all participating clients via a shuffler $\mathcal{S}$ before sending them to the central server, where the analyzer $\mathcal{A}$ output an unbiased and private estimator of the average of local parameters. That is,

$$\tilde{y}_l = \mathcal{P}\left(\{\vec{y}_l^u\}_{u\in U_l}\right) = \mathcal{A}(\mathcal{S}(\{\mathcal{R}(\vec{y}_l^u)\}_{u\in U_l})). \tag{C.6}$$

Before proving Theorem 4.8, we first show that the shuffle protocol in Algorithm 7 is (ε, δ)-SDP.

In this proof, we use $(\cdot)_j$ to denote the j-th element of an vector.

Theorem C.2. *For any $\varepsilon \in (0, 15), \delta \in (0, 1)$, Algorithm 7 is (ε, δ)-SDP, unbiased, and has error distribution which is sub-Gaussian with variance $\sigma_{ns}^2 = O\left(\frac{\Delta_2^2 \ln^2(s/\delta)}{\varepsilon^2 |U|^2}\right)$ and independent of the inputs.*

Proof. The proof for the privacy part follows from the SDP guarantee of vector summation protocol in [35]. In the following, we try to show the remaining part of the above theorem.

Consider an arbitrary coordinate $j \in [s]$. Note that the sum of the messages produced by $\mathcal{R}$ at client u is: $\bar{w}_{u,j} + \gamma_1 + \gamma_2$. Since γ_1 is drawn from $\texttt{Ber}(w_{u,j}g/(2\Delta_2) - \bar{w}_{u,j})$, which has expectation $\mathbb{E}[\gamma_1] = w_{u,j}g/(2\Delta_2) - \bar{w}_{u,j}$ and is 1/2-sub-Gaussian according to Hoeffding Lemma. Meanwhile, $\gamma_2 \sim \texttt{Bin}(b, p)$ indicates $\mathbb{E}[\gamma_2] = bp$ and $\sqrt{b}/2$-sub-Gaussian. Recall that $z_j = \frac{2\Delta_2}{g|U|}((\sum_{i=1}^{(g+b)U}(\vec{\phi}_j)_i) - b|U|p) = \frac{2\Delta_2}{g|U|}(\sum_{u\in U}(\bar{w}_{u,j} + \gamma_1 + \gamma_2) - b|U|p)$ and $o_j = z_j - \Delta_2$.

We have

$$
\begin{aligned}
\mathbb{E}[o_j] =& \mathbb{E}[z_j - \Delta_2] = \mathbb{E}\left[\frac{2\Delta_2}{g|U|}\left(\sum_{u\in U}(\bar{w}_{u,j} + \gamma_1 + \gamma_2) - b|U|p\right) - \Delta_2\right] \\
=& \frac{2\Delta_2}{g|U|}\left(\sum_{u\in U}(\bar{w}_{u,j} + \mathbb{E}[\gamma_1] + \mathbb{E}[\gamma_2]) - b|U|p\right) - \Delta_2 \\
=& \frac{2\Delta_2}{g|U|}\left(\sum_{u\in U}\left(\frac{w_{u,j}g}{2\Delta_2} + bp\right) - b|U|p\right) - \Delta_2 = \frac{1}{|U|}\sum_{u\in U} w_{u,j} - \Delta_2 \\
=& \frac{1}{|U|}\sum_{u\in U}((y_u)_j + \Delta_2) - \Delta_2 = \frac{1}{|U|}\sum_{u\in U}(y_u)_j,
\end{aligned}
$$

which indicates that the output is unbiased estimator of the average. In addition, according to the property of sub-Gaussian, we have the output o_j satisfies

$$
\begin{aligned}
\mathrm{Var}[o_j] =& \mathrm{Var}[z_j - \Delta_2] \\
=& \mathrm{Var}\left[\frac{2\Delta_2}{g|U|}\left(\sum_{u\in U}(\bar{w}_{u,j} + \gamma_1 + \gamma_2)\right)\right] \\
=& \mathrm{Var}\left[\frac{2\Delta_2}{g|U|}\left(\sum_{u\in U}(\gamma_1 + \gamma_2)\right)\right] \\
\leqslant& \frac{4\Delta_2^2}{g^2|U|^2}\left(\frac{|U|}{4} + \frac{b|U|}{4}\right) \\
\leqslant& \frac{\Delta_2^2}{g^2|U|^2}\left(|U| + |U|\cdot(\frac{180g^2\ln(4s/\delta)}{\hat{\varepsilon}^2|U|} + 1)\right) \\
=& \frac{\Delta_2^2}{g^2|U|^2}\left(2|U| + \frac{180g^2\ln(4s/\delta)}{\hat{\varepsilon}^2}\right) \\
\overset{(a)}{\leqslant}& \frac{180\Delta_2^2\ln(4s/\delta)}{|U|^2\hat{\varepsilon}^2} + \frac{180\Delta_2^2\ln(4s/\delta)}{|U|^2\hat{\varepsilon}^2} \\
=& \frac{360\Delta_2^2\ln(4s/\delta)}{|U|^2\hat{\varepsilon}^2} = O\left(\frac{\Delta_2^2\ln^2(s/\delta)}{|U|^2\varepsilon}\right),
\end{aligned}
$$

where (a) is derived from our choice of g in Eq. (D.55). The output o_j is $O\left(\frac{\Delta_2\ln(s/\delta)}{|U|\varepsilon}\right)$-sub-Gaussian. Then, the output vector $o = \{o_j\}_{j\in[s]}$ is a s-dimensional $O\left(\frac{\Delta_2\ln(s/\delta)}{|U|\varepsilon}\right)$-sub-

Gaussian vector according to the definition of the sub-Gaussian vector. □

Now, we are ready to prove Theorem 4.8.

Proof of Theorem 4.8. From Theorem C.2, we have the shuffle protocol in Algorithm 7 guarantees (ε, δ)-SDP with inputs $\{y_u\}_{u \in U}$. In the DP-DPE algorithm, we apply the shuffle protocol in Algorithm 7 as a PRIVATIZER with inputs $\{\vec{y}_l^u\}_{u \in U_l}$ and $\Delta_2 = \max \|\vec{y}_l^u\|_2 = B\sqrt{s_l}$ in each phase l. By admitting new clients in each phase, the DP-DPE algorithm is (ε, δ)-SDP. □

C.2 Proofs of Theorems in Section 4.6

In Section 4.6, we present the performance analysis of the DP-DPE algoritm in different DP models. In this section, we provide complete proofs for each of the theorems in Section 4.6.

C.2.1 Proof of Theorem 4.9

Before proving Theorem 4.9, we first provide the key concentration under DPE in the following Theorem C.3.

Theorem C.3. *For any phase l, under DPE with $\sigma_n = 0$, the following concentration inequalities hold, for any $x \in \mathcal{D}$,*

$$P\left\{\langle \tilde{\theta}_l - \theta^*, \mathbf{x}\rangle \geqslant W_l\right\} \leqslant 2\beta, \quad \textit{and} \quad P\left\{\langle \theta^* - \tilde{\theta}_l, \mathbf{x}\rangle \geqslant W_l\right\} \leqslant 2\beta. \tag{C.7}$$

Proof. We prove the first concentration inequality in (C.7) in the following, and the second inequality can be proved symmetrically. Note that for any action $x \in \mathcal{D}$, the gap between

the estimated reward with parameter $\tilde{\theta}_l$ and the true reward with θ^* satisfies

$$\langle \tilde{\theta}_l - \theta^*, \mathbf{x} \rangle = \left\langle \tilde{\theta}_l - \frac{1}{|U_l|} \sum_{u \in U_l} \theta_u + \frac{1}{|U_l|} \sum_{u \in U_l} \theta_u - \theta^*, x \right\rangle = \left\langle \tilde{\theta}_l - \frac{1}{|U_l|} \sum_{u \in U_l} \theta_u, x \right\rangle + \frac{1}{|U_l|} \sum_{u \in U_l} \langle \theta_u - \theta^*, x \rangle .$$

Then, with $\sigma_n = 0$, we have

$$\begin{aligned}
& P\left\{ \langle \tilde{\theta}_l - \theta^*, \mathbf{x} \rangle \geqslant W_l \right\} \\
=& P\left\{ \left\langle \tilde{\theta}_l - \frac{1}{|U_l|} \sum_{u \in U_l} \theta_u, x \right\rangle + \frac{1}{|U_l|} \sum_{u \in U_l} \langle \theta_u - \theta^*, \mathbf{x} \rangle \geqslant W_l \right\} \\
=& P\left\{ \left\langle \tilde{\theta}_l - \frac{1}{|U_l|} \sum_{u \in U_l} \theta_u, x \right\rangle + \frac{1}{|U_l|} \sum_{u \in U_l} \langle \theta_u - \theta^*, \mathbf{x} \rangle \geqslant \sqrt{\frac{4d}{h_l |U_l|} \log\left(\frac{1}{\beta}\right)} + \sqrt{\frac{2\sigma^2}{|U_l|} \log\left(\frac{1}{\beta}\right)} \right\} \\
\leqslant& P\left\{ \left\langle \tilde{\theta}_l - \frac{1}{|U_l|} \sum_{u \in U_l} \theta_u, x \right\rangle \geqslant \sqrt{\frac{4d}{h_l |U_l|} \log\left(\frac{1}{\beta}\right)} \right\} + P\left\{ \frac{1}{|U_l|} \sum_{u \in U_l} \langle \theta_u - \theta^*, \mathbf{x} \rangle \geqslant \sqrt{\frac{2\sigma^2}{|U_l|} \log\left(\frac{1}{\beta}\right)} \right\} .
\end{aligned} \tag{C.8}$$

In the following, we try to bound the above two terms, respectively. Under the non-private DP-DPE algorithm, the output of $\mathcal{P}$ is the exact average of local performance, i.e., $\tilde{y}_l =$

$\mathcal{P}\left(\{\vec{y}_l^u\}_{u\in U_l}\right) = \frac{1}{|U_l|}\sum_{u\in U_l}\vec{y}_l^u$. Then, the estimated model parameter satisfies

$$
\begin{aligned}
\tilde{\theta}_l &= V_l^{-1}G_l \\
&= V_l^{-1}\sum_{x\in\mathrm{supp}(\pi_l)} xT_l(\mathbf{x})\tilde{y}_l(\mathbf{x}) \\
&= V_l^{-1}\sum_{x\in\mathrm{supp}(\pi_l)} xT_l(\mathbf{x})\frac{1}{|U_l|}\sum_{u\in U_l} y_l^u(\mathbf{x}) \\
&= \frac{1}{|U_l|}\sum_{u\in U_l} V_l^{-1}\sum_{x\in\mathrm{supp}(\pi_l)} xT_l(\mathbf{x})y_l^u(\mathbf{x}) \\
&= \frac{1}{|U_l|}\sum_{u\in U_l} V_l^{-1}\sum_{x\in\mathrm{supp}(\pi_l)} x\sum_{t\in\mathcal{T}_l(\mathbf{x})}(x^\top\theta_u+\eta_{u,t}) \\
&= \frac{1}{|U_l|}\sum_{u\in U_l} V_l^{-1}\left(\sum_{x\in\mathrm{supp}(\pi_l)} T_l(\mathbf{x})xx^\top\theta_u + \sum_{x\in\mathrm{supp}(\pi_l)}\sum_{t\in\mathcal{T}_l(\mathbf{x})}\eta_{u,t}x\right) \\
&= \frac{1}{|U_l|}\sum_{u\in U_l} V_l^{-1}\left(V_l\theta_u + \sum_{t\in\mathcal{T}_l}\eta_{u,t}x_t\right) \\
&= \frac{1}{|U_l|}\sum_{u\in U_l}\theta_u + \frac{1}{|U_l|}\sum_{u\in U_l} V^{-1}\sum_{t\in\mathcal{T}_l}\eta_{u,t}x_t.
\end{aligned}
\tag{C.9}
$$

For any x in $\mathcal{D}$,

$$
\left\langle x, V_l^{-1}\sum_{t\in\mathcal{T}_l}\eta_{u,t}x_t\right\rangle = \sum_{t\in\mathcal{T}_l}\langle x, V_l^{-1}x_t\rangle\eta_{u,t}. \tag{C.10}
$$

Note that $\eta_{u,t}$ is *i.i.d.* 1-sub-Gaussian over t and that the chosen action x_t at t is deterministic in the l-th phase under the DP-DPE algorithm. Combining the following result,

$$
\sum_{t\in\mathcal{T}_l}\langle x, V_l^{-1}x_t\rangle^2 = x^\top V_l^{-1}\left(\sum_{t\in\mathcal{T}_l}x_tx_t^\top\right)V_l^{-1}x = \|x\|^2_{V_l^{-1}},
$$

where the second equality is due to $V_l = \sum_{t\in\mathcal{T}_l}x_tx_t^\top$, we derive that the LHS of Eq. (C.10) is $\|x\|_{V_l^{-1}}$-sub-Gaussian. Besides, we have $\|x\|^2_{V_l^{-1}} \leqslant \frac{\|x\|^2_{V_l(\pi_l)^{-1}}}{h_l} \leqslant \frac{g(\pi_l)}{h_l} \leqslant \frac{2d}{h_l}$ by the near-G-optimal design. According to the property of a sub-Gaussian random variable, we can

obtain

$$P\left\{\frac{1}{|U_l|}\sum_{u\in U_l}\left\langle x, V_l^{-1}\sum_{t\in\mathcal{T}_l}\eta_{u,t}x_t\right\rangle \geqslant \sqrt{\frac{4d}{h_l|U_l|}\log\left(\frac{1}{\beta}\right)}\right\} \leqslant \exp\left\{-\frac{|U_l|\frac{4d}{h_l|U_l|}\log(1/\beta)}{2\cdot\frac{2d}{h_l}}\right\} = \beta. \tag{C.11}$$

Combining the result in Eq. (C.9), we have

$$\begin{aligned}
&P\left\{\left\langle \tilde{\theta}_l - \frac{1}{|U_l|}\sum_{u\in U_l}\theta_u, x\right\rangle \geqslant \sqrt{\frac{4d}{h_l|U_l|}\log\left(\frac{1}{\beta}\right)}\right\}\\
=&P\left\{\frac{1}{|U_l|}\sum_{u\in U_l}\left\langle x, V_l^{-1}\sum_{t\in\mathcal{T}_l}\eta_{u,t}x_t\right\rangle \geqslant \sqrt{\frac{4d}{h_l|U_l|}\log\left(\frac{1}{\beta}\right)}\right\} \leqslant \beta.
\end{aligned}$$

For the second term of Eq. (D.25), we know that $\langle\theta_u-\theta^*,\mathbf{x}\rangle = \langle\xi_u, x\rangle$ is $\|x\|_2\sigma$-sub-Gaussian. Similarly, according to the sub-Gaussian property, we have

$$P\left\{\frac{1}{|U_l|}\sum_{u\in U_l}\langle\theta_u-\theta^*,\mathbf{x}\rangle \geqslant \sqrt{\frac{2\sigma^2}{|U_l|}\log\left(\frac{1}{\beta}\right)}\right\} \leqslant \exp\left\{-\frac{|U_l|\cdot\frac{2\sigma^2}{|U_l|}\log(\frac{1}{\beta})}{2\sigma^2\|x\|_2^2}\right\} \leqslant \beta. \tag{C.12}$$

Therefore, we have

$$P\left\{\langle\tilde{\theta}_l-\theta^*,\mathbf{x}\rangle \geqslant W_l\right\} \leqslant 2\beta.$$

The symmetrical argument completes the proof. □

To prove Theorem 4.9, we first present two observations with high probability based on Theorem C.3, then analyze the regret in a particular phase $l > 2$, and finally combine all phases to get the total regret.

Under DPE algorithm, define a "good" event at l-th phase as $\mathcal{E}_l$:

$$\mathcal{E}_l \triangleq \left\{\langle\theta^*-\tilde{\theta}_l, x^*\rangle \leqslant W_l \quad \text{and} \quad \forall x\in\mathcal{D}\backslash\{x^*\} \quad \langle\tilde{\theta}_l-\theta^*,\mathbf{x}\rangle \leqslant W_l\right\}.$$

According to Theorem C.3, it is not difficult to derive $P(\mathcal{E}_l) \geqslant 1 - 2k\beta$ via union bound. In addition, under event $\mathcal{E}_l$, we have the following two observations:

1. If the optimal action $x^* \in \mathcal{D}_l$, then $x^* \in \mathcal{D}_{l+1}$.

2. For any $x \in \mathcal{D}_{l+1}$, we have $\langle \theta^*, x^* - \mathbf{x} \rangle \leqslant 4W_l$.

Proof. **Observation 1:** Let $b \in \operatorname{argmax}_{x \in \mathcal{D}_l} \langle \tilde{\theta}_l, x \rangle$. If $x^* = b$, then $x^* \in \mathcal{D}_{l+1}$ according to the elimination step in Algorithm 5. If $x^* \neq b$, then under event $\mathcal{E}_l$, we have

$$\begin{aligned} \langle \tilde{\theta}_l, b - x^* \rangle &= \langle \tilde{\theta}_l, b \rangle - \langle \tilde{\theta}_l, x^* \rangle \\ &\leqslant \langle \theta^*, b \rangle + W_l - \langle \theta^*, x^* \rangle + W_l \\ &= \langle \theta^*, b - x^* \rangle + 2W_l \\ &\leqslant 2W_l, \end{aligned} \tag{C.13}$$

which means that x^* is not eliminated at the end of the l-th phase, i.e., $x^* \in \mathcal{D}_{l+1}$.

Observation 2: For any $x \in \mathcal{D}_{l+1}$, we have $\langle \tilde{\theta}_l, b - \mathbf{x} \rangle \leqslant 2W_l$. Then, we have the following steps:

$$\begin{aligned} 2W_l &\geqslant \langle \tilde{\theta}_l, b - \mathbf{x} \rangle \\ &\geqslant \langle \tilde{\theta}_l, x^* - \mathbf{x} \rangle \\ &\geqslant \langle \theta^*, x^* \rangle - W_l - \langle \theta^*, \mathbf{x} \rangle - W_l \\ &= \langle \theta^*, x^* - \mathbf{x} \rangle - 2W_l, \end{aligned} \tag{C.14}$$

where the second inequality is from Observation 1. Then, we derive Observation 2. □

Now, we are ready to prove Theorem 4.9.

Proof. **1) Regret in a specific phase $l \geqslant 2$.** Let r_l denote the incurred regret in the l-th phase, i.e., $r_l \triangleq \sum_{t \in \mathcal{T}_l} \langle \theta^*, x^* - x_t \rangle$. For any phase $l = 1, \ldots, L-1$, under event $\mathcal{E}_l$, we have

the following result

$$
\begin{aligned}
r_{l+1} &= \sum_{t\in\mathcal{T}_{l+1}} \langle \theta^*, x^* - x_t\rangle \\
&\leqslant \sum_{t\in\mathcal{T}_{l+1}} 4W_l \\
&= \sum_{t\in\mathcal{T}_{l+1}} 4\left(\sqrt{\frac{4d}{h_l|U_l|}\log\left(\frac{1}{\beta}\right)} + \sqrt{\frac{2\sigma^2}{|U_l|}\log\left(\frac{1}{\beta}\right)}\right) \\
&= \underbrace{4T_{l+1}\sqrt{\frac{4d}{h_l|U_l|}\log\left(\frac{1}{\beta}\right)}}_{①} + \underbrace{4T_{l+1}\sqrt{\frac{2\sigma^2}{|U_l|}\log\left(\frac{1}{\beta}\right)}}_{②}.
\end{aligned}
\tag{C.15}
$$

We derive an upper bound for each of the two terms in the above equation. Note that the total number of pulls in the $(l+1)$-th phase is $T_{l+1} = \sum_{x\in\mathrm{supp}(\pi_{l+1})} T_{l+1}(\mathbf{x})$, which satisfies

$$
h_1\cdot 2^l = h_{l+1} \leqslant T_{l+1} \leqslant h_{l+1} + |\mathrm{supp}(\pi_{l+1})| \leqslant h_1\cdot 2^l + S,
$$

where $S \triangleq 4d\log\log d + 16 \geqslant |\mathrm{supp}(\pi_{l+1})|$. In addition, we have $h_l = h_1\cdot 2^{l-1}$ and $2^{\alpha l} \leqslant |U_l| \leqslant 2^{\alpha l}+1$. Then, for ①, we have

$$
\begin{aligned}
① &\leqslant 4(h_1\cdot 2^l + S)\sqrt{\frac{8d}{h_1\cdot 2^{(1+\alpha)l}}\log\left(\frac{1}{\beta}\right)} \\
&= 8\sqrt{2d\log\left(\frac{1}{\beta}\right)}\left(\sqrt{h_1\cdot 2^{(1-\alpha)l}} + \frac{S}{\sqrt{h_1\cdot 2^{(\alpha+1)l}}}\right).
\end{aligned}
\tag{C.16}
$$

As to the second term ②, we have

$$
\begin{aligned}
② &\leqslant 4(h_1\cdot 2^l + S)\sqrt{\frac{2\sigma^2}{2^{\alpha l}}\log\left(\frac{1}{\beta}\right)} \\
&= 4\sigma\sqrt{2\log\left(\frac{1}{\beta}\right)}\left(h_1\sqrt{2^{(2-\alpha)l}} + \frac{S}{\sqrt{2^{\alpha l}}}\right).
\end{aligned}
\tag{C.17}
$$

Then, for any $l = 2, \cdots, L$, the regret in the l-th phase r_l is upper bounded by

$$\begin{aligned} r_l \leqslant & 8\sqrt{2d\log\left(\frac{1}{\beta}\right)}\left(\sqrt{h_1 \cdot 2^{(1-\alpha)(l-1)}} + \frac{S}{\sqrt{h_1 \cdot 2^{(\alpha+1)(l-1)}}}\right) \\ & + 4\sigma\sqrt{2\log\left(\frac{1}{\beta}\right)}\left(h_1\sqrt{2^{(2-\alpha)(l-1)}} + \frac{S}{\sqrt{2^{\alpha(l-1)}}}\right). \end{aligned} \tag{C.18}$$

2) Total regret:

Define $\mathcal{E}_g$ as the event where the "good" event occurs in every phase, i.e., $\mathcal{E}_g \triangleq \bigcap_{l=1}^{L} \mathcal{E}_l$. Based on to Theorem C.3, it is not difficult to obtain $P\{\mathcal{E}_g\} \geqslant 1 - 2k\beta L$ by applying union bound. At the same time, let R_g be the regret under event $\mathcal{E}_g$, and R_b be the regret if event $\mathcal{E}_g$ does not hold. Then, the expected total regret in T is $\mathbb{E}[R(T)] = P(\mathcal{E}_g)R_g + (1 - P(\mathcal{E}_g))R_b$.

Under event $\mathcal{E}_g$, the regret in the l-th phase r_l satisfies Eq. (C.18) for any $l \geqslant 2$. Combining $r_1 \leqslant 2T_1 \leqslant 2(h_1 + S)$ (since $\langle \theta^*, x^* - x\rangle \leqslant 2$ for all $\mathbf{x} \in \mathcal{D}$), we have

$$\begin{aligned} R_g = & \sum_{l=1}^{L} r_l \\ \leqslant & 2(h_1 + S) + \sum_{l=2}^{L} 8\sqrt{2d\log\left(\frac{1}{\beta}\right)}\left(\sqrt{h_1 \cdot 2^{(1-\alpha)(l-1)}} + \frac{S}{\sqrt{h_1 \cdot 2^{(\alpha+1)(l-1)}}}\right) \\ & + \sum_{l=1}^{L} 4\sigma\sqrt{2\log\left(\frac{1}{\beta}\right)}\left(h_1\sqrt{2^{(2-\alpha)(l-1)}} + \frac{S}{\sqrt{2^{\alpha(l-1)}}}\right) \\ \leqslant & 2(h_1 + S) + 8\sqrt{2d\log(1/\beta)}\left(C_0\sqrt{h_1 \cdot 2^{(L-1)(1-\alpha)}} + \frac{S}{\sqrt{h_1}(\sqrt{2^{1+\alpha}} - 1)}\right) \\ & + 4\sigma\sqrt{2\log(1/\beta)}\left(\frac{h_1\sqrt{2^{2-\alpha}}}{\sqrt{2^{2-\alpha}} - 1} \cdot \sqrt{2^{(L-1)(2-\alpha)}} + C_1 S\right) \\ = & 2(h_1 + S) + 8\sqrt{2d\log(1/\beta)} \cdot C_0\sqrt{h_1 2^{(L-1)(1-\alpha)}} + \frac{8S\sqrt{2d\log(1/\beta)}}{\sqrt{h_1}(\sqrt{2} - 1)} \\ & + 4\sigma\sqrt{2\log(1/\beta)}\left(4h_1\sqrt{2^{(L-1)(2-\alpha)}} + C_1 S\right), \end{aligned} \tag{C.19}$$

where $C_0 = \frac{\sqrt{2^{1-\alpha}}}{\sqrt{2^{1-\alpha}}-1}$ and $C_1 = \sum_{l=2}^{\infty} \frac{1}{\sqrt{2^{\alpha(l-1)}}}$. Note that $h_L \leqslant T_L \leqslant T$, which indicates $2^{L-1} \leqslant T/h_1$, and $L \leqslant \log(2T/h_1)$. Then, the above inequality becomes

$$
\begin{aligned}
R_g &= \sum_{l=1}^{L} r_l \\
&\leqslant 2(h_1+S) + 8\sqrt{2d\log(1/\beta)} \cdot C_0 h_1 (\sqrt{T/h_1})^{1-\alpha} + 20S\sqrt{2d/h_1 \log(1/\beta)} \\
&+ 4\sigma\sqrt{2\log(1/\beta)}\left(4h_1\sqrt{(T/h_1)^{2-\alpha}} + C_1 S\right) \\
&\leqslant 2(h_1+S) + 8C_0\sqrt{2dh_1^{\alpha}T^{1-\alpha}\log(1/\beta)} + 20S\sqrt{2d/h_1\log(1/\beta)} \\
&+ 16\sigma\sqrt{2h_1^{\alpha}\log(1/\beta)} \cdot T^{1-\alpha/2} + 4C_1\sigma S\sqrt{2\log(1/\beta)}.
\end{aligned}
$$

On the other hand, $R_b \leqslant 2T$ since $\langle \theta^*, x^* - x\rangle \leqslant 2$ for all $\mathbf{x} \in \mathcal{D}$. Choose $\beta = \frac{1}{kT}$ in Algorithm 5. Finally, we have the following results:

$$
\begin{aligned}
&\mathbb{E}[R(T)] \\
&= P(\mathcal{E}_g)R_g + (1-P(\mathcal{E}_g))R_b \\
&\leqslant R_g + 2k\beta L \cdot 2T \\
&\leqslant 2(h_1+S) + 8C_0\sqrt{2dh_1^{\alpha}T^{1-\alpha}\log(kT)} + 20S\sqrt{2d/h_1\log(kT)} \\
&+ 16\sigma\sqrt{2h_1^{\alpha}\log(kT)} \cdot T^{1-\alpha/2} + 4C_1\sigma S\sqrt{2\log(kT)} + 4\log(2T/h_1) \\
&= O(\sqrt{dT^{1-\alpha}\log(kT)}) + O(\sigma T^{1-\alpha/2}\sqrt{\log(kT)}) + O(d^{3/2}\sqrt{\log(kT)}),
\end{aligned}
$$

where the last equality is from $h_1 = 2$ and ignoring the logarithmic term regarding d in S.

3) Communication cost. Notice that the communicating data in each phase is the local average performance $y_l^u(\mathbf{x})$ for each chosen action x in the support set $\text{supp}(\pi_l)$. Therefore, the total communication cost is

$$
C(T) = \sum_{l=1}^{L} s_l|U_l| \leqslant \sum_{l=1}^{L}(4d\log\log d + 16)\cdot 2^{\alpha l} = O(dT^{\alpha}).
$$

Table C.1: Setting

Algorithm	σ_n	Notes
DPE	0	–
CDP-DPE	$\sigma_n = 2\sigma_{nc}\sqrt{Sd}$	$\sigma_{nc} = \frac{2B\sqrt{2s_l \ln(1.25/\delta)}}{\varepsilon\lvert U_l\rvert}$
LDP-DPE	$\sigma_n = 2\sigma_{nl}\sqrt{\frac{Sd}{\lvert U_l\rvert}}$	$\sigma_{nl} = \frac{2B\sqrt{2s_l \ln(1.25/\delta)}}{\varepsilon}$
SDP-DPE	$\sigma_n = 2\sigma_{ns}\sqrt{Sd}$	$\sigma_{ns} = O\left(\frac{B\sqrt{d}\ln(d/\delta)}{\varepsilon\lvert U_l\rvert}\right)$

□

C.2.2 Proof of Theorems 4.11, D.10, and D.11

To be clear, we list the parameter setting of Theorems 4.11, D.10, and D.11 in Table C.1. To prove the three theorems, we first prove the following concentration inequalities hold for the DP-DPE algorithm under the three DP models with the setting in Table C.1.

Theorem C.4. *Set DP-DPE in the central, local, and shuffle models (i.e., CDP-DPE, LDP-DPE, and SDP-DPE, respectively) based on Table C.1. In any phase l, all of CDP-DPE, LDP-DPE, and SDP-DPE satisfy the following concentration, for any $x \in \mathcal{D}_l$,*

$$P\left\{\langle \tilde{\theta}_l - \theta^*, \mathbf{x}\rangle \geqslant W_l\right\} \leqslant 3\beta \quad \textit{and} \quad P\left\{\langle \theta^* - \tilde{\theta}_l, \mathbf{x}\rangle \geqslant W_l\right\} \leqslant 3\beta. \tag{C.20}$$

Proof. We prove the first concentration inequality in (D.63) for CDP-DPE, LDP-DPE, and SDP-DPE, respectively, in the following, and the second inequality can be proved symmetrically.

According to Line 17 in Algorithm 5, we have

$$
\begin{aligned}
\tilde{\theta}_l &= V_l^{-1} G_l \\
&= V_l^{-1} \sum_{x \in \mathrm{supp}(\pi_l)} x T_l(\mathbf{x}) \tilde{y}_l(\mathbf{x}) \\
&= V_l^{-1} \sum_{x \in \mathrm{supp}(\pi_l)} x T_l(\mathbf{x}) \frac{1}{|U_l|} \sum_{u \in U_l} y_l^u(\mathbf{x}) + V_l^{-1} \sum_{x \in \mathrm{supp}(\pi_l)} x T_l(\mathbf{x}) \underbrace{\left(\tilde{y}_l(\mathbf{x}) - \frac{1}{|U_l|} \sum_{u \in U_l} y_l^u(\mathbf{x}) \right)}_{\gamma_p(\mathbf{x})} \\
&\overset{(a)}{=} V_l^{-1} \sum_{x \in \mathrm{supp}(\pi_l)} x T_l(\mathbf{x}) \frac{1}{|U_l|} \sum_{u \in U_l} y_l^u(\mathbf{x}) + V_l^{-1} \sum_{x \in \mathrm{supp}(\pi_l)} x T_l(\mathbf{x}) \gamma_p(\mathbf{x}) \\
&\overset{(b)}{=} \frac{1}{|U_l|} \sum_{u \in U_l} \theta_u + \frac{1}{|U_l|} \sum_{u \in U_l} V^{-1} \sum_{t \in \mathcal{T}_l} \eta_{u,t} x_t + V_l^{-1} \sum_{x \in \mathrm{supp}(\pi_l)} x T_l(\mathbf{x}) \gamma_p(\mathbf{x}),
\end{aligned}
$$

where we denote the noise introduced for privacy preserving associated with action x by $\gamma_p(\mathbf{x}) \triangleq \tilde{y}_l(\mathbf{x}) - \frac{1}{|U_l|} \sum_{u \in U_l} y_l^u(\mathbf{x})$, which varies according to the specified DP models, and (b) is derived from Eq. (C.9). Then, for any action $x' \in \mathcal{D}$, the gap between the estimated reward with parameter $\tilde{\theta}_l$ and the true reward with θ^* satisfies

$$
\begin{aligned}
&\langle \tilde{\theta}_l - \theta^*, x' \rangle \\
&= \left\langle \tilde{\theta}_l - \frac{1}{|U_l|} \sum_{u \in U_l} \theta_u + \frac{1}{|U_l|} \sum_{u \in U_l} \theta_u - \theta^*, x' \right\rangle \\
&= \left\langle \tilde{\theta}_l - \frac{1}{|U_l|} \sum_{u \in U_l} \theta_u, x' \right\rangle + \frac{1}{|U_l|} \sum_{u \in U_l} \langle \theta_u - \theta^*, x' \rangle \\
&= \frac{1}{|U_l|} \sum_{u \in U_l} \left\langle V_l^{-1} \sum_{t \in \mathcal{T}_l} \eta_{u,t} x_t, x' \right\rangle + \left\langle V_l^{-1} \sum_{x \in \mathrm{supp}(\pi_l)} x T_l(\mathbf{x}) \gamma_p(\mathbf{x}), x' \right\rangle + \frac{1}{|U_l|} \sum_{u \in U_l} \langle \theta_u - \theta^*, x' \rangle .
\end{aligned}
$$

Then, we have

$$
\begin{aligned}
&P\left\{\langle\tilde{\theta}_l-\theta^*,\mathbf{x}\rangle\geqslant W_l\right\}\\
=&P\left\{\frac{1}{|U_l|}\sum_{u\in U_l}\left\langle V_l^{-1}\sum_{t\in\mathcal{T}_l}\eta_{u,t}x_t,x'\right\rangle+\left\langle V_l^{-1}\sum_{x\in\text{supp}(\pi_l)}xT_l(\mathbf{x})\gamma_{j_x},x'\right\rangle+\frac{1}{|U_l|}\sum_{u\in U_l}\langle\theta_u-\theta^*,x'\rangle\geqslant W_l\right\}\\
\leqslant&P\left\{\frac{1}{|U_l|}\sum_{u\in U_l}\left\langle x',V_l^{-1}\sum_{t\in\mathcal{T}_l}\eta_{u,t}x_t\right\rangle\geqslant\sqrt{\frac{4d}{h_l|U_l|}\log\left(\frac{1}{\beta}\right)}\right\}\\
&+P\left\{\frac{1}{|U_l|}\sum_{u\in U_l}\langle\theta_u-\theta^*,\mathbf{x}\rangle\geqslant\sqrt{\frac{2\sigma^2}{|U_l|}\log\left(\frac{1}{\beta}\right)}\right\}\\
&+P\left\{\left\langle x',V_l^{-1}\sum_{x\in\text{supp}(\pi_l)}xT_l(\mathbf{x})\gamma_p(\mathbf{x})\right\rangle\geqslant\sqrt{2\sigma_n^2\log\left(\frac{1}{\beta}\right)}\right\}.
\end{aligned}
\tag{C.21}
$$

We have shown that either the first or the second probability in the above equation is less than β in Eq. (C.11) and Eq. (C.12), respectively. In the following, we try to show that the third term is less than β under each of the three DP models.

i) CDP-DPE. Under the DP-DPE algorithm with the central model PRIVATIZER, the output of the PRIVATIZER $\mathcal{P}$ is, $\tilde{y}_l=\frac{1}{|U_l|}\sum_{u\in U_l}\vec{y}_l^u+(\gamma_1,\ldots,\gamma_{s_l})$, where $\gamma_j\overset{i.i.d.}{\sim}\mathcal{N}(0,\sigma_{nc}^2)$. Then, $\gamma_p(\mathbf{x})\sim\mathcal{N}(0,\sigma_{nc}^2)$ in the central model and is *i.i.d.* across actions $x\in\text{supp}(\pi_l)$. Note that

$$
\left\langle x',V_l^{-1}\sum_{x\in\text{supp}(\pi_l)}xT_l(\mathbf{x})\gamma_p(\mathbf{x})\right\rangle=\sum_{x\in\text{supp}(\pi_l)}\left\langle x',V_l^{-1}x\right\rangle T_l(\mathbf{x})\gamma_p(\mathbf{x}),\tag{C.22}
$$

and $\gamma_p(\mathbf{x})\overset{i.i.d.}{\sim}\mathcal{N}(0,\sigma_{nc}^2)$. The variance (denoted by σ_{sum}^2) of the above sum of *i.i.d.* Gaussian variables is

$$
\sigma_{\text{sum}}^2=\sum_{x\in\text{supp}(\pi_l)}\left\langle x',V_l^{-1}x\right\rangle^2T_l(\mathbf{x})^2\sigma_{nc}^2\overset{(a)}{\leqslant}T_l\cdot x'^{\top}V_l^{-1}\left(\sum_{x\in\text{supp}(\pi_l)}T_l(\mathbf{x})xx^{\top}\right)V_l^{-1}x'\sigma_{nc}^2=T_l\|x'\|_{V_l^{-1}}^2\sigma_{nc}^2,
$$

where (a) is from $T_l(\mathbf{x}) \leqslant T_l$ for any x in the support set $\text{supp}(\pi_l)$. Therefore, the LHS of Eq. (C.22) is a Gaussian variable with variance

$$\sigma_{\text{sum}}^2 \leqslant T_l \|x'\|_{V_l^{-1}}^2 \sigma_{nc}^2 \leqslant T_l \cdot \frac{2d}{h_l} \cdot \sigma_{nc}^2 \overset{(a)}{\leqslant} \left(1 + \frac{S}{h_l}\right) 2d\sigma_{nc}^2 \leqslant 4Sd\sigma_{nc}^2 = \sigma_n^2,$$

where (a) is due to $T_l = \sum_{x \in \text{supp}(\pi_l)} T_l(\mathbf{x}) \leqslant h_l + |\text{supp}(\pi_l)| \leqslant h_l + S$, and the last step is based on our setting in Table C.1. Combining the tail bound for Gaussian variables, we have

$$P\left\{\left\langle x', V_l^{-1} \sum_{x \in \text{supp}(\pi_l)} x T_l(\mathbf{x}) \gamma_p(\mathbf{x}) \right\rangle \geqslant \sqrt{2\sigma_n^2 \log\left(\frac{1}{\beta}\right)}\right\} \leqslant \exp\left\{-\frac{2\sigma_n^2 \log\left(\frac{1}{\beta}\right)}{2\sigma_{\text{sum}}^2}\right\} \leqslant \beta.$$

Hence, the first concentration inequality in Eq. (D.63) holds for CDP-DEP algorithm.

ii) LDP-DPE. Under the DP-DPE algorithm with the local model Privatizer, the output of $\mathcal{P}$ is, $\tilde{y}_l = \frac{1}{|U_l|} \sum_{u \in U_l} (\vec{y}_l^u + (\gamma_{u,1}, \ldots, \gamma_{u,s_l}))$, where $\gamma_{u,j} \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma_{nl}^2)$. Let j_x denote the index corresponding to the action x in the support set $\text{supp}(\pi_l)$, i.e., $\tilde{y}_l(\mathbf{x}) = \frac{1}{|U_l|} \sum_{u \in U_l} (y_l^u(\mathbf{x}) + \gamma_{u,j_x})$ and $\gamma_p(\mathbf{x}) = \frac{1}{|U_l|} \sum_{u \in U_l} \gamma_{u,j_x}$ in the local model. Then,

$$\left\langle x', V_l^{-1} \sum_{x \in \text{supp}(\pi_l)} x T_l(\mathbf{x}) \gamma_p(\mathbf{x}) \right\rangle = \frac{1}{|U_l|} \sum_{u \in U_l} \sum_{x \in \text{supp}(\pi_l)} \left\langle x', V_l^{-1} x \right\rangle T_l(\mathbf{x}) \gamma_{u,j_x}. \tag{C.23}$$

Consider the sum at client u first, i.e.,

$$\sum_{x \in \text{supp}(\pi_l)} \left\langle x', V_l^{-1} x \right\rangle T_l(\mathbf{x}) \gamma_{u,j_x}. \tag{C.24}$$

Recall that $\gamma_{u,j_x} \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma_{nl}^2)$. The variance (denoted by $\sigma_{u,\text{sum}}^2$) of the above sum of *i.i.d.*

Gaussian variables at client u is

$$\sigma_{u,\text{sum}}^2 = \sum_{x\in\text{supp}(\pi_l)} \left\langle x', V_l^{-1}x\right\rangle^2 T_l(\mathbf{x})^2\sigma_{nl}^2 \overset{(a)}{\leqslant} T_l\cdot x'^\top V_l^{-1}\left(\sum_{x\in\text{supp}(\pi_l)} T_l(\mathbf{x})xx^\top\right)V_l^{-1}x'\sigma_{nl}^2 = T_l\|x'\|_{V_l^{-1}}^2\sigma_{nl}^2,$$

where (a) is from $T_l(\mathbf{x}) \leqslant T_l$ for any x in the support set $\text{supp}(\pi_l)$. Therefore, the term in Eq. (C.24) is a Gaussian variable with variance

$$\sigma_{u,\text{sum}}^2 \leqslant T_l\|x'\|_{V_l^{-1}}^2\sigma_{nl}^2 \leqslant T_l\cdot\frac{2d}{h_l}\cdot\sigma_{nl}^2 \overset{(a)}{\leqslant} \left(1+\frac{S}{h_l}\right)2d\sigma_{nl}^2 \leqslant 4Sd\sigma_{nl}^2 = |U_l|\sigma_n^2,$$

where (a) is due to $T_l = \sum_{x\in\text{supp}(\pi_l)} T_l(\mathbf{x}) \leqslant h_l + |\text{supp}(\pi_l)| \leqslant h_l + S$, and the last step is based on our setting in Table C.1. Combining the tail bound for Gaussian variables, we have

$$P\left\{\frac{1}{|U_l|}\sum_{u\in U_l}\sum_{x\in\text{supp}(\pi_l)}\left\langle x', V_l^{-1}xT_l(\mathbf{x})\gamma_{j_x}\right\rangle \geqslant \sqrt{2\sigma_n^2\log\left(\frac{1}{\beta}\right)}\right\} \leqslant \exp\left\{-\frac{|U_l|2\sigma_n^2\log\left(\frac{1}{\beta}\right)}{2\sigma_{u,\text{sum}}^2}\right\} \leqslant \beta.$$

Finally, based on Eq. (C.23), we have

$$P\left\{\left\langle x', V_l^{-1}\sum_{x\in\text{supp}(\pi_l)} xT_l(\mathbf{x})\gamma_p(\mathbf{x})\right\rangle \geqslant \sqrt{2\sigma_n^2\log\left(\frac{1}{\beta}\right)}\right\} \leqslant \beta.$$

Hence, the first concentration inequality in Eq. (D.63) holds for LDP-DEP algorithm.

iii) SDP-DPE. Under the DP-DPE algorithm with the shuffle model PRIVATIZER, the output of $\mathcal{P}$ is, $\tilde{y}_l = (\mathcal{A}\circ\mathcal{S}\circ\mathcal{R}^{|U_l|})(\{\vec{y}_l^u\}_{u\in U_l}) = \mathcal{A}(\mathcal{S}(\{\mathcal{R}(\vec{y}_l^u)\}_{u\in U_l}))$, where $\mathcal{A},\mathcal{S}$ and $\mathcal{R}$ follow Algorithm 7. From Theorem C.2, we know that the output of $(\mathcal{A}\circ\mathcal{S}\circ\mathcal{R}^{|U_l|})(\{\vec{y}_l^u\}_{u\in U_l})$ is an unbiased estimator of the average of the $|U_l|$ input vectors $\{\vec{y}_l^u\}_{u\in U_l}$ and that the error distribution is sub-Gaussian with variance $\sigma_{ns}^2 = O\left(\frac{B^2 s_l\ln^2(s_l/\delta)}{\varepsilon^2|U_l|^2}\right)$. Then, $\gamma_p(\mathbf{x}) = \tilde{y}_l(\mathbf{x}) - \frac{1}{|U_l|}\sum_{u\in U_l} y_l^u(\mathbf{x})$ is σ_{ns}-sub-Gaussian with $\mathbb{E}[\gamma_p(\mathbf{x})] = 0$ in the shuffle model. Besides, $\gamma_p(\mathbf{x})$ is *i.i.d.* over each coordinate corresponding to each action x in the support set $\text{supp}(\pi_l)$.

Note that

$$\left\langle x', V_l^{-1} \sum_{x \in \text{supp}(\pi_l)} x T_l(\mathbf{x}) \gamma_p(\mathbf{x}) \right\rangle = \sum_{x \in \text{supp}(\pi_l)} \left\langle x', V_l^{-1} x \right\rangle T_l(\mathbf{x}) \gamma_p(\mathbf{x}). \tag{C.25}$$

The variance (denoted by $\sigma^2_{\text{sub-G}}$) of the above sum of *i.i.d.* sub-Gaussian variables is

$$\sigma^2_{\text{sub-G}} = \sum_{x \in \text{supp}(\pi_l)} \left\langle x', V_l^{-1} x \right\rangle^2 T_l(\mathbf{x})^2 \sigma^2_{ns} \overset{(a)}{\leqslant} T_l \cdot x'^\top V_l^{-1} \left(\sum_{x \in \text{supp}(\pi_l)} T_l(\mathbf{x}) x x^\top \right) V_l^{-1} x' \sigma^2_{ns} = T_l \|x'\|^2_{V_l^{-1}} \sigma^2_{ns},$$

where (a) is from $T_l(\mathbf{x}) \leqslant T_l$ for any x in the support set $\text{supp}(\pi_l)$. Therefore, the LHS of Eq. (C.25) is $\sigma_{\text{sub-G}}$-sub-Gaussian variable with variance proxy

$$\sigma^2_{\text{sub-G}} \leqslant T_l \|x'\|^2_{V_l^{-1}} \sigma^2_{ns} \leqslant T_l \cdot \frac{2d}{h_l} \cdot \sigma^2_{ns} \overset{(a)}{\leqslant} \left(1 + \frac{S}{h_l}\right) 2d\sigma^2_{ns} \leqslant 4Sd\sigma^2_{ns} = \sigma^2_n,$$

where (a) is due to $T_l = \sum_{x \in \text{supp}(\pi_l)} T_l(\mathbf{x}) \leqslant h_l + |\text{supp}(\pi_l)| \leqslant h_l + S$, and the last step is based on our setting in Table C.1. Combining the property for sub-Gaussian variables, we have

$$P\left\{ \left\langle x', V_l^{-1} \sum_{x \in \text{supp}(\pi_l)} x T_l(\mathbf{x}) \gamma_p(\mathbf{x}) \right\rangle \geqslant \sqrt{2\sigma_n^2 \log\left(\frac{1}{\beta}\right)} \right\} \leqslant \exp\left\{ -\frac{2\sigma_n^2 \log\left(\frac{1}{\beta}\right)}{2\sigma^2_{\text{sum}}} \right\} \leqslant \beta.$$

Hence, the first concentration inequality in Eq. (D.63) holds for SDP-DEP algorithm.

With the symmetrical arguments under the three DP models, we derive the results in Eq. (D.63). □

To prove Theorem 4.11, D.10, and D.11, we follow a similar line to the proof of Theorem 4.9 by first analyzing the regret incurred in a particular phase with high probability, then summing up the regret over all phases, and finally analyzing the communication cost associated

with each instantiation of the DP-DPE algorithm.

Before getting into the proof for each DP-DPE instantiation, we present two important observations under the "good" event at l-th phase. Recall the "good" event definition at the l-th phase $\mathcal{E}_l$:

$$\mathcal{E}_l \triangleq \left\{ \langle \theta^* - \tilde{\theta}_l, x^* \rangle \leqslant W_l \quad \text{and} \quad \forall x \in \mathcal{D} \backslash \{x^*\} \quad \langle \tilde{\theta}_l - \theta^*, \mathbf{x} \rangle \leqslant W_l \right\}.$$

According to Theorem C.4, it is not difficult to derive $P(\mathcal{E}_l) \geqslant 1 - 3k\beta$ for the DP-DPE algorithm. In addition, under event $\mathcal{E}_l$, we can still have the following two observations for all the DP-DPE instantiations:

1. If the optimal action $x^* \in \mathcal{D}_l$, then $x^* \in \mathcal{D}_{l+1}$.

2. For any $x \in \mathcal{D}_{l+1}$, we have $\langle \theta^*, x^* - \mathbf{x} \rangle \leqslant 4W_l$.

For any $l = 1, \ldots, L-1$, according to the second observation under event $\mathcal{E}_l$, we have the regret incurred in the $(l+1)$-th phase satisfies

$$\begin{aligned} r_{l+1} &\triangleq \sum_{t \in \mathcal{T}_{l+1}} \langle \theta^*, x^* - x_t \rangle \\ &\leqslant \sum_{t \in \mathcal{T}_{l+1}} 4W_l \\ &= \sum_{t \in \mathcal{T}_{l+1}} 4 \left(\sqrt{\frac{4d}{h_l |U_l|} \log\left(\frac{1}{\beta}\right)} + \sqrt{\frac{2\sigma^2}{|U_l|} \log\left(\frac{1}{\beta}\right)} + \sqrt{2\sigma_n^2 \log\left(\frac{1}{\beta}\right)} \right) \\ &= \underbrace{4T_{l+1} \sqrt{\frac{4d}{h_l |U_l|} \log\left(\frac{1}{\beta}\right)}}_{①} + \underbrace{4T_{l+1} \sqrt{\frac{2\sigma^2}{|U_l|} \log\left(\frac{1}{\beta}\right)}}_{②} + \underbrace{4T_{l+1} \sqrt{2\sigma_n^2 \log\left(\frac{1}{\beta}\right)}}_{③}. \end{aligned} \tag{C.26}$$

We have shown that ① is bounded by

$$① \leqslant 8\sqrt{2d\log\left(\frac{1}{\beta}\right)}\left(\sqrt{h_1\cdot 2^{(1-\alpha)l}}+\frac{S}{\sqrt{h_1\cdot 2^{(\alpha+1)l}}}\right), \tag{C.27}$$

and that ② is bounded by

$$② \leqslant 4\sigma\sqrt{2\log\left(\frac{1}{\beta}\right)}\left(h_1\sqrt{2^{(2-\alpha)l}}+\frac{S}{\sqrt{2^{\alpha l}}}\right). \tag{C.28}$$

Regarding the third term ③, it varies according to different DP models. In the following, we analyze the term ③ in the central, local, and shuffle model respectively.

i) CDP-DPE. In the central model, $\sigma_n = 2\sigma_{nc}\sqrt{Sd}$ where $\sigma_{nc} = \frac{2B\sqrt{2s_l\ln(1.25/\delta)}}{\varepsilon|U_l|}$. Let $\sigma_0 \triangleq \frac{2B\sqrt{2\ln(1/\delta)}}{\varepsilon}$. Then, $\sigma_n = \frac{2\sigma_0\sqrt{s_l dS}}{|U_l|} \leqslant \frac{2\sigma_0 S\sqrt{d}}{|U_l|}$ since $s_l \leqslant S$ for any l. Combining $|U_l| \geqslant 2^{\alpha l}$, we have

$$\begin{aligned} ③ &\leqslant 4(h_1\cdot 2^l+S)\sqrt{\frac{8\sigma_0^2S^2d}{2^{2\alpha l}}\log\left(\frac{1}{\beta}\right)} \\ &\leqslant 8\sigma_0 S\sqrt{2d\log\left(\frac{1}{\beta}\right)}\left(h_1\cdot 2^{(1-\alpha)l}+\frac{S}{2^{\alpha l}}\right). \end{aligned}$$

ii) LDP-DPE. In the local model, $\sigma_n = 2\sigma_{nl}\sqrt{\frac{Sd}{|U_l|}}$ where $\sigma_{nl} = \frac{2B\sqrt{2s_l\ln(1.25/\delta)}}{\varepsilon}$. Substituting σ_0, we have $\sigma_n = \frac{2\sigma_0\sqrt{s_l dS}}{\sqrt{|U_l|}} \leqslant \frac{2\sigma_0 S\sqrt{d}}{\sqrt{|U_l|}}$ and then derive

$$\begin{aligned} ③ &\leqslant 4(h_1\cdot 2^l+S)\sqrt{\frac{8\sigma_0^2S^2d}{2^{\alpha l}}\log\left(\frac{1}{\beta}\right)} \\ &\leqslant 8\sigma_0 S\sqrt{2d\log\left(\frac{1}{\beta}\right)}\left(h_1\cdot\sqrt{2^{(2-\alpha)l}}+\frac{S}{\sqrt{2^{\alpha l}}}\right). \end{aligned}$$

iii) SDP-DPE. In the shuffle model, $\sigma_n = 2\sigma_{ns}\sqrt{Sd}$, where $\sigma_{ns} = O\left(\frac{B\sqrt{s_l}\ln(s_l/\delta)}{\varepsilon|U_l|}\right)$. Let

$\sigma_{ns} = \frac{C_s B\sqrt{s_l}\ln(s_l/\delta)}{\varepsilon|U_l|}$ and $\sigma_0' \triangleq \frac{C_s B\ln(S/\delta)}{\varepsilon}$. Then, we have $\sigma_n \leqslant \frac{2\sigma_0' S\sqrt{d}}{|U_l|}$ and

$$
\begin{aligned}
③ &\leqslant 4(h_1 \cdot 2^l + S)\sqrt{\frac{8{\sigma_0'}^2 S^2 d}{2^{2\alpha l}}\log\left(\frac{1}{\beta}\right)} \\
&\leqslant 8\sigma_0' S\sqrt{2d\log\left(\frac{1}{\beta}\right)}\left(h_1 \cdot 2^{(1-\alpha)l} + \frac{S}{2^{\alpha l}}\right).
\end{aligned}
$$

Proof of Theorem 4.11. (CDP-DPE). Under the "good" event, the regret in the $(l+1)$-th phase satisfies

$$
\begin{aligned}
r_{l+1} &\leqslant ① + ② + ③ \\
&\leqslant 8\sqrt{2d\log\left(\frac{1}{\beta}\right)}\left(\sqrt{h_1 \cdot 2^{(1-\alpha)l}} + \frac{S}{\sqrt{h_1 \cdot 2^{(\alpha+1)l}}}\right) \\
&+ 4\sigma\sqrt{2\log\left(\frac{1}{\beta}\right)}\left(h_1\sqrt{2^{(2-\alpha)l}} + \frac{S}{\sqrt{2^{\alpha l}}}\right) \\
&+ 8\sigma_0 S\sqrt{2d\log\left(\frac{1}{\beta}\right)}\left(h_1 \cdot 2^{(1-\alpha)l} + \frac{S}{2^{\alpha l}}\right).
\end{aligned} \tag{C.29}
$$

Assume the "good" event hold in every phase, i.e., under event $\mathcal{E}_g = \bigcap_{l=1}^{L}\mathcal{E}_l$. We have

$P\{\mathcal{E}_g\} \geqslant 1 - 3k\beta L$ by applying union bound and the total regret R_g under $\mathcal{E}_g$ satisfies

$$
\begin{aligned}
R_g =& \sum_{l=1}^{L} r_l \\
\leqslant& 2(h_1 + S) + \sum_{l=2}^{L} r_l \\
\leqslant& 2(h_1 + S) + \sum_{l=2}^{L} 8\sqrt{2d\log\left(\frac{1}{\beta}\right)}\left(\sqrt{h_1 \cdot 2^{(1-\alpha)(l-1)}} + \frac{S}{\sqrt{h_1 \cdot 2^{(\alpha+1)(l-1)}}}\right) \\
&+ \sum_{l=2}^{L} 4\sigma\sqrt{2\log\left(\frac{1}{\beta}\right)}\left(h_1\sqrt{2^{(2-\alpha)(l-1)}} + \frac{S}{\sqrt{h_1 \cdot 2^{\alpha(l-1)}}}\right) \\
&+ \sum_{l=2}^{L} 8\sigma_0 S\sqrt{2d\log\left(\frac{1}{\beta}\right)}\left(h_1 \cdot 2^{(1-\alpha)(l-1)} + \frac{S}{2^{\alpha(l-1)}}\right) \\
\overset{(a)}{\leqslant}& 2(h_1 + S) + 8\sqrt{2d\log(1/\beta)} \cdot C_0\sqrt{h_1 2^{(L-1)(1-\alpha)}} + \frac{8S\sqrt{2d\log(1/\beta)}}{\sqrt{h_1}(\sqrt{2}-1)} \\
&+ 4\sigma\sqrt{2\log(1/\beta)}\left(4h_1\sqrt{2^{(L-1)(2-\alpha)}} + C_1 S\right) \\
&+ 8\sigma_0 S\sqrt{2d\log(1/\beta)}\left(\frac{h_1 \cdot 2^{(1-\alpha)(L-1)}}{2^{1-\alpha}-1} + C_2 S\right) \\
\overset{(b)}{\leqslant}& 2(h_1 + S) + 8\sqrt{2d\log(1/\beta)} \cdot C_0\sqrt{h_1}(\sqrt{T/h_1})^{1-\alpha} + 20S\sqrt{2d/h_1\log(1/\beta)} \\
&+ 4\sigma\sqrt{2\log(1/\beta)}\left(4h_1\sqrt{(T/h_1)^{2-\alpha}} + C_1 S\right) \\
&+ 8\sigma_0 S\sqrt{2d\log(1/\beta)}\left(\frac{h_1 \cdot (T/h_1)^{1-\alpha}}{2^{1-\alpha}-1} + C_2 S\right) \\
\overset{(c)}{\leqslant}& 2(h_1 + S) + 8C_0\sqrt{2dh_1^{\alpha}T^{1-\alpha}\log(kT)} + 20S\sqrt{2d/h_1\log(kT)} \\
&+ 16\sigma\sqrt{2h_1^{\alpha}\log(kT)}T^{1-\alpha/2} + 4C_1 S\sigma\sqrt{2\log(kT)} \\
&+ 8\sigma_0 h_1^{\alpha} S\sqrt{2d\log(kT)}\left(\frac{T^{1-\alpha}}{2^{1-\alpha}-1} + C_2 S\right) \\
=& O\left(\sqrt{dT^{1-\alpha}\log(kT)}\right) + O\left(\sigma T^{1-\alpha/2}\sqrt{\log(kT)}\right) + O\left(\sigma_0 T^{1-\alpha} S\sqrt{d\log(kT)}\right) + O\left(\sigma_0 S^2\sqrt{d\log(kT)}\right) \\
\overset{(d)}{=}& O\left(\sigma T^{1-\alpha/2}\sqrt{\log(kT)}\right) + O\left(\frac{Bd^{3/2}T^{1-\alpha}\sqrt{\ln(1/\delta)\log(kT)}}{\varepsilon}\right) + O\left(\frac{Bd^{5/2}\sqrt{\ln(1/\delta)\log(kT)}}{\varepsilon}\right),
\end{aligned}
$$

where in (a) $C_0 = \frac{\sqrt{2^{1-\alpha}}}{\sqrt{2^{1-\alpha}-1}}$, $C_1 = \sum_{l=2}^{\infty} \frac{1}{\sqrt{2^{\alpha(l-1)}}}$ and $C_2 = \sum_{l=2}^{\infty} \frac{1}{2^{\alpha(l-1)}}$, (b) is from $h_1 \cdot 2^{L-1} = h_L \leqslant T_L \leqslant T$, (c) is by setting $\beta = \frac{1}{kT}$, and (d) is derived by substituting $\sigma_0 = \frac{2B\sqrt{2\ln(1.25/\delta)}}{\varepsilon}$ and $S = 4d \log\log d + 16$.

Finally, the expected regret of CDP-DPE algorithm is upper bounded by

$$\begin{aligned}
&\mathbb{E}[R(T)] = P(\mathcal{E}_g)R_g + (1 - P(\mathcal{E}_g)) \cdot 2T \\
&\leqslant R_g + 2\beta kL \cdot 2T \\
&= O\left(\sigma T^{1-\alpha/2}\sqrt{\log(kT)}\right) + O\left(\frac{Bd^{3/2}T^{1-\alpha}\sqrt{\ln(1/\delta)\log(kT)}}{\varepsilon}\right) + O\left(\frac{Bd^{5/2}\sqrt{\ln(1/\delta)\log(kT)}}{\varepsilon}\right).
\end{aligned} \tag{C.30}$$

Communication cost. Notice that the communicating data in each phase is the local average performance $y_l^u(\mathbf{x})$ for each chosen action x in the support set $\text{supp}(\pi_l)$. Therefore, the total communication cost is

$$C(T) = \sum_{l=1}^{L} s_l |U_l| \leqslant \sum_{l=1}^{L} (4d \log\log d + 16) \cdot 2^{\alpha l} = O(dT^{\alpha}).$$

□

Proof of Theorem D.10. (LDP-DPE). Under the "good" event, the regret in the $(l+1)$-th phase satisfies

$$\begin{aligned}
r_{l+1} &\leqslant ① + ② + ③ \\
&\leqslant 8\sqrt{2d\log\left(\frac{1}{\beta}\right)}\left(\sqrt{h_1 \cdot 2^{(1-\alpha)l}} + \frac{S}{\sqrt{h_1 \cdot 2^{(\alpha+1)l}}}\right) \\
&+ 4\sigma\sqrt{2\log\left(\frac{1}{\beta}\right)}\left(h_1\sqrt{2^{(2-\alpha)l}} + \frac{S}{\sqrt{2^{\alpha l}}}\right) \\
&+ 8\sigma_0 S\sqrt{2d\log\left(\frac{1}{\beta}\right)}\left(h_1 \cdot \sqrt{2^{(2-\alpha)l}} + \frac{S}{\sqrt{2^{\alpha l}}}\right).
\end{aligned} \tag{C.31}$$

Assume the "good" event hold in every phase, i.e., under event $\mathcal{E}_g = \bigcap_{l=1}^{L} \mathcal{E}_l$. We have $P\{\mathcal{E}_g\} \geqslant 1 - 3k\beta L$ by applying union bound. Then, the total regret satisfies

$$
\begin{aligned}
R_g =& \sum_{l=1}^{L} r_l \\
\leqslant& 2(h_1 + S) + \sum_{l=2}^{L} 8\sqrt{2d\log\left(\frac{1}{\beta}\right)}\left(\sqrt{h_1 \cdot 2^{(1-\alpha)(l-1)}} + \frac{S}{\sqrt{h_1 \cdot 2^{(\alpha+1)(l-1)}}}\right) \\
&+ \sum_{l=2}^{L}(4\sigma + 8\sigma_0 S\sqrt{d})\sqrt{2\log\left(\frac{1}{\beta}\right)}\left(h_1\sqrt{2^{(2-\alpha)(l-1)}} + \frac{S}{\sqrt{2^{\alpha(l-1)}}}\right) \\
\overset{(a)}{\leqslant}& 2(h_1 + S) + 8\sqrt{2d\log(1/\beta)} \cdot C_0\sqrt{h_1 2^{(L-1)(1-\alpha)}} + \frac{8S\sqrt{2d\log(1/\beta)}}{\sqrt{h_1}(\sqrt{2}-1)} \\
&+ (4\sigma + 8\sigma_0 S\sqrt{d})\sqrt{2\log(1/\beta)}\left(4h_1\sqrt{2^{(L-1)(2-\alpha)}} + C_1 S\right) \\
\overset{(b)}{\leqslant}& 2(h_1 + S) + 8\sqrt{2d\log(1/\beta)} \cdot C_0\sqrt{h_1}(\sqrt{T/h_1})^{1-\alpha} + 20S\sqrt{2d/h_1\log(1/\beta)} \\
&+ (4\sigma + 8\sigma_0 S\sqrt{d})\sqrt{2\log(1/\beta)}\left(4h_1\sqrt{(T/h_1)^{2-\alpha}} + C_1 S\right) \\
\overset{(c)}{\leqslant}& 2(h_1 + S) + 8C_0\sqrt{2dh_1^{\alpha}T^{1-\alpha}\log(kT)} + 20S\sqrt{2d/h_1\log(kT)} \\
&+ (16\sigma + 32\sigma_0 S\sqrt{d})\sqrt{2h_1^{\alpha}\log(kT)}T^{1-\alpha/2} + 4C_1 S\sigma\sqrt{2\log(kT)} + 8C_1\sigma_0 S^2\sqrt{2d\log(kT)} \\
=& O\left(\sqrt{dT^{1-\alpha}\log(kT)}\right) + O\left(\sigma T^{1-\alpha/2}\sqrt{\log(kT)}\right) + O\left(\sigma_0 T^{1-\alpha/2}S\sqrt{d\log(kT)}\right) + O\left(\sigma_0 S^2\sqrt{d\log(kT)}\right) \\
\overset{(d)}{=}& O\left(\sigma T^{1-\alpha/2}\sqrt{\log(kT)}\right) + O\left(\frac{Bd^{3/2}T^{1-\alpha/2}\sqrt{\ln(1/\delta)\log(kT)}}{\varepsilon}\right) + O\left(\frac{Bd^{5/2}\sqrt{\ln(1/\delta)\log(kT)}}{\varepsilon}\right),
\end{aligned}
$$

where in (a) $C_0 = \frac{\sqrt{2^{1-\alpha}}}{\sqrt{2^{1-\alpha}}-1}$ and $C_1 = \sum_{l=2}^{\infty}\frac{1}{\sqrt{2^{\alpha(l-1)}}}$, (b) is from $h_1 \cdot 2^{L-1} = h_L \leqslant T_L \leqslant T$, (c) is by setting $\beta = \frac{1}{kT}$, and (d) is derived by substituting $\sigma_0 = \frac{2B\sqrt{2\ln(1.25/\delta)}}{\varepsilon}$ and $S = 4d\log\log d + 16$.

Finally, the expected regret of LDP-DPE algorithm is upper bounded by

$$\begin{aligned}
&\mathbb{E}[R(T)] = P(\mathcal{E}_g)R_g + (1 - P(\mathcal{E}_g)) \cdot 2T \\
&\leqslant R_g + 2\beta kL \cdot 2T \\
&= O\left(\sigma T^{1-\alpha/2}\sqrt{\log(kT)}\right) + O\left(\frac{Bd^{3/2}T^{1-\alpha/2}\sqrt{\ln(1/\delta)\log(kT)}}{\varepsilon}\right) + O\left(\frac{Bd^{5/2}\sqrt{\ln(1/\delta)\log(kT)}}{\varepsilon}\right).
\end{aligned} \tag{C.32}$$

Communication cost. In the local model, the communicating data in each phase is the private local average reward $(y_l^u(\mathbf{x}) + \gamma_{u,j_x})$ for each chosen action x in the support set $\text{supp}(\pi_l)$. It is still an s_l-dimensional vector from each client. Therefore, the total communication cost is

$$C(T) = \sum_{l=1}^{L} s_l|U_l| \leqslant \sum_{l=1}^{L}(4d\log\log d + 16) \cdot 2^{\alpha l} = O(dT^{\alpha}).$$

□

Proof of Theorem D.11. (SDP-DPE). Under the "good" event, the regret in the $(l+1)$-th phase satisfies

$$\begin{aligned}
r_{l+1} &\leqslant ① + ② + ③ \\
&\leqslant 8\sqrt{2d\log\left(\frac{1}{\beta}\right)}\left(\sqrt{h_1 \cdot 2^{(1-\alpha)l}} + \frac{S}{\sqrt{h_1 \cdot 2^{(\alpha+1)l}}}\right) \\
&+ 4\sigma\sqrt{2\log\left(\frac{1}{\beta}\right)}\left(h_1\sqrt{2^{(2-\alpha)l}} + \frac{S}{\sqrt{2^{\alpha l}}}\right) \\
&+ 8\sigma_0' S\sqrt{2d\log\left(\frac{1}{\beta}\right)}\left(h_1 \cdot 2^{(1-\alpha)l} + \frac{S}{2^{\alpha l}}\right).
\end{aligned} \tag{C.33}$$

Assume the "good" event hold in every phase, i.e., under event $\mathcal{E}_g = \bigcap_{l=1}^{L}\mathcal{E}_l$. We have $P\{\mathcal{E}_g\} \geqslant 1 - 3k\beta L$ by applying union bound. Notice that the regret r_{l+1} under SDP-DPE

share the same form as CDP-DPE except replacing σ_0 in CDP-DPE with σ_0'. Hence, we have the total regret satisfies

$$\begin{aligned}R_g =& O\left(\sqrt{dT^{1-\alpha}\log(kT)}\right) + O\left(\sigma T^{1-\alpha/2}\sqrt{\log(kT)}\right) + O\left(\sigma_0' T^{1-\alpha} S\sqrt{d\log(kT)}\right) + O\left(\sigma_0' S^2\sqrt{d\log(kT)}\right)\\ =& O\left(\sigma T^{1-\alpha/2}\sqrt{\log(kT)}\right) + O\left(\frac{Bd^{3/2}T^{1-\alpha}\ln(d/\delta)\sqrt{\log(kT)}}{\varepsilon}\right) + O\left(\frac{Bd^{5/2}\ln(d/\delta)\sqrt{\log(kT)}}{\varepsilon}\right),\end{aligned}$$

where the last step is derived by substituting $\sigma_0' = \frac{C_s B\ln(S/\delta)}{\varepsilon}$ and $S = O(d)$.

Finally, the expected regret of SDP-DPE algorithm is upper bounded by

$$\begin{aligned}&\mathbb{E}[R(T)] = P(\mathcal{E}_g)R_g + (1 - P(\mathcal{E}_g))\cdot 2T\\ &\leqslant R_g + 2\beta kL\cdot 2T\\ &= O\left(\sigma T^{1-\alpha/2}\sqrt{\log(kT)}\right) + O\left(\frac{Bd^{3/2}T^{1-\alpha}\ln(d/\delta)\sqrt{\log(kT)}}{\varepsilon}\right) + O\left(\frac{Bd^{5/2}\ln(d/\delta)\sqrt{\log(kT)}}{\varepsilon}\right).\end{aligned} \tag{C.34}$$

Communication cost. The communicate cost in the shuffle model is slightly different from the central model and the local model because it communicates $(g+b)s_l$ bits from each participating client in the l-th phase instead of an s_l-dimensional real vector. Based on our setting (Eq. (D.55) in Algorithm 7), we have $(g+b) = \max\left\{O\left(\frac{\sqrt{|U_l|}}{\ln(d)}\right), O\left(\sqrt{d} + \frac{d\ln(d)}{|U_l|}\right), O\left(\frac{\ln(d)}{|U_l|}\right)\right\}$. Combining $s_l \leqslant S \approx O(d)$, the total communication cost is

$$\sum_{l=1}^{L}(g+b)s_l|U_l| = O\left(\sum_{l=1}^{L}\frac{2^{\frac{3}{2}\alpha l}S}{\ln(d)}\right) = O\left(dT^{(3/2)\alpha}\right).$$

□

C.3 Differentially Private Linear Bandits

In this section, we consider the standard stochastic linear bandits [84] and provide differentially private algorithms in the central, local, and shuffle DP models.

C.3.1 Model and Algorithmic Framework

Stochastic linear bandits. In the stochastic linear bandits, there is no client-related uncertainty, and any user/client u can provide direct (noisy) reward observations (i.e., $\theta_u = \theta^*$ in our notations). Specifically, at each round t, the learning agent selects an action x_t from the decision set $\mathcal{D} \subseteq \{x \in \mathbb{R}^d : \|x\|_2^2 \leqslant 1\}$ with $|\mathcal{D}| = k$ and receives a reward with mean $\langle \theta^*, x_t \rangle$, where $\theta^* \in \mathbb{R}^d$ with $\|\theta^*\|_2 \leqslant 1$ is unknown to the agent. The goal of the agent is to maximize the cumulative reward in T rounds by selecting x_t sequentially. Without knowing θ^*, the agent learns it gradually by collecting the noisy reward observation $y_t = \langle \theta^*, x_t \rangle + \eta_t$ at each $t \in [T]$ from a client. The noise η_t is assume to be conditionally 1-sub-Gaussian and *i.i.d.* over time. Moreover, we assume that the reward observations are bounded, i.e., $|y_t| \leqslant B$ for all $t \in [T]$. Let $x^* \in \text{argmax}_{x \in \mathcal{D}} \langle \theta^*, \mathbf{x} \rangle$ be an optimal action. Then, the objective of maximizing the cumulative reward is equivalent to minimizing the regret defined as follows:

$$R(T) \triangleq T\langle \theta^*, x^* \rangle - \sum_{t=1}^{T} \langle \theta^*, x_t \rangle. \tag{C.35}$$

Privacy. To protect clients' privacy involved in their reward observations, we still consider differential privacy (DP) guarantee in the three trust models in Section 4.5 when collecting clients' observations y_t for all t.

DP algorithmic framework. To ensure DP in the standard stochastic linear bandits, we only need to address the challenges (ⓒ and ⓓ) mentioned Section 4.4.1. Following a

similar way of ensuring DP with a general **Privatizer** $\mathcal{P} = (\mathcal{R}, \mathcal{S}, \mathcal{A})$, we slightly modify our DP-DPE framework in Section 4.4 and design a new differentially private phased elimination algorithmic framework (DP-PE) for the standard linear bandits. We present the detailed pseudo-code of DP-PE in Algorithm 8.

The DP-PE algorithm runs in phases and maintains a set of active actions $\mathcal{D}_l$, which is updated at the end of each phase. At a high level, each phase consists of the following steps. First, compute a near-G-optimal design $\pi_l(\cdot)$(i.e., a distribution) over a set of possibly optimal actions $\mathcal{D}_l$. For each action x in the support set of π_l, send x to $T_l(\mathbf{x})$ clients, denoted as $U_l(\mathbf{x})$, where the action x is played and a reward $y_u(\mathbf{x})$ is observed at each client $u \in U_l(\mathbf{x})$. Before being used to estimate θ^*, the reward observations $y_u(\mathbf{x})$ at all clients $u \in U_l(\mathbf{x})$ for each chosen action x is processed by a **Privatizer** $\mathcal{P}$ to ensure differential privacy as in the DP-DPE algorithm. We still consider a **Privatizer** $\mathcal{P} = (\mathcal{R}, \mathcal{S}, \mathcal{A})$ as a process completed by the clients, the server, and/or a trusted third party. As instantiations of $\mathcal{P}$, we also consider the central, local and shuffle models and provide the detailed implementations of $\mathcal{R}, \mathcal{S}, \mathcal{A}$ in Section C.3.2. In all the DP models, the final output $\tilde{y}_l(\mathbf{x})$ of $\mathcal{P}$ for each action x is a private sum of its reward observations. With the aggregated statistics $\tilde{y}_l(\mathbf{x})$ for each action $x \in \text{supp}(\pi_l)$, the agent computes the least-square estimator $\tilde{\theta}_l$ according to Eq. (C.36). Finally, low-rewarding actions are eliminated from $\mathcal{D}_l$ (Line 16) based on the following confidence width:

$$W_l \triangleq \left(\underbrace{\sqrt{\frac{2d}{h_l}}}_{\text{action-related}} + \underbrace{\sigma_n}_{\text{privacy noise}} \right) \sqrt{2 \log \left(\frac{1}{\beta} \right)}, \tag{C.37}$$

where σ_n is determined by the privacy noise added in the DP model.

C.3.2 DP-PE Instantiations with different DP Models

We now briefly explain how to instantiate the PRIVATIZER $\mathcal{P} = (\mathcal{R}, \mathcal{S}, \mathcal{A})$ in DP-PE using the three representative DP trust models: the central, local, and shuffle models. In addition, we also present the formal definition of the privacy guarantees regarding $\mathcal{P}$ under each trust model, which further implies the respective privacy guarantee of DP-PE according to the post-processing property of DP [50, Proposition 2.1].

The Central Model

In the central model, each client trusts the server, and the outputs of the server on two neighboring datasets (differing by only one client) should be indistinguishable [51].

Consider a particular phase l and an action x in $\text{supp}(\pi_l)$. The PRIVATIZER $\mathcal{P}$ is (ε, δ)-differentially-private (or (ε, δ)-DP) if the following is satisfied for any pair of $U_l(\mathbf{x}), U_l'(\mathbf{x}) \subseteq \mathcal{U}$ that differ by at most one client and for any output $\tilde{y}$ of $\mathcal{A}$:

$$\mathbb{P}[\mathcal{P}\left(\{y_u(\mathbf{x})\}_{u \in U_l(\mathbf{x})}\right) = \tilde{y}] \leqslant e^{\varepsilon} \cdot \mathbb{P}[\mathcal{P}\left(\{y_u(\mathbf{x})\}_{u \in U_l'(\mathbf{x})}\right) = \tilde{y}] + \delta.$$

To achieve this, the PRIVATIZER functions as follows: while both $\mathcal{R}$ and $\mathcal{S}$ are simply identity mappings, $\mathcal{A}$ adds well-tuned Gaussian noise to the sum of $T_l(\mathbf{x})$ reward observations from clients $U_l(\mathbf{x})$ for each action x in $\text{supp}(\pi_l)$ for privacy. That is,

$$\tilde{y}_l(\mathbf{x}) = \mathcal{P}\left(\{y_u(\mathbf{x})\}_{u \in U_l(\mathbf{x})}\right) = \mathcal{A}\left(\{y_u(\mathbf{x})\}_{u \in U_l(\mathbf{x})}\right) = \sum_{u \in U_l(\mathbf{x})} y_u(\mathbf{x}) + \gamma_x, \quad \forall x \in \text{supp}(\pi_l), \tag{C.38}$$

where $\gamma_x \sim \mathcal{N}(0, \sigma_{nc}^2)$ is *i.i.d.* across actions, and the variance σ_{nc}^2 depends on the sensitivity of $\sum_{u \in U_l(\mathbf{x})} y_u(\mathbf{x})$, which is $2B$. Combining the Gaussian mechanism in Theorem D.13 with

the post-processing property of DP in [50], it is not difficult to obtain the following DP guarantee.

Theorem C.5. *The DP-PE instantiation using the* Privatizer *in Eq.* (C.38) *with* $\sigma_{nc} = \frac{2B\sqrt{2\ln(1.25/\delta)}}{\varepsilon}$ *guarantees* (ε, δ)*-DP.*

At the same time, with the above DP guarantee, we can show that the regret under DP-PE in the central model satisfies the following result.

Theorem C.6 (CDP-PE)**.** *With* $\sigma_n = 2d\sigma_{nc}\sqrt{s_l}/h_l$ *in each phase* l *and* $\beta = 1/(kT)$*, the DP-PE algorithm with the central model* Pivatizer *achieves expected regret*

$$\mathbb{E}[R(T)] = O(\sqrt{dT\log(kT)} + O\left(\frac{Bd^{3/2}\log(T)\sqrt{\ln(1/\delta)\log(kT)}}{\varepsilon}\right) + O\left(\frac{Bd^{3/2}\sqrt{\ln(1/\delta)\log(kT)}}{\varepsilon}\right). \tag{C.39}$$

The Local Model

In the local DP model, since clients do not trust the server, each client with a local randomizer $\mathcal{R}$ is responsible for privacy protection by injecting Gaussian noise; $\mathcal{S}$ is an identity mapping; $\mathcal{A}$ simply sums up the (private) rewards from $U_l(\mathbf{x})$ corresponding to action x. That is,

$$\tilde{y}_l(\mathbf{x}) = \sum_{u \in U_l(\mathbf{x})} \mathcal{R}(y_u(\mathbf{x})) = \sum_{u \in U_l(\mathbf{x})} \left(y_u(\mathbf{x}) + \gamma_{u,x}\right), \tag{C.40}$$

where $\gamma_{u,x} \sim \mathcal{N}(0, \sigma_{nl}^2)$ is *i.i.d.* across clients, and the variance σ_{nl}^2 is chosen according to the sensitivity of $y_u(\mathbf{x})$, which is also $2B$. Consider any phase l and any x in $\text{supp}(\pi_l)$. Let Y_u be the set of all possible values of the reward observation $y_u(\mathbf{x})$ at client u for any action x. The Privatizer $\mathcal{P}$ is (ε, δ)-local-differentially-private (or (ε, δ)-LDP) if the following is satisfied

for any client u, for any pair of $y_u(\mathbf{x}), y'_u(\mathbf{x}) \in Y_u$, and for any output $o \in \{\mathcal{R}(y)|y \in Y_u\}$:

$$\mathbb{P}[\mathcal{R}(y) = o] \leqslant e^{\varepsilon} \cdot \mathbb{P}[\mathcal{R}(y') = o] + \delta.$$

With the above definition, we present the privacy guarantee of DP-PE in the local DP model in Theorem C.7.

Theorem C.7. *The DP-PE instantiation using the* PRIVATIZER *in Eq. (C.40) with* $\sigma_{nl} = \frac{2B\sqrt{2\ln(1.25/\delta)}}{\varepsilon}$ *guarantees* (ε, δ)*-LDP.*

Theorem C.8 (LDP-PE)**.** *With* $\sigma_n = 2d\sigma_{nc}\sqrt{2s_l/h_l}$ *in each phase* l *and* $\beta = 1/(kT)$*, the DP-DPE algorithm with the local model* PIVATIZER *achieves expected regret*

$$\mathbb{E}[R(T)] = O(\sqrt{dT\log(kT)} + O\left(\frac{Bd^{3/2}\sqrt{\ln(1/\delta)T\log(kT)}}{\varepsilon}\right) + O\left(\frac{Bd^2\sqrt{\ln(1/\beta)\log(kT)}}{\varepsilon}\right). \tag{C.41}$$

The Shuffle Model

In the shuffle model, without a trusted agent, we instantiate DP-PE by building on the scalar sum protocol $\mathcal{P}_{\text{1D}}$ recently developed in [35]. Consider a particular phase l and any action x in $\text{supp}(\pi_l)$. Specifically, each local randomizer $\mathcal{R}$ encodes its inputs (the reward observation $y_u(\mathbf{x})$) by adding random bits; the analyzer $\mathcal{A}$ outputs the random number whose expectation is the sum of inputs ($\sum_{u\in U_l(\mathbf{x})} y_u(\mathbf{x})$); in addition, a third-party shuffler $\mathcal{S}$ is utilized to uniformly at random permute clients' messages (in bits) to hide their sources u. We present the concrete pseudocode of $\mathcal{R}$, $\mathcal{S}$, and $\mathcal{A}$ for the shuffle model PRIVATIZER in Algorithm 9. Finally, the private sum $\tilde{y}_l(\mathbf{x})$ is

$$\tilde{y}_l(\mathbf{x}) = \mathcal{P}\left(\{y_u(\mathbf{x})\}_{u\in U_l(\mathbf{x})}\right) = \mathcal{A}\left(\mathcal{S}\left(\{\mathcal{R}(y_u(\mathbf{x}))\}_{u\in U_l(\mathbf{x})}\right)\right). \tag{C.42}$$

Similar to the shuffle model in DP-DPE, we let $(\mathcal{S}\circ\mathcal{R})(U_l(\mathbf{x})) \triangleq \mathcal{S}(\{\mathcal{R}(y_u(\mathbf{x}))\}_{u\in U_l(\mathbf{x})})$ denote the composite mechanism. Formally, the **Privatizer** $\mathcal{P}$ is (ε,δ)-shuffle-differentially-private (or (ε,δ)-SDP) if the following is satisfied for any pair of $U_l(\mathbf{x})$, $U_l'(\mathbf{x}) \subseteq \mathcal{U}$ that differ by one client and for any possible output z of $\mathcal{S}\circ\mathcal{R}$:

$$\mathbb{P}[(\mathcal{S}\circ\mathcal{R})(U_l(\mathbf{x})) = z] \leqslant e^{\varepsilon}\cdot\mathbb{P}[(\mathcal{S}\circ\mathcal{R})(U_l'(\mathbf{x})) = z] + \delta.$$

Before showing the privacy guarantee of the shuffle model in Algorithm 9, we provide a lemma derived directly from the original results in [35].

Lemma C.9 (Lemma 3.1 in [35])**.** *Fix any number of users n, $\hat{\varepsilon} < 15$, and $0 < \delta < 1/2$. Let $g \geqslant B\sqrt{n}$, $b > \frac{180g^2\ln(1/\delta)}{\hat{\varepsilon}^2 n}$, and $p = \frac{90g^2\ln(2/\delta)}{b\hat{\varepsilon}^2 n}$. Then,*

1. *the* **Privatizer** $\mathcal{P}$ *in Algorithm 9 is* $\left(\hat{\varepsilon}\left(\frac{2}{g}+1\right),\delta\right)$*-SDP;*
2. *for any $Y \in [-B,B]^n$, $\mathcal{P}$ is an unbiased estimator of $\sum_{i=1}^n y_i$, and the error is sub-Gaussian with variance $\sigma_{ns}^2 = O\left(\frac{B^2\log(1/\delta)}{\hat{\varepsilon}^2}\right)$.*

Set the parameters p, b, g according to Eq. (C.43) with $\varepsilon = 2\hat{\varepsilon}$ in the **Privatizer** specified in Algorithm 9. Then, we derive the following privacy guarantee.

Theorem C.10. *For any $\varepsilon \in (0,30)$ and $\delta \in (1,1/2)$, the DP-PE instantiation using the* **Privatizer** *specified in Algorithm 9 guarantees (ε,δ)-SDP.*

Theorem C.11 (SDP-PE)**.** *With $\sigma_n = 2d\sigma_{ns}\sqrt{s_l}/h_l = O\left(\frac{Bd\sqrt{s_l\ln(1/\delta)}}{\varepsilon h_l}\right)$ in each phase l and $\beta = 1/(kT)$, the DP-PE algorithm with the shuffle model* **Pivatizer** *specified in Algorithm 9 achieves expected regret*

$$\mathbb{E}[R(T)] = O(\sqrt{dT\log(kT)} + O\left(\frac{Bd^{3/2}\log(T)\sqrt{\ln(1/\delta)\log(kT)}}{\varepsilon}\right) + O\left(\frac{Bd^{3/2}\sqrt{\ln(1/\delta)\log(kT)}}{\varepsilon}\right). \tag{C.44}$$

Discussions

Regarding the results derived in this subsection, we make the following remarks.

Remark C.12 (Privacy "for free")**.** In [84], we know that the non-private phased elimination algorithm achieves $O(\sqrt{dT\log(kT)})$ regret. From the above Theorems C.5, C.7 and Theorem C.7, we derive that the DP-PE algorithm enables us to achieve privacy guarantee "for free" in the central and shuffle model as in the DP-DPE algorithm.

Remark C.13 (Extended to other privacy noise)**.** To be consistent with our main content, we employ the Gaussian mechanism here to achieve the corresponding approximate DP in the central and local DP models. However, a Laplacian mechanism can also be employed in the central and the local models to achieve pure $(\varepsilon, 0)$-DP privacy guarantee.

Remark C.14 (Phase length initialization h_1)**.** In the standard linear bandits without client-related uncertainty, we initialize the phase length to be the upper bound of the support set size in every phase, i.e., $h_1 = 4d\log\log d + 16$, in order to derive a better regret cost (i.e., a lower order of d) due to privacy guarantees, especially for the local DP model.

C.3.3 Proofs for the Results in Section C.3

To prove the regret upper bound of the DP-PE algorithm under three DP models, we follow a similar line to the proof of Theorem 4.11, D.10, and D.11. First, we present the concentration result for the DP-PE algorithm under the three DP models with the setting in Table C.2.

Theorem C.15. *Set DP-PE in the central, local, and shuffle models (i.e., CDP-PE, LDP-PE, and SDP-PE respectively) based on Table C.2. In any phase l, all of CDP-PE, LDP-PE, and SDP-PE satisfies the following concentration, for any particular $x \in \mathcal{D}_l$*

$$P\left\{\langle\tilde{\theta}_l - \theta^*, \mathbf{x}\rangle \geqslant W_l\right\} \leqslant 2\beta, \quad \textit{and} \quad P\left\{\langle\theta^* - \tilde{\theta}_l, \mathbf{x}\rangle \geqslant W_l\right\} \leqslant 2\beta. \tag{C.45}$$

Table C.2: Setting

Algorithm	σ_n	Notes
CDP-PE	$\sigma_n = 2d\sigma_{nc}\sqrt{s_l}/h_l$	$\sigma_{nc} = \frac{2B\sqrt{2\ln(1.25/\delta)}}{\varepsilon}$
LDP-PE	$\sigma_n = 2d\sigma_{nl}\sqrt{2s_l/h_l}$	$\sigma_{nl} = \frac{2B\sqrt{2\ln(1.25/\delta)}}{\varepsilon}$
SDP-PE	$\sigma_n = 2d\sigma_{ns}\sqrt{s_l}/h_l$	$\sigma_{ns} = O\left(\frac{B\sqrt{\ln(1/\delta)}}{\varepsilon}\right)$

Proof. We prove the first concentration inequality in Eq. (C.45) for CDP-PE, LDP-PE, and SDP-PE in the following, and the second inequality can be proved symmetrically. According to Eq. (C.36) in Algorithm 8, we have

$$
\begin{aligned}
\tilde{\theta}_l &= V_l^{-1} G_l \\
&= V_l^{-1} \sum_{x \in \mathrm{supp}(\pi_l)} x \tilde{y}_l(\mathbf{x}) \\
&= V_l^{-1} \sum_{x \in \mathrm{supp}(\pi_l)} x \sum_{u \in U_l(\mathbf{x})} y_u(\mathbf{x}) + V_l^{-1} \sum_{x \in \mathrm{supp}(\pi_l)} x \underbrace{\left(\tilde{y}_l(\mathbf{x}) - \sum_{u \in U_l(\mathbf{x})} y_u(\mathbf{x}) \right)}_{\gamma_p(\mathbf{x})} \\
&\overset{(a)}{=} V_l^{-1} \sum_{x \in \mathrm{supp}(\pi_l)} x \sum_{u \in U_l(\mathbf{x})} y_u(\mathbf{x}) + V_l^{-1} \sum_{x \in \mathrm{supp}(\pi_l)} x \gamma_p(\mathbf{x}) \\
&= V_l^{-1} \sum_{x \in \mathrm{supp}(\pi_l)} x \sum_{t \in \mathcal{T}_l(\mathbf{x})} (x^\top \theta^* + \eta_t) + V_l^{-1} \sum_{x \in \mathrm{supp}(\pi_l)} x \gamma_p(\mathbf{x}) \\
&= V_l^{-1} \sum_{x \in \mathrm{supp}(\pi_l)} T_l(\mathbf{x}) x x^\top \theta^* + V_l^{-1} \sum_{x \in \mathrm{supp}(\pi_l)} x \sum_{t \in \mathcal{T}_l(\mathbf{x})} \eta_t + V_l^{-1} \sum_{x \in \mathrm{supp}(\pi_l)} x \gamma_p(\mathbf{x}) \\
&= \theta^* + V_l^{-1} \sum_{t \in \mathcal{T}_l} x_t \eta_t + V_l^{-1} \sum_{x \in \mathrm{supp}(\pi_l)} x \gamma_p(\mathbf{x}),
\end{aligned}
$$

where we represent the noise introduced for protecting privacy associated with action x by $\gamma_p(\mathbf{x}) \triangleq \tilde{y}_l(\mathbf{x}) - \sum_{u \in U_l(\mathbf{x})} y_u(\mathbf{x})$, which varies according to the specified DP models, and the last step is due to our decision $x_t = x$ for any x in $\mathcal{T}_l(\mathbf{x})$. Then, the difference between

estimation and the true reward of each action $x' \in \mathcal{D}_l$ is

$$\langle \tilde{\theta} - \theta^*, x' \rangle = \left\langle x', V_l^{-1} \sum_{t \in \mathcal{T}_l} x_t \eta_t \right\rangle + \left\langle x', V_l^{-1} \sum_{x \in \mathrm{supp}(\pi_l)} x \gamma_p(\mathbf{x}) \right\rangle. \tag{C.46}$$

Note that η_t is *i.i.d* 1-sub-Gaussian over t and that the chosen action x_t at t is deterministic in the l-th phase under the DP-PE algorithm. Combining the following result,

$$\sum_{t \in \mathcal{T}_l} \langle x, V_l^{-1} x_t \rangle^2 = x^\top V_l^{-1} \left(\sum_{t \in \mathcal{T}_l} x_t x_t^\top \right) V_l^{-1} x = \|x\|^2_{V_l^{-1}},$$

where the second equality is due to $V_l = \sum_{t \in \mathcal{T}_l} x_t x_t^\top$, we derive that the first term of the RHS of Eq. (C.46) is $\|x\|_{V_l^{-1}}$-sub-Gaussian. Besides, we have $\|x\|^2_{V_l^{-1}} \leqslant \frac{\|x\|^2_{V_l(\pi_l)^{-1}}}{h_l} \leqslant \frac{g(\pi_l)}{h_l} \leqslant \frac{2d}{h_l}$ by the near-G-optimal design. Based on the property of sub-Gaussian, we obtain

$$P\left\{ \left\langle x', V_l^{-1} \sum_{t \in \mathcal{T}_l} x_t \eta_t \right\rangle \geqslant \sqrt{\frac{4d}{h_l} \log\left(\frac{1}{\beta}\right)} \right\} \leqslant \exp\left\{ -\frac{\frac{4d}{h_l} \log(1/\beta)}{2\|x'\|^2_{V_l^{-1}}} \right\} = \beta. \tag{C.47}$$

Based on the union bound, we have

$$\begin{aligned}
& P\left\{ \langle \tilde{\theta}_l - \theta^*, \mathbf{x} \rangle \geqslant W_l \right\} \\
=& P\left\{ \left\langle x', V_l^{-1} \sum_{t \in \mathcal{T}_l} x_t \eta_t \right\rangle + \left\langle x', V_l^{-1} \sum_{x \in \mathrm{supp}(\pi_l)} x \gamma_p(\mathbf{x}) \right\rangle \geqslant W_l \right\} \\
\leqslant& P\left\{ \sum_{t \in \mathcal{T}_l} \left\langle x', V_l^{-1} x_t \right\rangle \eta_t \geqslant \sqrt{\frac{4d}{h_l} \log\left(\frac{1}{\beta}\right)} \right\} \\
&+ P\left\{ \left\langle x', V_l^{-1} \sum_{x \in \mathrm{supp}(\pi_l)} x \gamma_p(\mathbf{x}) \right\rangle \geqslant \sqrt{2\sigma_n^2 \log\left(\frac{1}{\beta}\right)} \right\} \\
\leqslant& \beta + P\left\{ \left\langle x', V_l^{-1} \sum_{x \in \mathrm{supp}(\pi_l)} x \gamma_p(\mathbf{x}) \right\rangle \geqslant \sqrt{2\sigma_n^2 \log\left(\frac{1}{\beta}\right)} \right\}.
\end{aligned} \tag{C.48}$$

To derive the concentration in Eq. (C.45), it remains to show that the second term is less than β under each of the three DP models. Due to different $\gamma_p(\mathbf{x})$ in different DP models, we analyze the second term respectively.

i) CDP-PE. In the central model, the private output of the Privatizer $\mathcal{P}$ is $\tilde{y}_l(\mathbf{x}) = \sum_{u\in U_l(\mathbf{x})} y_u(\mathbf{x}) + \gamma_x$, where $\gamma_x \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma_{nc}^2)$. Then, $\gamma_p(\mathbf{x}) \sim \mathcal{N}(0, \sigma_{nc}^2)$ in the central model and is *i.i.d.* across actions $x \in \text{supp}(\pi_l)$. Note that

$$\left\langle x', V_l^{-1} \sum_{x\in\text{supp}(\pi_l)} x\gamma_p(\mathbf{x}) \right\rangle = \sum_{x\in\text{supp}(\pi_l)} \left\langle x', V_l^{-1} x \right\rangle \gamma_p(\mathbf{x}),$$

and that $\gamma_p(\mathbf{x}) \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma_{nc}^2)$. The variance (denoted by σ_{sum}^2) of the above sum of *i.i.d.* Gaussian variables is

$$\sigma_{\text{sum}}^2 = \sum_{x\in\text{supp}(\pi_l)} \left\langle x', V_l^{-1} x \right\rangle^2 \sigma_{nc}^2 \overset{(a)}{\leqslant} \sum_{x\in\text{supp}(\pi_l)} \left(\max_{x\in\mathcal{D}_l} \|x\|_{V_l^{-1}}^2 \right)^2 \sigma_{nc}^2 \leqslant \frac{s_l \cdot 4d^2 \cdot \sigma_{nc}^2}{h_l^2} = \sigma_n^2,$$

where (a) is from $\langle x', V_l^{-1}\mathbf{x}\rangle \leqslant \max_{x\in\mathcal{D}_l} \|x\|_{V_l^{-1}}^2$ for the positive definite matrix V_l. Combining the tail bound for Gaussian variables, we have

$$P\left\{ \left\langle x', V_l^{-1} \sum_{x\in\text{supp}(\pi_l)} x\gamma_p(\mathbf{x}) \right\rangle \geqslant \sqrt{2\sigma_n^2 \log\left(\frac{1}{\beta}\right)} \right\} \leqslant \exp\left\{ -\frac{2\sigma_n^2 \log(1/\beta)}{2\sigma_{\text{sum}}^2} \right\} \leqslant \beta.$$

Hence, the first concentration inequality in Eq. (C.45) holds for CDP-PE algorithm.

ii) LDP-PE. In the local model, the output of the Privatizer $\mathcal{P}$ is $\tilde{y}_l(\mathbf{x}) = \sum_{u\in U_l} y_u(\mathbf{x}) + \gamma_{u,x}$, where $\gamma_{u,x} \sim \mathcal{N}(0, \sigma_{nl}^2)$ is *i.i.d.* across clients. Then $\gamma_p(\mathbf{x}) = \sum_{u\in U_l} \gamma_{u,x}$ in the local model, and

$$\left\langle x', V_l^{-1} \sum_{x\in\text{supp}(\pi_l)} x\gamma_p(\mathbf{x}) \right\rangle = \sum_{x\in\text{supp}(\pi_l)} \sum_{u\in U_l(\mathbf{x})} \left\langle x', V_l^{-1} x \right\rangle \gamma_{u,x}. \tag{C.49}$$

The variance (denoted by σ^2_{sum}) of the sum of the above $T_l(\mathbf{x}) \times |\text{supp}(\pi_l)|$ *i.i.d.* Gaussian variables satisfies

$$\sigma^2_{\text{sum}} = \sum_{u \in U_l(\mathbf{x})} \sum_{x \in \text{supp}(\pi_l)} \left\langle x', V_l^{-1} x \right\rangle^2 \sigma^2_{nl} \leqslant \frac{T_l(\mathbf{x}) s_l \cdot 4d^2 \cdot \sigma^2_{nl}}{h_l^2} \leqslant \frac{8 s_l d^2 \sigma^2_{nl}}{h_l} = \sigma^2_n,$$

where the last step is from $T_l(\mathbf{x}) \leqslant T_l = \sum_{x \in \text{supp}(\pi_l) T_l(\mathbf{x})} \leqslant h_l + s_l \leqslant h_l + h_1 \leqslant 2h_l$. Then, we have

$$P\left\{ \sum_{u \in U_l(\mathbf{x})} \sum_{x \in \text{supp}(\pi_l)} \left\langle x', V_l^{-1} x \gamma_x \right\rangle \geqslant \sqrt{2\sigma_n^2 \log\left(\frac{1}{\beta}\right)} \right\} \leqslant \exp\left\{ -\frac{2\sigma_n^2 \log\left(\frac{1}{\beta}\right)}{2\sigma^2_{\text{sum}}} \right\} \leqslant \beta.$$

Hence, the first concentration inequality in Eq. (C.45) holds for LDP-PE algorithm.

iii) SDP-PE. In the shuffle model, the output of the PRIVATIZER $\mathcal{P}$ is, $\tilde{y}_l = \mathcal{A}\left(\mathcal{S}\left(\{\mathcal{R}(y_u(\mathbf{x}))\}_{u \in U_l(\mathbf{x})}\right)\right)$, where $\mathcal{A}, \mathcal{S}$ and $\mathcal{R}$ follow Algorithm 9. From Lemma C.9, we know that the output of the $\mathcal{P}\left(\{y_u(\mathbf{x})\}_{u \in U_l(\mathbf{x})}\right)$ is an unbiased estimator of $\sum_{u \in U_l(\mathbf{x})} y_u(\mathbf{x})$ and that the error distribution is sub-Gaussian with variance $\sigma^2_{ns} = O\left(\frac{B^2 \ln(1/\delta)}{\varepsilon^2}\right)$. Then, $\gamma_p(\mathbf{x}) = \tilde{y}_l(\mathbf{x}) - \sum_{u \in U_l(\mathbf{x})} y_u(\mathbf{x})$ is σ_{ns}-sub-Gaussian with $\mathbb{E}[\gamma_p(\mathbf{x})] = 0$ in the shuffle model. Besides, $\gamma_p(\mathbf{x})$ is *i.i.d.* over each coordinate corresponding to each action x in the support set $\text{supp}(\pi_l)$. Recall that

$$\left\langle x', V_l^{-1} \sum_{x \in \text{supp}(\pi_l)} x \gamma_p(\mathbf{x}) \right\rangle = \sum_{x \in \text{supp}(\pi_l)} \left\langle x', V_l^{-1} x \right\rangle \gamma_p(\mathbf{x}).$$

The variance (denoted by σ^2_{sum}) of the above sum of *i.i.d.* sub-Gaussian variables is

$$\sigma^2_{\text{sum}} = \sum_{x \in \text{supp}(\pi_l)} \left\langle x', V_l^{-1} x \right\rangle^2 \sigma^2_{ns} \overset{(a)}{\leqslant} \sum_{x \in \text{supp}(\pi_l)} \left(\max_{x \in \mathcal{D}_l} \|x\|^2_{V_l^{-1}} \right)^2 \sigma^2_{ns} \leqslant \frac{s_l \cdot 4d^2 \cdot \sigma^2_{ns}}{h_l^2} = \sigma^2_n,$$

where (a) is from $\langle x', V_l^{-1} \mathbf{x} \rangle \leqslant \max_{x \in \mathcal{D}_l} \|x\|^2_{V_l^{-1}}$ for the positive definite matrix V_l. Combining

the tail bound for sub-Gaussian variables, we have

$$P\left\{\left\langle x', V_l^{-1} \sum_{x \in \text{supp}(\pi_l)} x\gamma_p(\mathbf{x})\right\rangle \geqslant \sqrt{2\sigma_n^2 \log\left(\frac{1}{\beta}\right)}\right\} \leqslant \exp\left\{-\frac{2\sigma_n^2 \log(1/\beta)}{2\sigma_{\text{sum}}^2}\right\} \leqslant \beta.$$

Hence, the first concentration inequality in Eq. (C.45) holds for SDP-PE algorithm.

By now, we complete the proof for Eq. (C.45) with the symmetrical argument. □

Proof of Theorem C.6

In the following, we start with analyzing regret in a specific phase under CDP-PE and then combine all phases together to get the total regret incurred by the CDP-PE algorithm.

Proof. **1) Regret in a specific phase.** Based on the concentration in Theorem C.4, we define a "good" event at l-th phase as $\mathcal{E}_l$:

$$\mathcal{E}_l \triangleq \left\{\langle \theta^* - \tilde{\theta}_l, x^* \rangle \leqslant W_l \quad \text{and} \quad \forall x \in \mathcal{D}_l \backslash \{x^*\} \quad \langle \tilde{\theta}_l - \theta^*, \mathbf{x} \rangle \leqslant W_l\right\}.$$

It is not difficult to derive $P(\mathcal{E}_l) \geqslant 1 - 2k\beta$ via union bound.

Under event $\mathcal{E}_l$, we have the following two observations:

1. If the optimal action $x^* \in \mathcal{D}_l$, then $x^* \in \mathcal{D}_{l+1}$.

2. For any $x \in \mathcal{D}_{l+1}$, we have $\langle \theta^*, x^* - \mathbf{x} \rangle \leqslant 4W_l$.

For any particular l, under event $\mathcal{E}_l$, we have the regret incurred in the $(l+1)$-th phase

satisfies (according to the second observation)

$$
\begin{aligned}
r_{l+1} &= \sum_{t\in\mathcal{T}_{l+1}} \langle \theta^*, x^* - x_t \rangle \\
&\leqslant \sum_{t\in\mathcal{T}_{l+1}} 4W_l \\
&= 4T_{l+1}W_l \\
&\leqslant \underbrace{4T_{l+1}\sqrt{\frac{4d}{h_l}\log\left(\frac{1}{\beta}\right)}}_{①} + \underbrace{4T_{l+1}\sqrt{2\sigma_n^2\log\left(\frac{1}{\beta}\right)}}_{②}.
\end{aligned}
\tag{C.50}
$$

Note that $T_l = \sum_{x\in\mathrm{supp}(\pi_l)} T_l(\mathbf{x}) \leqslant h_l + s_l$. For ①, we have

$$
\begin{aligned}
① &\leqslant\ 4(h_1\cdot 2^l + s_{l+1})\sqrt{\frac{4d}{h_1\cdot 2^{l-1}}\log\left(\frac{1}{\beta}\right)} \\
&= 8\sqrt{2d\log\left(\frac{1}{\beta}\right)}\left(\sqrt{h_1\cdot 2^l} + \frac{s_{l+1}}{\sqrt{h_1\cdot 2^l}}\right) \\
&\leqslant 8\sqrt{2d\log\left(\frac{1}{\beta}\right)}\left(\sqrt{h_1\cdot 2^l} + \frac{S}{\sqrt{h_1\cdot 2^l}}\right),
\end{aligned}
\tag{C.51}
$$

where $S \triangleq 4d\log\log d + 16 \geqslant s_l$. As to the second term ②, we have $\sigma_n^2 = \frac{4s_l d^2\sigma_{nc}^2}{h_l^2}$, and then

$$
\begin{aligned}
② &\leqslant 4(h_1\cdot 2^l + s_{l+1})\sqrt{\frac{8s_l d^2\sigma_{nc}^2\log(1/\beta)}{h_1^2\cdot 2^{2(l-1)}}} \\
&= 16d\sigma_{nc}\sqrt{2s_l\log\left(\frac{1}{\beta}\right)}\left(1 + \frac{s_{l+1}}{h_1\cdot 2^l}\right) \\
&= 16d\sigma_{nc}\sqrt{2S\log\left(\frac{1}{\beta}\right)}\left(1 + \frac{S}{h_1\cdot 2^l}\right).
\end{aligned}
\tag{C.52}
$$

Then, for any $l \geqslant 2$, the regret in the l-th phase r_l is upper bounded by

$$r_l \leqslant 8\sqrt{2d\log\left(\frac{1}{\beta}\right)}\left(\sqrt{h_1 \cdot 2^{l-1}} + \frac{S}{\sqrt{h_1 \cdot 2^{l-1}}}\right) + 16d\sigma_{nc}\sqrt{2S\log\left(\frac{1}{\beta}\right)}\left(1 + \frac{S}{h_1 \cdot 2^{l-1}}\right). \tag{C.53}$$

2) Total regret. Define $\mathcal{E}_g$ as the event where the "good" event occurs in every phase, i.e., $\mathcal{E}_g \triangleq \bigcap_{l=1}^{L} \mathcal{E}_l$. It is not difficult to obtain $P\{\mathcal{E}_g\} \geqslant 1 - 2k\beta L$ by applying union bound. At the same time, let R_g be the regret under event $\mathcal{E}_g$, and R_b be the regret if event $\mathcal{E}_g$ does not hold. Then, the expected total regret in T is $\mathbb{E}[R(T)] = P(\mathcal{E}_g)R_g + (1 - P(\mathcal{E}_g))R_b$.

Under event $\mathcal{E}_g$, the regret in the l-th phase r_l satisfies Eq. (C.53) for any $l \geqslant 2$. Combining $r_1 \leqslant 2T_1 \leqslant 4h_1$ (since $\langle \theta^*, x^* - x\rangle \leqslant 2$ for all $\mathbf{x} \in \mathcal{D}$), we have

$$\begin{aligned}
R_g &= \sum_{l=1}^{L} r_l \\
&\leqslant 2(h_1 + S) + \sum_{l=2}^{L} 8\sqrt{2d\log\left(\frac{1}{\beta}\right)}\left(\sqrt{h_1 \cdot 2^{l-1}} + \frac{S}{\sqrt{h_1 \cdot 2^{l-1}}}\right) \\
&+ \sum_{l=2}^{L} 16d\sigma_{nc}\sqrt{2S\log\left(\frac{1}{\beta}\right)}\left(1 + \frac{S}{h_1 \cdot 2^{l-1}}\right) \\
&\leqslant 2(h_1 + S) + 8\sqrt{2d\log(1/\beta)}\left(4\sqrt{h_1 2^{L-1}} + \frac{3S}{\sqrt{h_1}}\right) \\
&+ 16d\sigma_{nc}\sqrt{2S\log(1/\beta)}\left(L - 1 + S/h_1\right).
\end{aligned}$$

Note that $h_L \leqslant T_L \leqslant T$, which indicates $2^{L-1} \leqslant T/h_1$, and $L \leqslant \log(2T/h_1)$. Then, the above inequality becomes

$$\begin{aligned}
R_g &= \sum_{l=1}^{L} r_l \\
&\leqslant 2(h_1 + S) + 8\sqrt{2d\log(1/\beta)} \cdot 4\sqrt{T} + 24S\sqrt{2d\log(1/\beta)}/\sqrt{h_1} \\
&+ 16d\sigma_{nc}\sqrt{2S\log(1/\beta)} \cdot \log(T/h_1) + 16dS/h_1\sigma_{nc}\sqrt{2S\log(1/\beta)}.
\end{aligned}$$

On the other hand, $R_b \leqslant 2T$ since $\langle \theta^*, x^* - x \rangle \leqslant 2$ for all $\mathbf{x} \in \mathcal{D}$. Choose $\beta = \frac{1}{kT}$ in Algorithm 8. Finally, we have the following results:

$$\begin{aligned}
\mathbb{E}[R(T)] &= P(\mathcal{E}_g)R_g + (1 - P(\mathcal{E}_g))R_b \\
&\leqslant R_g + 2k\beta L \cdot 2T \\
&\leqslant 2(h_1 + S) + 32\sqrt{2dT\log(kT)} + 24S\sqrt{2d/h_1\log(kT)} \\
&+ 16d\sigma_{nc}\sqrt{2S\log(kT)}\log(T) + 16dS/h_1\sigma_{nc}\sqrt{2S\log(kT)} + 4\log(2T/h_1) \\
&= O(\sqrt{dT\log(kT)} + O(\sigma_{nc}d^{3/2}\log(T)\sqrt{\log(kT)} + O(\sigma_{nc}d^{3/2}\sqrt{\log(kT)}).
\end{aligned}$$

Finally, substituting $\sigma_{nc} = \frac{2B\sqrt{2\ln(1.25/\delta)}}{\varepsilon}$, we have the total expected regret under DP-PE with the central model PRIVATIZER is

$$\mathbb{E}[R(T)] = O(\sqrt{dT\log(kT)} + O\left(\frac{Bd^{3/2}\log(T)\sqrt{\ln(1/\delta)\log(kT)}}{\varepsilon}\right) + O\left(\frac{Bd^{3/2}\sqrt{\ln(1/\delta)\log(kT)}}{\varepsilon}\right). \tag{C.54}$$

□

Proof of Theorem C.7

Proof. **1) Regret in a specific phase.** Recall the "good" event at l-th phase $\mathcal{E}_l = \{\langle \theta^* - \tilde{\theta}_l, x^* \rangle \leqslant W_l\} \bigcap \{\langle \tilde{\theta}_l - \theta^*, \mathbf{x} \rangle \leqslant W_l, \forall x \in \mathcal{D}_l \backslash \{x^*\}\}$. Under event $\mathcal{E}_l$, we have the following two observations:

1. If the optimal action $x^* \in \mathcal{D}_l$, then $x^* \in \mathcal{D}_{l+1}$.

2. For any $x \in \mathcal{D}_{l+1}$, we have $\langle \theta^*, x^* - \mathbf{x} \rangle \leqslant 4W_l$.

For any particular l, under event $\mathcal{E}_l$, we have the regret incurred in the $(l+1)$-th phase satisfies (according to the second observation)

$$\begin{aligned} r_{l+1} &= \sum_{t\in\mathcal{T}_{l+1}} \langle \theta^*, x^* - x_t \rangle \\ &\leqslant \sum_{t\in\mathcal{T}_{l+1}} 4W_l \\ &= 4T_{l+1}W_l \\ &\leqslant \underbrace{4T_{l+1}\sqrt{\frac{4d}{h_l}\log\left(\frac{1}{\beta}\right)}}_{①} + \underbrace{4T_{l+1}\sqrt{2\sigma_n^2\log\left(\frac{1}{\beta}\right)}}_{②}. \end{aligned} \tag{C.55}$$

For ①, we already have ① $= 8\sqrt{2d\log\left(\frac{1}{\beta}\right)}\left(\sqrt{h_1\cdot 2^l} + \frac{S}{\sqrt{h_1\cdot 2^l}}\right)$. As to the second term ②, we have $\sigma_n^2 = \frac{8d^2 s_l \sigma_{nl}^2}{h_l}$ in the local model and then

$$\begin{aligned} ② &\leqslant 4(h_1\cdot 2^l + s_{l+1})\sqrt{\frac{16 s_l d^2 \sigma_{nl}^2 \log(1/\beta)}{h_1\cdot 2^{(l-1)}}} \\ &= 16d\sigma_{nl}\sqrt{2s_l\log\left(\frac{1}{\beta}\right)}\left(\sqrt{h_1\cdot 2^l} + \frac{s_{l+1}}{\sqrt{h_1\cdot 2^l}}\right) \\ &\leqslant 16d\sigma_{nl}\sqrt{2S\log\left(\frac{1}{\beta}\right)}\left(\sqrt{h_1\cdot 2^l} + \frac{S}{\sqrt{h_1\cdot 2^l}}\right). \end{aligned} \tag{C.56}$$

Then, the regret in the l-th phase is upper bounded by

$$\begin{aligned} r_l &\leqslant 8\sqrt{2d\log\left(\frac{1}{\beta}\right)}\left(\sqrt{h_1\cdot 2^{l-1}} + \frac{1}{\sqrt{h_1\cdot 2^{l-1}}}\right) + 16d\sigma_{nl}\sqrt{2S\log\left(\frac{1}{\beta}\right)}\left(\sqrt{h_1\cdot 2^{l-1}} + \frac{S}{\sqrt{h_1\cdot 2^{l-1}}}\right) \\ &= (8\sqrt{d} + 16d\sigma_{nl}\sqrt{S})\sqrt{2\log\left(\frac{1}{\beta}\right)}\left(\sqrt{h_1\cdot 2^{l-1}} + \frac{S}{\sqrt{h_1\cdot 2^{l-1}}}\right). \end{aligned}$$

3) Total regret.

Under the "good" event, the total regret is

$$
\begin{aligned}
R_g &= \sum_{l=1}^{L} r_l \\
&\leqslant 2(h_1+S) + \sum_{l=2}^{L}(8\sqrt{d}+16d\sigma_{nl}\sqrt{S})\sqrt{2\log\left(\frac{1}{\beta}\right)}\left(\sqrt{h_1\cdot 2^{l-1}}+\frac{S}{\sqrt{h_1\cdot 2^{l-1}}}\right) \\
&\leqslant 2(h_1+S) + (8\sqrt{d}+16d\sigma_{nl}\sqrt{S})\sqrt{2\log\left(\frac{1}{\beta}\right)}\left(4\sqrt{h_1\cdot 2^{L-1}}+\frac{3S}{\sqrt{h_1}}\right) \\
&\leqslant 2(h_1+S) + (8\sqrt{d}+16d\sigma_{nl}\sqrt{S})\sqrt{2\log\left(\frac{1}{\beta}\right)}\left(4\sqrt{T}+\frac{S}{\sqrt{h_1}}\right) && (h_L = 2^{L-1}h_1 \leqslant T) \\
&\leqslant 2(h_1+S) + 32\sqrt{2dT\log(kT)} + 64d\sigma_{nl}\sqrt{2ST\log(kT)} \\
&+ (8S\sqrt{d/h_1}+16d\sigma_{nl}S^{3/2}/\sqrt{h_1})\sqrt{2\log(kT)} && (\beta = \frac{1}{kT}) \\
&= O(\sqrt{dT\log(kT)}) + O(d^{3/2}\sigma_{nl}\sqrt{T\log(kT)}) + O(\sigma_{nl}d^2\sqrt{\log(kT)}).
\end{aligned}
$$

Finally, substituting $\sigma_{nl} = \frac{2B\sqrt{2\ln(1.25/\delta)}}{\varepsilon}$ and combining $R_b \leqslant 2k\beta L\cdot 2T = O(\log(T))$, we have the total expected regret under DP-PE with the local model **PRIVATIZER** is

$$
\mathbb{E}[R(T)] = O(\sqrt{dT\log(kT)} + O\left(\frac{Bd^{3/2}\sqrt{\ln(1/\delta)T\log(kT)}}{\varepsilon}\right) + O\left(\frac{Bd^2\sqrt{\ln(1/\beta)\log(kT)}}{\varepsilon}\right).
$$

□

Proof of Theorem C.10

Proof. **1) Regret in a specific phase.** In the shuffle model, $\sigma_n^2 = \frac{4s_l d^2\sigma_{ns}^2}{h_l^2}$, and then the regret in the l-th phase has the same form as in the central model, i.e.,

$$
r_l \leqslant 8\sqrt{2d\log\left(\frac{1}{\beta}\right)}\left(\sqrt{h_1\cdot 2^{l-1}}+\frac{S}{\sqrt{h_1\cdot 2^{l-1}}}\right) + 16d\sigma_{ns}\sqrt{2S\log\left(\frac{1}{\beta}\right)}\left(1+\frac{S}{h_1\cdot 2^{l-1}}\right).
$$

2) Total regret. Under the "good" event, the total regret is

$$
\begin{aligned}
R_g &= \sum_{l=1}^{L} r_l \\
&\leqslant 2(h_1+S) + \sum_{l=2}^{L} 8\sqrt{2d\log\left(\frac{1}{\beta}\right)}\left(\sqrt{h_1\cdot 2^{l-1}} + \frac{S}{\sqrt{h_1\cdot 2^{l-1}}}\right) \\
&+ \sum_{l=2}^{L} 16d\sigma_{ns}\sqrt{2S\log\left(\frac{1}{\beta}\right)}\left(1+\frac{S}{h_1\cdot 2^{l-1}}\right) \\
&\leqslant 2(h_1+S) + 8\sqrt{2d\log(1/\beta)}\left(4\sqrt{h_1 2^{L-1}} + \frac{3S}{\sqrt{h_1}}\right) \\
&+ 16d\sigma_{ns}\sqrt{2S\log(1/\beta)}\,(L-1+S/h_1) \\
&\leqslant 2(h_1+S) + 8\sqrt{2d\log(1/\beta)}\cdot 4\sqrt{T} + 24S\sqrt{2d\log(1/\beta)}/\sqrt{h_1} \\
&+ 16d\sigma_{ns}\sqrt{2S\log(1/\beta)}\cdot\log(T/h_1) + 16dS/h_1\sigma_{ns}\sqrt{2S\log(1/\beta)} && (h_L = h_1\cdot 2^{L-1} \leqslant T) \\
&\leqslant 2(h_1+S) + 32\sqrt{2dT\log(kT)} + 24S\sqrt{2d/h_1\log(kT)} \\
&+ 16d\sigma_{ns}\sqrt{2S\log(kT)}\cdot\log(T/h_1) + 16dS/h_1\sigma_{ns}\sqrt{2S\log(kT)} && (\beta = \frac{1}{kT}) \\
&= O(\sqrt{dT\log(kT)} + O(\sigma_{ns}d^{3/2}\log(T)\sqrt{\log(kT)} + O(\sigma_{ns}d^{3/2}\sqrt{\log(kT)}).
\end{aligned}
$$

Finally, substituting $\sigma_{ns} = O\left(\frac{B\sqrt{d}\ln(d/\delta)}{\varepsilon}\right)$ and combining $R_b \leqslant 2k\beta L\cdot 2T = O(\log(T))$, we have the total expected regret under DP-PE with the shuffle model **Privatizer** is

$$
\mathbb{E}[R(T)] = O(\sqrt{dT\log(kT)} + O\left(\frac{Bd^{3/2}\log(T)\sqrt{\ln(1/\delta)\log(kT)}}{\varepsilon}\right) + O\left(\frac{Bd^{3/2}\sqrt{\ln(1/\delta)\log(kT)}}{\varepsilon}\right).
$$

□

Algorithm 7 $\mathcal{M}$: (ε, δ)-SDP vector average mechanism for a set U of clients

Input: $\{y_u\}_{u \in U}$, where each $y_u \in \mathbb{R}^s$, $\|y_u\|_2 \leqslant \Delta_2$

Let

$$\begin{cases} \hat{\varepsilon} = \frac{\varepsilon}{18\sqrt{\log(2/\delta)}} \\ g = \max\{\hat{\varepsilon}\sqrt{|U|}/(6\sqrt{5\ln((4s)/\delta)}), \sqrt{s}, 10\} \\ b = \lceil \frac{180 g^2 \ln(4s/\delta)}{\hat{\varepsilon}^2 |U|} \rceil \\ p = \frac{90 g^2 \ln(4s/\delta)}{b \hat{\varepsilon}^2 |U|} \end{cases} \tag{C.5}$$

// Local Randomizer

function $\mathcal{R}(y_u)$

  for coordinate $j \in [s]$ **do**

    Shift data to enforce non-negativity: $w_{u,j} = (y_u)_j + \Delta_2, \forall u \in U$

    //randomizer for each entry

    Set $\bar{w}_{u,j} \leftarrow \lfloor w_{u,j} g/(2\Delta_2) \rfloor$ $\quad$ //$\max|(y_u)_j + \Delta_2| \leqslant 2\Delta_2$

    Sample rounding value $\gamma_1 \sim \mathbf{Ber}(w_{u,j} g/(2\Delta_2) - \bar{w}_{u,j})$

    Sample privacy noise value $\gamma_2 \sim \mathbf{Bin}(b, p)$

    Let ϕ_j^u be a multi-set of $(g+b)$ bits associated with the j-th coordinate of client u, where ϕ_j^u consists of $\bar{w}_{u,j} + \gamma_1 + \gamma_2$ copies of 1 and $g + b - (\bar{w}_{i,j} + \gamma_1 + \gamma_2)$ copies of 0

  end for

  Report $\{(j, \phi_j^u)\}_{j \in [s]}$ to the shuffler

end function

// Shuffler

function $\mathcal{S}(\{(j, \vec{\phi}_j)\}_{j \in [s]})$ $\quad$ //$\vec{\phi}_j = (\phi_j^u)_{u \in U}$

  for each coordinate $j \in [s]$ **do**

    Shuffle and output all $(g+b)|U|$ bits in $\vec{\phi}_j$

  end for

end function

// Analyzer

function $\mathcal{A}(\mathcal{S}(\{(j, \vec{\phi}_j)\}_{j \in [s]})$

  for coordinate $j \in [s]$ **do**

    Compute $z_j \leftarrow \frac{2\Delta_2}{g|U|}((\sum_{i=1}^{(g+b)|U|} (\vec{\phi}_j)_i) - b|U|p)$ $\quad$ // $(\vec{\phi})_i$ denotes the i-th bit in $\vec{\phi}_j$

    Re-center: $o_j \leftarrow z_j - \Delta_2$

  end for

  Output the estimator of vector average $o = (o_j)_{j \in [s]}$

end function

Algorithm 8 Differentially Private Phased Elimination (DP-PE)

Input: $\mathcal{D} \subseteq \mathbb{R}^d$, $\phi \in (0,1)$, $\alpha \in (0,1)$, $\beta \in (0,1)$, and σ_n
Initialization: $l = 1$, $t_1 = 1$, $\mathcal{D}_1 = \mathcal{D}$, and $h_1 = 4d \log\log d + 16$
while $t_l \leqslant T$ **do**
  Find a distribution $\pi_l(\cdot)$ over $\mathcal{D}_l$ such that $g(\pi_l) \triangleq \max_{x \in \mathcal{D}_l} \|x\|^2_{V(\pi_l)^{-1}} \leqslant 2d$ and $|\text{supp}(\pi_l)| \leqslant 4d \log\log d + 16$, where $V(\pi_l) \triangleq \sum_{x \in \mathcal{D}_l} \pi_l(\mathbf{x}) x x^\top$
  for each action $x \in \text{supp}(\pi_l)$ **do**
    Select $T_l(\mathbf{x}) = \lceil h_l \pi_l(\mathbf{x}) \rceil$ clients, denoted as $U_l(\mathbf{x})$
    for each client u in $U_l(\mathbf{x})$ **do**
      Play the action x and observes the reward $y_u(\mathbf{x})$
      # The local randomizer $\mathcal{R}$ at each client:
      Run the local randomizer $\mathcal{R}$ and send the output $\mathcal{R}(y_u(\mathbf{x}))$ to $\mathcal{S}$
      If the total number of action pulls reaches T, exit
    end for
    # Computation $\mathcal{S}$ at a trusted third party:
    Run the computation function $\mathcal{S}$ and send the output $\mathcal{S}(\{\mathcal{R}(y_u(\mathbf{x}))\}_{u \in U_l(\mathbf{x})})$ to the analyzer $\mathcal{A}$
    # The analyzer $\mathcal{A}$ at the server:
    Generate the privately aggregated statistics: $\tilde{y}_l(\mathbf{x}) = \mathcal{A}\left(\mathcal{S}\left(\{\mathcal{R}(y_u)\}_{u \in U_l(\mathbf{x})}\right)\right)$
  end for
  Compute the following quantities:

$$\begin{cases} V_l = \sum_{x \in \text{supp}(\pi_l)} T_l(\mathbf{x}) x x^\top \\ G_l = \sum_{x \in \text{supp}(\pi_l)} x \tilde{y}_l(\mathbf{x}) \\ \tilde{\theta}_l = V_l^{-1} G_l \end{cases} \tag{C.36}$$

  Find low-rewarding actions with confidence width W_l:

$$E_l = \left\{ x \in \mathcal{D}_l : \max_{b \in \mathcal{D}_l} \langle \tilde{\theta}_l, b - \mathbf{x} \rangle > 2W_l \right\}$$

  Update: $\mathcal{D}_{l+1} = \mathcal{D}_l \backslash E_l$, $h_{l+1} = 2h_l$, $t_{l+1} = t_l + T_l$, and $l = l + 1$
end while

Algorithm 9 $\mathcal{P}$: A shuffle protocol for summing n scalars [35]

Input: Scalar database $Y = (y_1, \ldots, y_n) \in [-B, B]^n$; $g, b \in \mathbb{N}; p \in (0, 1/2)$; ε, δ

Let

$$\begin{cases} g \geqslant 2B\sqrt{n} \\ b = \lceil \frac{180g^2 \ln(2/\delta)}{(\varepsilon/2)^2 n} \rceil \\ p = \frac{90g^2 \ln(2/\delta)}{b(\varepsilon/2)^2 n} \end{cases} \tag{C.43}$$

// Local Randomizer

function $\mathcal{R}(y)$

  Shift data to enforce non-negativity: $y \leftarrow y + B$

  Set $\bar{y} \leftarrow \lfloor yg/(2B) \rfloor$

  Sample rounding value $\gamma_1 \sim \mathbf{Ber}(yg/(2B) - \bar{y})$

  Set $\hat{y} \leftarrow \bar{y} + \gamma_1$

  Sample privacy noise value $\gamma_2 \sim \mathbf{Bin}(b, p)$

  Let ϕ be a multi-set of $(g + b)$ bits, containing $\hat{y} + \gamma_2$ copies of 1 and $g + b - (\hat{y} + \gamma_2)$ copies of 0

  Report ϕ to the shuffler

end function

// Shuffler

function $\mathcal{S}(\phi_1, \ldots, \phi_n)$ // each ϕ_i consists of $(g + b)$ bits

  Shuffle and output all $(g + b)n$ bits

end function

// Analyzer

function $\mathcal{A}(\mathcal{S}(\phi_1, \ldots, \phi_n))$

  Compute $z \leftarrow \frac{2B}{g}(\mathrm{sum}(\mathcal{S}(\phi_1, \ldots, \phi_n)) - nbp)$

  Re-center: $o \leftarrow z - nB$

  Output the estimator o

end function

Appendix D

Appendix for Chapter 5

D.1 Kernelized Bandits: Useful Definitions and Useful Results

D.1.1 Example Kernel Functions

In the following, we list some commonly used kernel functions $k : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$:

- Linear kernel: $k_{\text{lin}}(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}'$
- Squared exponential kernel: $k_{\text{SE}}(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'}{2l^2}\right)$
- Matern kernel: $k_{\text{Mat}}(\mathbf{x}, \mathbf{x}') = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}\|\mathbf{x}-\mathbf{x}'\|}{l}\right) J_\nu \left(\frac{\sqrt{2\nu}\|\mathbf{x}-\mathbf{x}'\|}{l}\right)$

where l denote the length-scale hyperparameter, $\nu > 0$ is an additional hyperparameter that dictates the smoothness, and J_ν and Γ_ν denote the modified Bessel function and the Gamma function, respectively [113].

D.1.2 Maximum Information Gain for Different Kernels

We present the bounds on γ_T and regret under two common kernels below in Table D.1.

Table D.1: Bounds on γ_T and Regret under Two Common Kernels [134]

Kernel	`Upper Bound on` γ_T	`Regret Lower Bound`	`Regret Upper Bound` $O(\sqrt{\gamma_T T})$
SE	$O\left(\log^{d+1}(T)\right)$	$\Omega\left(\sqrt{T\log^{\frac{d}{2}}(T)}\right)$	$O\left(\sqrt{T\log^{d+1}(T)}\right)$
Matérn-ν	$O\left(T^{\frac{d}{2\nu+d}}\log^{\frac{2\nu}{2\nu+d}}(T)\right)$	$\Omega\left(T^{\frac{\nu+d}{2\nu+d}}\right)$	$O\left(T^{\frac{\nu+d}{2\nu+d}}\log^{\frac{\nu}{2\nu+d}}(T)\right)$

D.1.3 Useful Results

Lemma D.1 (Sum of variance. Lemma 6 in [114])**.** *Let* $\mathbf{X}_t = [\mathbf{x}_1^\top, \dots, \mathbf{x}_t^\top]^\top$*, and* $\sigma_t^2(\mathbf{x}) \triangleq k(\mathbf{x},\mathbf{x}) - \mathbf{k}(\mathbf{x},\mathbf{X}_t)^\top(\mathbf{K}_{\mathbf{X}_t\mathbf{X}_t} + \lambda\mathbf{I})^{-1}\mathbf{k}(\mathbf{x},\mathbf{X}_t)$ *for any* $\mathbf{x} \in \mathcal{D}$*. Then, we have*

$$\sum_{s=1}^{t} \sigma_s^2(x_s) \leqslant \lambda \ln|\lambda^{-1}\mathbf{K}_{\mathbf{X}_t\mathbf{X}_t} + \mathbf{I}| \leqslant 2\lambda\gamma_t. \tag{D.1}$$

Lemma D.2 (Proposition A.1 in [23]/Lemma 4 in [22])**.** *For any kernel* k*, set of points* $\mathbf{X}_\tau$*,* $\mathbf{x} \in \mathcal{D}$*, and* $\tau' < \tau$

$$1 \leqslant \frac{\sigma_{\tau'}^2(\mathbf{x})}{\sigma_\tau^2(\mathbf{x})} \leqslant 1 + \sum_{s=\tau'+1}^{\tau} \sigma_{\tau'}^2(\mathbf{x}_s). \tag{D.2}$$

D.1.4 Formulation in Feature Space

For several of the proofs, it will be useful to introduce the so-called feature space (RKHS) formulation of any point in the primal space $\mathbb{R}^d$. In particular, we define a feature map $\varphi(\mathbf{x}) = k(\mathbf{x},\cdot)$ where $\varphi : \mathcal{D} \to \mathcal{H}_k$ with $\mathcal{H}_k$ being the reproducing kernel Hilbert space (RKHS) associated with kernel function k. According to the properties of RKHS, we have the following observations:

- For any $\mathbf{x}, \mathbf{x}'$, $k(\mathbf{x},\mathbf{x}') = \varphi(\mathbf{x})^\top\varphi(\mathbf{x}')$
- For any function $f \in \mathcal{H}$, $f(\mathbf{x}) = \langle f, \varphi(\mathbf{x})\rangle = \varphi(\mathbf{x})^\top f$

- Fundamental linear algebra equality

$$(BB^\top + \lambda \mathbf{I})^{-1}B = B(B^\top B + \lambda \mathbf{I})^{-1} \tag{D.3}$$

- Define $\Phi_h \triangleq [\varphi(\mathbf{a}_1)^\top, \ldots, \varphi(\mathbf{a}_h)^\top]^\top$. Then, the kernel matrix $\mathbf{K}_{\mathbf{A}_h\mathbf{A}_h} = \Phi_h\Phi_h^\top$ and $k(\mathbf{x}, \mathbf{A}_h) = \Phi_h\varphi(\mathbf{x})$, and the variance function $\Sigma_h^2(\cdot)$ represented in the feature space will be

$$\begin{aligned}\Sigma_h^2(\mathbf{x}) &= k(\mathbf{x},\mathbf{x}) - \mathbf{k}(\mathbf{x},\mathbf{A}_h)^\top(\mathbf{K}_{\mathbf{A}_h\mathbf{A}_h} + \lambda\mathbf{W}_h^{-1})^{-1}\mathbf{k}(\mathbf{x},\mathbf{A}_h)\\ &= \varphi(\mathbf{x})^\top\varphi(\mathbf{x}) - \varphi(\mathbf{x})^\top\Phi_h^\top(\Phi_h\Phi_h^\top + \lambda\mathbf{W}_h^{-1})^{-1}\Phi_h\varphi(\mathbf{x})\end{aligned} \tag{D.4}$$

- Consider any phase l. Recall that H_l is the number of batches in the l-th phase. Define $\Phi_{H_l} \triangleq [\varphi(\mathbf{a}_1)^\top, \ldots, \varphi(\mathbf{a}_{H_l})^\top]^\top$. Then, the kernel matrix $\mathbf{K}_{\mathbf{A}_{H_l}\mathbf{A}_{H_l}} = \Phi_{H_l}\Phi_{H_l}^\top$ and $k(\mathbf{x}, \mathbf{A}_{H_l}) = \Phi_{H_l}\varphi(\mathbf{x})$

- Define $\Phi_\tau \triangleq [\varphi(\mathbf{x}_{t_l+1})^\top, \ldots, \varphi(\mathbf{x}_{t_l+\tau})^\top]^\top$. Then, the kernel matrix $\mathbf{K}_{\mathbf{X}_\tau\mathbf{X}_\tau} = \Phi_\tau\Phi_\tau^\top$, $k(\mathbf{x}, \mathbf{X}_\tau) = \Phi_\tau\varphi(\mathbf{x})$, and the variance function $\sigma_\tau^2(\cdot)$ represented in the feature space will be

$$\begin{aligned}\sigma_\tau^2(\mathbf{x}) &= k(\mathbf{x},\mathbf{x}) - \mathbf{k}(\mathbf{x},\mathbf{X}_\tau)^\top(\mathbf{K}_{\mathbf{X}_\tau\mathbf{X}_\tau} + \lambda\mathbf{I})^{-1}\mathbf{k}(\mathbf{x},\mathbf{X}_\tau)\\ &= \varphi(\mathbf{x})^\top\varphi(\mathbf{x}) - \varphi(\mathbf{x})^\top\Phi_\tau^\top(\Phi_\tau\Phi_\tau^\top + \lambda\mathbf{I})^{-1}\Phi_\tau\varphi(\mathbf{x})\\ &= \varphi(\mathbf{x})^\top\varphi(\mathbf{x}) - \varphi(\mathbf{x})^\top(\Phi_\tau^\top\Phi_\tau + \lambda\mathbf{I})^{-1}\Phi_\tau^\top\Phi_\tau\varphi(\mathbf{x})\\ &= \varphi(\mathbf{x})^\top(\Phi_\tau^\top\Phi_\tau + \lambda\mathbf{I})^{-1}(\Phi_\tau^\top\Phi_\tau + \lambda\mathbf{I})\varphi(\mathbf{x}) - \varphi(\mathbf{x})^\top(\Phi_\tau^\top\Phi_\tau + \lambda\mathbf{I})^{-1}\Phi_\tau^\top\Phi_\tau\varphi(\mathbf{x})\\ &= \lambda\varphi(\mathbf{x})^\top(\Phi_\tau^\top\Phi_\tau + \lambda\mathbf{I})^{-1}\varphi(\mathbf{x})\end{aligned} \tag{D.5}$$

- Define $\Phi_{T_l} \triangleq [\varphi(\mathbf{x}_{t_l+1})^\top, \ldots, \varphi(\mathbf{x}_{t_l+T_l})^\top]^\top$. Then, the kernel matrix $\mathbf{K}_{\mathbf{X}_{T_l}\mathbf{X}_{T_l}} = \Phi_{T_l}\Phi_{T_l}^\top$, $k(\mathbf{x}, \mathbf{X}_{T_l}) = \Phi_{T_l}\varphi(\mathbf{x})$, and $f(\mathbf{X}_{T_l}) = \Phi_{T_l}f$.

D.2 Auxiliary Results and Proofs for Regret Analysis

D.2.1 Equivalent Representations

Consider any phase l. We use τ to denote the within-phase time index, i.e., $\tau \in \{1, \cdots, T_l\}$. Define τ_h as the last within-phase time index of the h-th batch, i.e., $\tau_h \triangleq \max\{\tau : t_l + \tau \in \mathcal{T}_l(\mathbf{a}_h)\}$. Then, after playing τ_h actions, the posterior variance in the traditional GP model is as follows,

$$\sigma^2_{\tau_h}(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}) - \mathbf{k}(\mathbf{x}, \mathbf{X}_{\tau_h})^\top (\mathbf{K}_{\mathbf{X}_{\tau_h}\mathbf{X}_{\tau_h}} + \lambda \mathbf{I})^{-1} \mathbf{k}(\mathbf{x}, \mathbf{X}_{\tau_h}). \tag{D.6}$$

For the posterior mean, without the observations $\mathbf{y}_{T_l} = [y_{t_l+1}, \ldots, y_{t_l+T_l}]^\top$ corresponding to the actions $\mathbf{X}_{T_l} = [\mathbf{x}^\top_{t_l+1} \ldots, \mathbf{x}^\top_{t_l+T_l}]^\top$, we replace $\mathbf{y}_{T_l}$ with $\frac{1}{|U_l|}\sum_{u \in U_l} \mathbf{y}_{l,u}$ where $\mathbf{y}_{l,u} = [y_{u,t_l+1}, \ldots, y_{u,t_l+T_l}]^\top$ in the traditional GP model. Then, the posterior mean becomes the following:

$$\mu_{T_l}(\mathbf{x}) = \frac{1}{|U_l|} \sum_{u \in U_l} \mathbf{k}(\mathbf{x}, \mathbf{X}_{T_l})^\top (\mathbf{K}_{\mathbf{X}_{T_l}\mathbf{X}_{T_l}} + \lambda \mathbf{I})^{-1} \mathbf{y}_{l,u}. \tag{D.7}$$

In our algorithm, in order to save computation complexity and communication cost, we use Eq. (5.7) and Eq. (5.11) instead of the above formula. In the following lemma, we show that they are equivalent.

Lemma D.3 (Equivalent representations)**.** *Consider any phase l. By the end of the h-th phase, the posterior variance Eq. (D.6) in the traditional GP model is equivalent to Eq. (5.7) used in our* `DPBE` *algorithm. That is, for any $\mathbf{x} \in \mathcal{D}$,*

$$\sigma^2_{\tau_h}(\mathbf{x}) = \Sigma^2_h(\mathbf{x}), \quad \forall h = 1, \ldots, H_l \tag{D.8}$$

Moreover, we have the two representations (Eq. (D.7) and Eq. (5.11)) for the posterior mean

function are equivalent. That is, for any $\mathbf{x} \in \mathcal{D}$*,*

$$\mu_{T_l}(\mathbf{x}) = \bar{\mu}_l(\mathbf{x}). \tag{D.9}$$

Proof. First, we have the following result, which helps connect the two representations of mean and variance functions.

$$\begin{aligned}
\Phi_{\tau_h}^\top \Phi_{\tau_h} &= \sum_{t=t_l+1}^{t_l+\tau_h} \varphi(\mathbf{x}_t)\varphi(\mathbf{x}_t)^\top \\
&\overset{(a)}{=} \sum_{i=1}^{h} T_l(\mathbf{a}_i)\varphi(\mathbf{a}_i)\varphi(\mathbf{a}_i)^\top \\
&= \Phi_h^\top \mathbf{W}_h \Phi_h,
\end{aligned} \tag{D.10}$$

where (a) is due to our algorithm decisions: $\mathbf{x}_t = \mathbf{a}_i$ for any $t \in \mathcal{T}_l(\mathbf{a}_i) = \{t_l + \tau_{i-1} + 1, t_l + \tau_{i-1} + T_l(\mathbf{a}_i)\}$ and the last step holds because $\mathbf{W}_h$ is a diagonal matrix with $(\mathbf{W}_h)_{ii} = T_l(\mathbf{a}_i)$ for any $i \in [h]$.

Then, we are ready to derive the equivalence of two representations of mean function.

1) Variance representation equivalence: $\sigma^2_{\tau_h}(\mathbf{x}) = \Sigma^2_h(\mathbf{x})$ for $h = 1, \ldots, H_l$. That means,

$$\begin{aligned}
&k(\mathbf{x}, \mathbf{x}) - \mathbf{k}(\mathbf{x}, \mathbf{X}_{\tau_h})^\top (\mathbf{K}_{\mathbf{X}_{\tau_h}\mathbf{X}_{\tau_h}} + \lambda \mathbf{I})^{-1} \mathbf{k}(\mathbf{x}, \mathbf{X}_{\tau_h}) \\
&= k(\mathbf{x}, \mathbf{x}) - \mathbf{k}(\mathbf{x}, \mathbf{A}_h)^\top (\mathbf{K}_{\mathbf{A}_h\mathbf{A}_h} + \lambda \mathbf{W}_h^{-1})^{-1} \mathbf{k}(\mathbf{x}, \mathbf{A}_h).
\end{aligned}$$

It remains to show that

$$\mathbf{k}(\mathbf{x}, \mathbf{X}_{\tau_h})^\top (\mathbf{K}_{\mathbf{X}_{\tau_h}\mathbf{X}_{\tau_h}} + \lambda \mathbf{I})^{-1} \mathbf{k}(\mathbf{x}, \mathbf{X}_{\tau_h}) = \mathbf{k}(\mathbf{x}, \mathbf{A}_h)^\top (\mathbf{K}_{\mathbf{A}_h\mathbf{A}_h} + \lambda \mathbf{W}_h^{-1})^{-1} \mathbf{k}(\mathbf{x}, \mathbf{A}_h). \tag{D.11}$$

Using the feature space formulations, we have

$$
\begin{aligned}
&\mathbf{k}(\mathbf{x}, \mathbf{A}_h)^\top (\mathbf{K}_{\mathbf{A}_h\mathbf{A}_h} + \lambda \mathbf{W}_h^{-1})^{-1} \mathbf{k}(\mathbf{x}, \mathbf{A}_h) \\
=&\varphi(\mathbf{x})^\top \Phi_h^\top (\Phi_h \Phi_h^\top + \lambda \mathbf{W}_h^{-1})^{-1} \Phi_h \varphi(\mathbf{x}) \\
=&\varphi(\mathbf{x})^\top \Phi_h^\top \mathbf{W}_h^{1/2} (\mathbf{W}_h^{1/2} \Phi_h \Phi_h^\top \mathbf{W}_h^{1/2} + \lambda \mathbf{I})^{-1} \mathbf{W}_h^{1/2} \Phi_h \varphi(\mathbf{x}) \\
=&\varphi(\mathbf{x})^\top (\Phi_h^\top \mathbf{W}_h^{1/2} \mathbf{W}_h^{1/2} \Phi_h + \lambda \mathbf{I})^{-1} \Phi_h^\top \mathbf{W}_h^{1/2} \mathbf{W}_h^{1/2} \Phi_h \varphi(\mathbf{x}) \\
=&\varphi(\mathbf{x})^\top (\Phi_h^\top \mathbf{W}_h \Phi_h + \lambda \mathbf{I})^{-1} \Phi_h^\top \mathbf{W}_h \Phi_h \varphi(\mathbf{x}) \\
\overset{(a)}{=}&\varphi(\mathbf{x})^\top (\Phi_{\tau_h}^\top \Phi_{\tau_h} + \lambda \mathbf{I})^{-1} \Phi_{\tau_h}^\top \Phi_{\tau_h} \varphi(\mathbf{x}) \\
=&\varphi(\mathbf{x})^\top \Phi_{\tau_h}^\top (\Phi_{\tau_h} \Phi_{\tau_h}^\top + \lambda \mathbf{I})^{-1} \Phi_{\tau_h} \varphi(\mathbf{x}) \\
=&\mathbf{k}(\mathbf{x}, \mathbf{X}_{\tau_h})^\top (\mathbf{K}_{\mathbf{X}_{\tau_h}\mathbf{X}_{\tau_h}} + \lambda \mathbf{I})^{-1} \mathbf{k}(\mathbf{x}, \mathbf{X}_{\tau_h})
\end{aligned}
\tag{D.12}
$$

where (a) is from Eq. (D.10). Then, we have $\sigma^2_{\tau_h}(\mathbf{x}) = \Sigma^2_h(\mathbf{x})$.

2) Mean representation equivalence: $\mu_{T_l}(\mathbf{x}) = \bar{\mu}_l(\mathbf{x})$, i.e.,

$$
\frac{1}{|U_l|} \sum_{u \in U_l} \mathbf{k}(\mathbf{x}, \mathbf{X}_{T_l})^\top (\mathbf{K}_{\mathbf{X}_{T_l}\mathbf{X}_{T_l}} + \lambda \mathbf{I})^{-1} \mathbf{y}_{l,u} = k(\mathbf{x}, \mathbf{A}_{H_l})^\top (\mathbf{K}_{\mathbf{A}_{H_l}\mathbf{A}_{H_l}} + \lambda \mathbf{W}_{H_l}^{-1})^{-1} \bar{\mathbf{y}}_l. \tag{D.13}
$$

For the last within-phase index $\tau_{H_l} = T_l$, we also have the following result

$$
\begin{aligned}
\frac{1}{|U_l|}\sum_{u\in U_l}\Phi_{T_l}^\top \mathbf{y}_{l,u} &= \frac{1}{|U_l|}\sum_{u\in U_l}\sum_{t=t_l+1}^{t_l+T_l} y_{u,t}\varphi(\mathbf{x}_t) \\
&= \frac{1}{|U_l|}\sum_{u\in U_l}\sum_{h=1}^{H_l}\sum_{t\in\mathcal{T}_l(\mathbf{a}_h)} y_{u,t}\varphi(\mathbf{x}_t) \\
&= \frac{1}{|U_l|}\sum_{u\in U_l}\sum_{h=1}^{H_l}\varphi(\mathbf{a}_h)\sum_{t\in\mathcal{T}_l(\mathbf{a}_h)} y_{u,t} \\
&= \frac{1}{|U_l|}\sum_{u\in U_l}\sum_{h=1}^{H_l}\varphi(\mathbf{a}_h)T_l(\mathbf{a}_h)y_l^u(\mathbf{a}_h) \\
&= \sum_{h=1}^{H_l} T_l(\mathbf{a}_h)y_l(\mathbf{a}_h)\varphi(\mathbf{a}_h) \\
&= \Phi_{H_l}^\top \mathbf{W}_{H_l}\bar{\mathbf{y}}_l.
\end{aligned}
\tag{D.14}
$$

Then, we are ready to derive the equivalence of two representations of mean function.

$$
\begin{aligned}
&\frac{1}{|U_l|}\sum_{u\in U_l}\mathbf{k}(\mathbf{x},\mathbf{X}_{T_l})^\top(\mathbf{K}_{\mathbf{X}_{T_l}\mathbf{X}_{T_l}}+\lambda\mathbf{I})^{-1}\mathbf{y}_{l,u} \\
&= \frac{1}{|U_l|}\sum_{u\in U_l}\varphi(\mathbf{x})^\top\Phi_{T_l}^\top(\Phi_{T_l}\Phi_{T_l}^\top+\lambda\mathbf{I})^{-1}\mathbf{y}_{l,u} \\
&= \frac{1}{|U_l|}\sum_{u\in U_l}\varphi(\mathbf{x})^\top(\Phi_{T_l}^\top\Phi_{T_l}+\lambda\mathbf{I})^{-1}\Phi_{T_l}^\top\mathbf{y}_{l,u} \\
&= \varphi(\mathbf{x})^\top(\Phi_{T_l}^\top\Phi_{T_l}+\lambda\mathbf{I})^{-1}\cdot\frac{1}{|U_l|}\sum_{u\in U_l}\Phi_{T_l}^\top\mathbf{y}_{l,u} \\
&\overset{(a)}{=} \varphi(\mathbf{x})^\top(\Phi_{H_l}^\top\mathbf{W}_{H_l}\Phi_{H_l}+\lambda\mathbf{I})^{-1}\Phi_{H_l}^\top\mathbf{W}_{H_l}\bar{\mathbf{y}}_l \\
&= \varphi(\mathbf{x})^\top(\Phi_{H_l}^\top\mathbf{W}_{H_l}^{1/2}\mathbf{W}_{H_l}^{1/2}\Phi_{H_l}+\lambda\mathbf{I})^{-1}\Phi_{H_l}^\top\mathbf{W}_{H_l}^{1/2}\mathbf{W}_{H_l}^{1/2}\bar{\mathbf{y}}_l \\
&= \varphi(\mathbf{x})^\top((\mathbf{W}_{H_l}^{1/2}\Phi_{H_l})^\top(\mathbf{W}_{H_l}^{1/2}\Phi_{H_l})+\lambda\mathbf{I})^{-1}(\mathbf{W}_{H_l}^{1/2}\Phi_{H_l})^\top\mathbf{W}_{H_l}^{1/2}\bar{\mathbf{y}}_l \\
&= \varphi(\mathbf{x})^\top\Phi_{H_l}^\top\mathbf{W}_{H_l}^{1/2}(\mathbf{W}_{H_l}^{1/2}\Phi_{H_l}\Phi_{H_l}^\top\mathbf{W}_{H_l}^{1/2}+\lambda\mathbf{I})^{-1}\mathbf{W}_{H_l}^{1/2}\bar{\mathbf{y}}_l \\
&= \varphi(\mathbf{x})^\top\Phi_{H_l}^\top(\Phi_{H_l}\Phi_{H_l}^\top+\lambda\mathbf{W}_{H_l}^{-1})^{-1}\bar{\mathbf{y}}_l \\
&= \mathbf{k}(\mathbf{x},\mathbf{A}_{H_l})^\top(\mathbf{K}_{\mathbf{A}_{H_l}\mathbf{A}_{H_l}}+\lambda\mathbf{W}_{H_l}^{-1})^{-1}\bar{\mathbf{y}}_l = \bar{\mu}_l(\mathbf{x}),
\end{aligned}
\tag{D.15}
$$

where (a) is from Eq. (D.10) with $\tau_{H_l} = T_l$ and the result in Eq. (D.14). □

D.2.2 Impact of Batch Schedule Strategy on Posterior Variance

In our batch schedule strategy, the decision $\mathbf{x}_t$ does not change for $T_l(\mathbf{a}_h)$ rounds when starting choosing $\mathbf{a}_h$ after τ_{h-1} rounds within the l-th phase. Applying Lemma D.2 to our setting with $\tau' = \tau_{h-1}$, we obtain the following corollary.

Corollary D.4. *Consider any phase l. Recall that τ_{h-1} is the within-phase time index before starting choosing $\mathbf{a}_h$. Then, give any set of chosen actions $\mathbf{A}_{h-1}$ for the first $h-1$ batches, for any kernel k, any $\mathbf{x} \in \mathcal{D}$, and any $\tau \in [\tau_{h-1}+1, \tau_{h-1}+T_l(\mathbf{a}_h)]$, we have*

$$1 \leqslant \frac{\Sigma_{h-1}(\mathbf{x})}{\sigma_\tau(\mathbf{x})} \leqslant C. \tag{D.16}$$

Proof. Applying Lemma D.2 to our setting, we have

$$1 \leqslant \frac{\sigma^2_{\tau_{h-1}}(\mathbf{x})}{\sigma^2_\tau(\mathbf{x})} \leqslant 1 + T_l(\mathbf{a}_h)\sigma^2_{\tau_h}(\mathbf{a}_h). \tag{D.17}$$

Moreover, by selecting $T_l(\mathbf{a}_h) = \lfloor (C^2-1)/\Sigma^2_{h-1}(\mathbf{a}_h) \rfloor = \lfloor (C^2-1)/\sigma^2_{\tau_{h-1}}(\mathbf{a}_h) \rfloor$ (Lemma D.3) in our algorithm, we derive the result in Eq. (D.16). □

One key step to get the regret upper bound is to bound the confidence width, which is related to the maximal value of the posterior variance by the end of each phase. (See Eq. (5.12). In the following, we provide a bound for the maximal value of the posterior variance.

Lemma D.5. *The posterior variance after H_l batches (decisions) in the l-th phase satisfies*

$$\max_{\mathbf{x}\in\mathcal{D}_l} \Sigma_{H_l}(\mathbf{x}) \leqslant \sqrt{\frac{2\sigma^2 C^2 \gamma_{T_l}}{T_l}} \tag{D.18}$$

Proof. Recall that DPBE plays action $\mathbf{a}_h$ when $\tau \in [\tau_{h-1}+1, \tau_{h-1}+T_l(\mathbf{a}_h)]$ within the l-th phase. First, we have for any $\mathbf{x} \in \mathcal{D}_l$, any $h \leqslant H_l$,

$$\Sigma_{H_l}(\mathbf{x}) \overset{(a)}{\leqslant} \Sigma_{h-1}(\mathbf{x}) \overset{(b)}{\leqslant} \Sigma_{h-1}(\mathbf{a}_h) = \sigma_{\tau_{h-1}}(\mathbf{a}_h), \tag{D.19}$$

where the first inequality (a) holds because $\Sigma_h(\cdot)$ is non-increasing in h, Eq. (b) is based on our decision, and last step from the equivalent representation result. Then, we have the following results:

$$\begin{aligned}
\max_{\mathbf{x}\in\mathcal{D}_l} \Sigma_{H_l}(\mathbf{x}) &\leqslant \frac{1}{T_l}\sum_{h=1}^{H_l} T_l(\mathbf{a}_h)\Sigma_{h-1}(\mathbf{a}_h) \\
&= \frac{1}{T_l}\sum_{h=1}^{H_l}\sum_{\tau=\tau_{h-1}+1}^{\tau_{h-1}+T_l(\mathbf{a}_h)} \Sigma_{h-1}(\mathbf{a}_h) \\
&= \frac{1}{T_l}\sum_{h=1}^{H_l}\sum_{\tau=\tau_{h-1}+1}^{\tau_{h-1}+T_l(\mathbf{a}_h)} \frac{\Sigma_{h-1}(\mathbf{a}_h)}{\sigma_\tau(\mathbf{a}_h)} \cdot \sigma_\tau(\mathbf{a}_h) \\
&\overset{(a)}{\leqslant} \frac{1}{T_l}\sum_{h=1}^{H_l}\sum_{\tau=\tau_{h-1}+1}^{\tau_{h-1}+T_l(\mathbf{a}_h)} C\sigma_\tau(\mathbf{a}_h) \\
&\overset{(b)}{=} \frac{C}{T_l}\sum_{h=1}^{H_l}\sum_{\tau=\tau_{h-1}+1}^{\tau_{h-1}+T_l(\mathbf{a}_h)} \sigma_\tau(\mathbf{x}_{t_l+\tau}) \\
&= \frac{C}{T_l}\sum_{\tau=1}^{T_l} \sigma_\tau(\mathbf{x}_{t_l+\tau}) \\
&\overset{(c)}{\leqslant} \frac{C}{T_l}\sqrt{T_l\sum_{\tau=1}^{T_l}\sigma_\tau^2(\mathbf{x}_{t_l+\tau})} \\
&\overset{(d)}{\leqslant} \frac{C}{T_l}\sqrt{T_l\cdot 2\lambda\gamma_{T_l}} = \sqrt{\frac{2\lambda C^2\gamma_{T_l}}{T_l}},
\end{aligned} \tag{D.20}$$

where the inequality (a) is from Corollary D.4, (b) is based on our algorithm decision: $\mathbf{x}_{t_l+\tau} = \mathbf{a}_h$ for any $\tau \in [\tau_{h-1}+1, \tau_{h-1}+T_l(\mathbf{a}_h)]$, (c) is by Cauchy-Schwartz inequality, and (d) is from

Lemma D.1. □

D.2.3 Other Useful Results

Lemma D.6. *Consider any particular phase l. In the traditional GP models, without noise in the reward observations, the difference between the ground truth and regression estimator satisfies*

$$\left| f(\mathbf{x}) - \mathbf{k}(\mathbf{x}, \mathbf{X}_{T_l})^\top (\mathbf{K}_{\mathbf{X}_{T_l}\mathbf{X}_{T_l}} + \lambda \mathbf{I})^{-1} f(\mathbf{X}_{T_l}) \right| \leqslant B\sigma_{T_l}(\mathbf{x}). \tag{D.21}$$

Proof.

$$\begin{aligned}
&\left| f(\mathbf{x}) - \mathbf{k}(\mathbf{x}, \mathbf{X}_{T_l})^\top (\mathbf{K}_{\mathbf{X}_{T_l}\mathbf{X}_{T_l}} + \lambda \mathbf{I})^{-1} f(\mathbf{X}_{T_l}) \right| \\
&= \left| \varphi(\mathbf{x})^\top f - \varphi(\mathbf{x})^\top \Phi_{T_l}^\top (\Phi_{T_l}\Phi_{T_l}^\top + \lambda \mathbf{I})^{-1} \Phi_{T_l} f \right| \\
&= \left| \varphi(\mathbf{x})^\top f - \varphi(\mathbf{x})^\top (\Phi_{T_l}^\top \Phi_{T_l} + \lambda \mathbf{I})^{-1} \Phi_{T_l}^\top \Phi_{T_l} f \right| \\
&= \left| \lambda \varphi(\mathbf{x})^\top (\Phi_{T_l}^\top \Phi_{T_l} + \lambda \mathbf{I})^{-1} f \right| \\
&\leqslant \|f\|_k \|\lambda (\Phi_{T_l}^\top \Phi_{T_l} + \lambda \mathbf{I})^{-1} \varphi(\mathbf{x})\|_k \\
&\leqslant B\sqrt{\lambda \varphi(\mathbf{x})^\top (\Phi_{T_l}^\top \Phi_{T_l} + \lambda \mathbf{I})^{-1} \lambda \mathbf{I} (\Phi_{T_l}^\top \Phi_{T_l} + \lambda \mathbf{I})^{-1} \varphi(\mathbf{x})} \\
&\leqslant B\sqrt{\lambda \varphi(\mathbf{x})^\top (\Phi_{T_l}^\top \Phi_{T_l} + \lambda \mathbf{I})^{-1} (\Phi_{T_l}^\top \Phi_{T_l} + \lambda \mathbf{I}) (\Phi_{T_l}^\top \Phi_{T_l} + \lambda \mathbf{I})^{-1} \varphi(\mathbf{x})} \\
&\leqslant B\sqrt{\lambda \varphi(\mathbf{x})^\top (\Phi_{T_l}^\top \Phi_{T_l} + \lambda \mathbf{I})^{-1} \varphi(\mathbf{x})} \\
&= B\sigma_{T_l}(\mathbf{x}),
\end{aligned} \tag{D.22}$$

where the last step is from Eq. (D.5) □

D.3 Proofs of Theorem 5.3

Before proving Theorem 5.3, we first provide the key concentration inequality under `DPBE` in Theorem D.7.

Theorem D.7. *For any particular phase l, with probability at least $1-4\beta$, the following holds*

$$|f(\mathbf{x}) - \bar{\mu}_l(\mathbf{x})| \leqslant w_l(\mathbf{x}), \tag{D.23}$$

where mean function $\bar{\mu}_l(\mathbf{x})$ and confidence width function $w_l(\mathbf{x})$ are defined in Eq. (5.11) and Eq. (5.12).

Proof. In this proof, we will show the following concentration inequality holds for any $\mathbf{x} \in \mathcal{D}$

$$\mathbb{P}[|f(\mathbf{x}) - \bar{\mu}_l(\mathbf{x})| \geqslant w_l(\mathbf{x})] \leqslant 4\beta. \tag{D.24}$$

For any $\mathbf{x} \in \mathcal{D}$, we let $w_l(\mathbf{x}) = w_{l,1}(\mathbf{x}) + w_{l,2}(\mathbf{x})$, where

$$w_{l,1}(\mathbf{x}) \triangleq \sqrt{\frac{2k(\mathbf{x},\mathbf{x})\log(1/\beta)}{|U_l|}} \quad \text{and } w_{l,2}(\mathbf{x}) \triangleq \Sigma_{H_l}(\mathbf{x})\left(\sqrt{\frac{2\log(1/\beta)}{|U_l|}} + B\right).$$

First, for any $\mathbf{x} \in \mathcal{D}$, we have the following inequality,

$$|f(\mathbf{x}) - \bar{\mu}_l(\mathbf{x})| \leqslant \left|f(\mathbf{x}) - \frac{1}{|U_l|}\sum_{u\in U_l} f_u(\mathbf{x})\right| + \left|\frac{1}{|U_l|}\sum_{u\in U_l} f_u(\mathbf{x}) - \bar{\mu}_l(\mathbf{x})\right|.$$

Then, we have

$$\begin{aligned}
&\mathbb{P}\left[|f(\mathbf{x}) - \bar{\mu}_l(\mathbf{x})| \geqslant w_l(\mathbf{x})\right] \\
\leqslant &\mathbb{P}\left[\left|f(\mathbf{x}) - \frac{1}{|U_l|}\sum_{u\in U_l} f_u(\mathbf{x})\right| + \left|\frac{1}{|U_l|}\sum_{u\in U_l} f_u(\mathbf{x}) - \bar{\mu}_l(\mathbf{x})\right| \geqslant w_{l,1}(\mathbf{x}) + w_{l,2}(\mathbf{x})\right] \\
\leqslant &\mathbb{P}\left[\left|f(\mathbf{x}) - \frac{1}{|U_l|}\sum_{u\in U_l} f_u(\mathbf{x})\right| \geqslant w_{l,1}(\mathbf{x})\right] + \mathbb{P}\left[\left|\frac{1}{|U_l|}\sum_{u\in U_l} f_u(\mathbf{x}) - \bar{\mu}_l(\mathbf{x})\right| \geqslant w_{l,2}(\mathbf{x})\right],
\end{aligned} \tag{D.25}$$

where the last inequality is from union bound.

In the following, we try to bound the above two terms, respectively.

i) Recall that each user u is associated with a local reward function $f_u \sim \mathcal{GP}(f(\cdot), k(\cdot))$. Hence,

$$f_u(\mathbf{x}) \sim \mathcal{N}(f(\mathbf{x}), k(\mathbf{x}, \mathbf{x})), \quad \forall \mathbf{x} \in \mathcal{D}. \tag{D.26}$$

Note that U_l is a set of $\lceil 2^{\alpha l} \rceil$ random users independently sampled from the population, and then we have

$$\frac{1}{|U_l|} \sum_{u \in U_l} f_u(\mathbf{x}) \sim \mathcal{N}\left(f(\mathbf{x}), \frac{k(\mathbf{x}, \mathbf{x})}{|U_l|}\right), \quad \forall \mathbf{x} \in \mathcal{D}.$$

Combining the concentration inequality for Gaussian random variables, we have

$$\mathbb{P}\left[\left|\frac{1}{|U_l|} \sum_{u \in U_l} f_u(\mathbf{x}) - f(\mathbf{x})\right| \geqslant w_{l,1}(\mathbf{x})\right] \leqslant 2 \exp\left(-\frac{|U_l| w_{l,1}^2(\mathbf{x})}{2k(\mathbf{x}, \mathbf{x}))}\right) = 2\beta \tag{D.27}$$

ii) Then, we want to bound the second term in Eq. (D.25).

$$\begin{aligned}
&\mathbb{P}\left[\left|\frac{1}{|U_l|} \sum_{u \in U_l} f_u(\mathbf{x}) - \bar{\mu}_l(\mathbf{x})\right| \geqslant w_{l,2}(\mathbf{x})\right] \\
&= \sum_{\Lambda} \mathbb{P}[\Lambda = \{\mathbf{y}_{l,u}\}_{u \in U_l}] \cdot \mathbb{P}\left[\left|\frac{1}{|U_l|} \sum_{u \in U_l} f_u(\mathbf{x}) - \bar{\mu}_l(\mathbf{x})\right| \geqslant w_{l,2}(\mathbf{x}) \middle| \{\mathbf{y}_{l,u}\}_{u \in U_l}\right]
\end{aligned}$$

where $\mathbf{y}_{l,u} = [y_{u,t_l+1}, \ldots, y_{u,t_l+T_l}]^\top$ denotes the realization of the local reward observations at user u in the l-th phase. According to our assumption, the participant user u is associated with a local reward function f_u sampled from Gaussian Process $\mathcal{GP}(f(\cdot), k(\cdot, \cdot))$. Given the points $\mathbf{X}_{T_l} = [x_{t_l+1}^\top, \ldots, \mathbf{x}_{t_l+T_l}^\top]^\top$ in $\mathcal{D}$, the corresponding vector of local rewards $\mathbf{y}_{l,u} = [y_{u,t_l+1}, \ldots, y_{u,t_l+T_l}]^\top$ has the multivariate Gaussian distribution $\mathcal{N}(f(\mathbf{X}_{T_l}), (\mathbf{K}_{\mathbf{X}_{T_l}\mathbf{X}_{T_l}} + \lambda \mathbf{I}))$ where $f(\mathbf{X}_{T_l}) = [f(\mathbf{x}_{t_l+1}), \cdots, f(\mathbf{x}_{t_l+T_l})]^\top$ and $\mathbf{K}_{\mathbf{X}_{T_l}\mathbf{X}_{T_l}} = [k(\mathbf{x}, \mathbf{x}')]_{\mathbf{x}, \mathbf{x}' \in \mathbf{X}_{T_l}}$ is the kernel matrix for the T_l selected actions in the l-th phase. Based on the properties of GPs, we have

that $\mathbf{y}_{l,u}$ and $f_u(\mathbf{x})$ are jointly Gaussian given $\mathbf{X}_{T_l}$:

$$\begin{bmatrix} f_u(\mathbf{x}) \\ \mathbf{y}_{l,u} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} f(\mathbf{x}) \\ f(\mathbf{X}_{T_l}) \end{bmatrix}, \begin{bmatrix} k(\mathbf{x},\mathbf{x}) & \mathbf{k}(\mathbf{x},\mathbf{X}_{T_l})^\top \\ \mathbf{k}(\mathbf{x},\mathbf{X}_{T_l}) & \mathbf{K}_{\mathbf{X}_{T_l}\mathbf{X}_{T_l}} + \lambda \mathbf{I} \end{bmatrix} \right), \tag{D.28}$$

where $k(\mathbf{x},\mathbf{X}_{T_l}) = [k(\mathbf{x},\mathbf{x}_{t_l+1}),\ldots,k(\mathbf{x},\mathbf{x}_{t_l+T_l})]^\top$. According to the basic formula for conditional distributions of Gaussian random vectors (see [112, Appendix A.2], or [75, Proposition 3.2]), we have, conditioned on $\mathbf{y}_{l,u}$ corresponding to the points $\mathbf{X}_{T_l}$,

$$f_u(\mathbf{x})|\mathbf{y}_{l,u} \sim \mathcal{N}(m_u(\mathbf{x}), \sigma^2_{T_l}(\mathbf{x})),$$

where

$$m_u(\mathbf{x}) \triangleq f(\mathbf{x}) + \mathbf{k}(\mathbf{x},\mathbf{X}_{T_l})^\top (\mathbf{K}_{\mathbf{X}_{T_l}\mathbf{X}_{T_l}} + \lambda\mathbf{I})^{-1}(\mathbf{y}_{l,u} - f(\mathbf{X}_{T_l})), \tag{D.29}$$

$$\sigma^2_{T_l}(\mathbf{x}) = k(\mathbf{x},\mathbf{x}) - \mathbf{k}(\mathbf{x},\mathbf{X}_{T_l})^\top (\mathbf{K}_{\mathbf{X}_{T_l}\mathbf{X}_{T_l}} + \lambda\mathbf{I})^{-1}\mathbf{k}(\mathbf{x},\mathbf{X}_{T_l}). \tag{D.30}$$

Note that we sample the participants U_l independently and that the local reward noise is also independent across participants. Then, we have the following result:

$$\left(\frac{1}{|U_l|}\sum_{u\in U_l} f_u(\mathbf{x})\right) \Big| \{\mathbf{y}_{l,u}\}_{u\in U_l} = \frac{1}{|U_l|}\sum_{u\in U_l}(f_u(\mathbf{x}) \mid \mathbf{y}_{l,u}) \sim \mathcal{N}\left(\frac{1}{|U_l|}\sum_{u\in U_l} m_u(\mathbf{x}), \frac{\sigma^2_{T_l}(\mathbf{x})}{|U_l|}\right).$$

Combining the Gaussian concentration inequality, we have the following result

$$\mathbb{P}\left[\left|\frac{1}{|U_l|}\sum_{u\in U_l} f_u(\mathbf{x}) - \frac{1}{|U_l|}\sum_{u\in U_l} m_u(\mathbf{x})\right| \geqslant \sqrt{\frac{2\sigma^2_{T_l}(\mathbf{x})\log(1/\beta)}{|U_l|}} \Bigg| \{\mathbf{y}_{l,u}\}_{u\in U_l}\right] \leqslant 2\beta. \tag{D.31}$$

From Lemma D.3, we have the following equation:

$$\frac{1}{|U_l|}\sum_{u\in U_l}\mathbf{k}(\mathbf{x},\mathbf{X}_{T_l})^\top(\mathbf{K}_{\mathbf{X}_{T_l}\mathbf{X}_{T_l}}+\lambda\mathbf{I})^{-1}\mathbf{y}_{l,u}=\mathbf{k}(\mathbf{x},\mathbf{A}_h)^\top(\mathbf{K}_{\mathbf{A}_h\mathbf{A}_h}+\lambda\mathbf{W}_h^{-1})^{-1}\bar{\mathbf{y}}_l=\bar{\mu}_l(\mathbf{x}), \tag{D.32}$$

which implies

$$\frac{1}{|U_l|}\sum_{u\in U_l}m_u(\mathbf{x})=\bar{\mu}_l(\mathbf{x})+f(\mathbf{x})-\mathbf{k}(\mathbf{x},\mathbf{X}_{T_l})^\top(\mathbf{K}_{\mathbf{X}_{T_l}\mathbf{X}_{T_l}}+\lambda\mathbf{I})^{-1}f(\mathbf{X}_{T_l}). \tag{D.33}$$

Then, the gap between the average local function $\frac{1}{|U_l|}\sum_{u\in U_l}f_u(\cdot)$ and the estimator $\bar{\mu}_l(\cdot)$ satisfies

$$\begin{aligned}
&\left|\frac{1}{|U_l|}\sum_{u\in U_l}f_u(\mathbf{x})-\bar{\mu}_l(\mathbf{x})\right|\\
\leqslant&\left|\frac{1}{|U_l|}\sum_{u\in U_l}f_u(\mathbf{x})-\frac{1}{|U_l|}\sum_{u\in U_l}m_u(\mathbf{x})\right|+\left|f(\mathbf{x})-\mathbf{k}(\mathbf{x},\mathbf{X}_{T_l})^\top(\mathbf{K}_{\mathbf{X}_{T_l}\mathbf{X}_{T_l}}+\lambda\mathbf{I})^{-1}f(\mathbf{X}_{T_l})\right|\\
\overset{(a)}{\leqslant}&\left|\frac{1}{|U_l|}\sum_{u\in U_l}f_u(\mathbf{x})-\frac{1}{|U_l|}\sum_{u\in U_l}m_u(\mathbf{x})\right|+B\sigma_{T_l}(\mathbf{x}),
\end{aligned} \tag{D.34}$$

where (a) is from Lemma D.6. Combining the result in Eq. (D.31), we have

$$\begin{aligned}
&\mathbb{P}\left[\left|\frac{1}{|U_l|}\sum_{u\in U_l}f_u(\mathbf{x})-\bar{\mu}_l(\mathbf{x})\right|\geqslant w_{l,2}(\mathbf{x})\middle|\{\mathbf{y}_{l,u}\}_{u\in U_l}\right]\\
\leqslant&\mathbb{P}\left[\left|\frac{1}{|U_l|}\sum_{u\in U_l}f_u(\mathbf{x})-\frac{1}{|U_l|}\sum_{u\in U_l}m_u(\mathbf{x})\right|+B\sigma_{T_l}(x_0)\geqslant w_{l,2}(\mathbf{x})\middle|\{\mathbf{y}_{l,u}\}_{u\in U_l}\right]\\
\overset{(a)}{=}&\mathbb{P}\left[\left|\frac{1}{|U_l|}\sum_{u\in U_l}f_u(\mathbf{x})-\frac{1}{|U_l|}\sum_{u\in U_l}m_u(\mathbf{x})\right|\geqslant\sqrt{\frac{2\sigma_{T_l}^2(\mathbf{x})\log(1/\beta)}{|U_l|}}\middle|\{\mathbf{y}_{l,u}\}_{u\in U_l}\right]\leqslant 2\beta,
\end{aligned} \tag{D.35}$$

where (a) is from $\sigma_{T_l}^2(\mathbf{x})=\Sigma_{H_l}^2(\mathbf{x})$ according to Lemma D.3. Therefore, we derive the desired

result

$$
\begin{aligned}
&\mathbb{P}\left[\left|\frac{1}{|U_l|}\sum_{u\in U_l} f_u(\mathbf{x}) - \bar{\mu}_l(\mathbf{x})\right| \geqslant w_{l,2}(\mathbf{x})\right] \\
=&\sum_{\Lambda} \mathbb{P}[\Lambda = \{\mathbf{y}_{l,u}\}_{u\in U_l}] \cdot \mathbb{P}\left[\left|\frac{1}{|U_l|}\sum_{u\in U_l} f_u(\mathbf{x}) - \bar{\mu}_l(\mathbf{x})\right| \geqslant w_{l,2}(\mathbf{x}) \middle| \{\mathbf{y}_{l,u}\}_{u\in U_l}\right] \\
\leqslant&\sum_{\Lambda} \mathbb{P}[\Lambda = \{\mathbf{y}_{l,u}\}_{u\in U_l}] \cdot 2\beta = 2\beta.
\end{aligned}
\tag{D.36}
$$

□

To prove Theorem 5.3, we first present three main conclusions when the concentration inequality in Theorem D.7 holds, then get an upper bound for the regret incurred in a particular phase l with high probability, and finally sum up the regret over all phases.

Define a "good" event when Eq. (D.23) holds in the l-th phase as:

$$
\mathcal{E}_l \triangleq \{\forall \mathbf{x} \in \mathcal{D}_l, |f(\mathbf{x}) - \bar{\mu}_l(\mathbf{x})| \leqslant w_l(\mathbf{x})\}.
$$

We have $\mathbb{P}[\mathcal{E}_l] \geqslant 1 - 4|\mathcal{D}|\beta$ via the union bound. Then, under event $\mathcal{E}_l$ in the l-th phase, we have the following three observations:

1. For any optimal action $\mathbf{x}^* \in \operatorname{argmax}_{\mathbf{x}\in\mathcal{D}} f(\mathbf{x})$, if $\mathbf{x}^* \in \mathcal{D}_l$, then $\mathbf{x}^* \in \mathcal{D}_{l+1}$.

2. Let $f^* = \max_{\mathbf{x}\in\mathcal{D}} f(\mathbf{x})$. Supposed that $x^* \in \mathcal{D}_l$. For any $\mathbf{x} \in \mathcal{D}_{l+1}$, its reward gap from the optimal reward is bounded by $4\max_{\mathbf{x}\in\mathcal{D}_l} w_l(\mathbf{x})$, i.e.,

$$
f^* - f(\mathbf{x}) \leqslant 4\max_{\mathbf{x}\in\mathcal{D}_l} w_l(\mathbf{x}).
$$

3. The confidence width function satisfies

$$\max_{\mathbf{x}\in\mathcal{D}_l} w_l(\mathbf{x}) \leqslant \sqrt{\frac{2\kappa^2\log(1/\beta)}{|U_l|}} + \sqrt{\frac{4\sigma^2C^2\gamma_{T_l}\log(1/\beta)}{T_l|U_l|}} + \sqrt{\frac{2\sigma^2B^2C^2\gamma_{T_l}}{T_l}}$$

Proof. **Observation 1:** Let $\mathbf{b} \in \operatorname{argmax}_{\mathbf{x}\in\mathcal{D}_l}(\bar{\mu}_l(\mathbf{x}) - w_l(\mathbf{x}))$. Then under event $\mathcal{E}_l$, we have

$$\bar{\mu}_l(\mathbf{x}^*) + w_l(\mathbf{x}^*) \geqslant f(\mathbf{x}^*) \geqslant f(\mathbf{b}) \geqslant \bar{\mu}_l(\mathbf{b}) - w_l(\mathbf{b}) \tag{D.37}$$

which indicates $\mathbf{x}^* \in \mathcal{D}_{l+1}$ according to Eq. (5.10).

Observation 2: For any $\mathbf{x} \in \mathcal{D}_{l+1}$, we have $\mathbf{x} \in \mathcal{D}_l$ and

$$\bar{\mu}_l(\mathbf{x}) + w_l(\mathbf{x}) \geqslant \bar{\mu}_l(\mathbf{b}) - w_l(\mathbf{b}) \geqslant \bar{\mu}_l(\mathbf{x}^*) - w_l(\mathbf{x}^*). \tag{D.38}$$

Then, we have the regret of choosing any action $\mathbf{x} \in \mathcal{D}_{l+1}$ satisfying

$$\begin{aligned} f(\mathbf{x}^*) - f(\mathbf{x}) &\overset{(a)}{\leqslant} \bar{\mu}_l(\mathbf{x}^*) + w_l(\mathbf{x}^*) - \bar{\mu}_l(\mathbf{x}) + w_l(\mathbf{x}) \\ &\overset{(b)}{\leqslant} 2(w_l(\mathbf{x}) + w_l(\mathbf{x}^*)) \\ &\leqslant 4\max_{\mathbf{x}\in\mathcal{D}_l} w_l(\mathbf{x}), \end{aligned} \tag{D.39}$$

where (a) holds under event $\mathcal{E}_l$ and the second inequality (b) is from Eq. (D.38). Then, we derive Observation 2.

Observation 3: Based on the result in Lemma D.5, we have

$$\begin{aligned}
\max_{\mathbf{x}\in\mathcal{D}_l} w_l(\mathbf{x}) &= \max_{\mathbf{x}\in\mathcal{D}_l}\left(\sqrt{\frac{2k(\mathbf{x},\mathbf{x})\log(1/\beta)}{|U_l|}} + \Sigma_{H_l}(\mathbf{x})\left(\sqrt{\frac{2\log(1/\beta)}{|U_l|}} + B\right)\right) \\
&\leqslant \sqrt{\frac{2\kappa^2\log(1/\beta)}{|U_l|}} + \max_{\mathbf{x}\in\mathcal{D}_l}\Sigma_{H_l}(\mathbf{x})\left(\sqrt{\frac{2\log(1/\beta)}{|U_l|}} + B\right) \\
&\leqslant \sqrt{\frac{2\kappa^2\log(1/\beta)}{|U_l|}} + \sqrt{\frac{4\lambda C^2\gamma_{T_l}\log(1/\beta)}{T_l|U_l|}} + \sqrt{\frac{2\lambda B^2C^2\gamma_{T_l}}{T_l}} \\
&= \sqrt{\frac{2\kappa^2\log(1/\beta)}{|U_l|}} + \sqrt{\frac{4\sigma^2C^2\gamma_{T_l}\log(1/\beta)}{T_l|U_l|}} + \sqrt{\frac{2\sigma^2B^2C^2\gamma_{T_l}}{T_l}}.
\end{aligned} \tag{D.40}$$

□

Then, we are ready to prove Theorem 5.3.

Proof of Theorem 5.3. Let the regret in the l-th phase be $r_l \triangleq \sum_{t\in\mathcal{T}_l}(\max_{\mathbf{x}\in\mathcal{D}} f(\mathbf{x}) - f(\mathbf{x}_t))$. For any $l \geqslant 2$, we assume event $\mathcal{E}_{l-1}$ holds. Then, we have the following result

$$\begin{aligned}
r_l &= \sum_{t\in\mathcal{T}_l}(\max_{\mathbf{x}\in\mathcal{D}} f(\mathbf{x}) - f(\mathbf{x})) \\
&\leqslant \sum_{t\in\mathcal{T}_l} 4\max_{\mathbf{x}\in\mathcal{D}_{l-1}} w_{l-1}(\mathbf{x}) \\
&\leqslant 4T_l \max_{\mathbf{x}\in\mathcal{D}_{l-1}} w_{l-1}(\mathbf{x}) \\
&\leqslant 4T_l\left(\sqrt{\frac{2\kappa^2\log(1/\beta)}{|U_{l-1}|}} + \sqrt{\frac{4\sigma^2C^2\gamma_{T_{l-1}}\log(1/\beta)}{T_{l-1}|U_{l-1}|}} + \sqrt{\frac{2\sigma^2B^2C^2\gamma_{T_{l-1}}}{T_{l-1}}}\right) \\
&\overset{(a)}{\leqslant} 4\cdot 2^{l-1}\left(\sqrt{\frac{2\kappa^2\log(1/\beta)}{2^{\alpha(l-1)}}} + \sqrt{\frac{4\sigma^2C^2\gamma_T\log(1/\beta)}{2^{(1+\alpha)(l-1)-1}}} + \sqrt{\frac{2\sigma^2B^2C^2\gamma_T}{2^{l-2}}}\right) \\
&\leqslant 4\sqrt{2\kappa^2\log(1/\beta)}\sqrt{2^{(2-\alpha)(l-1)}} + 8\sigma C\sqrt{2\gamma_T\log(1/\beta)}\sqrt{2^{(1-\alpha)(l-1)}} + 8\sigma BC\sqrt{\gamma_T 2^{l-1}},
\end{aligned} \tag{D.41}$$

where (a) is from $\gamma_{T_{l-1}} \leqslant \gamma_T$ and $|U_l| \geqslant 2^{\alpha l}$.

Define $\mathcal{E}_g$ as the event where the "good" event occurs in every phase, i.e., $\mathcal{E}_g \triangleq \bigcap_{l=1}^{L} \mathcal{E}_l$. It is not difficult to obtain $\mathbb{P}[\mathcal{E}_g] \geqslant 1 - 4|\mathcal{D}|\beta L$ by applying union bound. At the same time, let R_g be the regret under event $\mathcal{E}_g$, and R_b be the regret if event $\mathcal{E}_g$ does not hold. Then, the expected total regret in T is $\mathbb{E}[R(T)] = \mathbb{P}[\mathcal{E}_g]R_g + (1 - \mathbb{P}[\mathcal{E}_g])R_b$.

Under event $\mathcal{E}_g$, the regret in the l-th phase r_l satisfies Eq. (D.41) for any $l \geqslant 2$. Note that $r_1 \leqslant 2T_1 B\kappa \leqslant 2B\kappa$ since $T_1 = 1$ and for any $\mathbf{x} \in \mathcal{D}$

$$|f(\mathbf{x})| = |\langle f, k(\mathbf{x}, \cdot)\rangle_k| \leqslant \|f\|_k \langle k(\mathbf{x}, \cdot), k(\mathbf{x}, \cdot)\rangle_k^{1/2} \leqslant Bk(\mathbf{x}, \mathbf{x})^{1/2} \leqslant B\kappa.$$

Then, we have

$$\begin{aligned}
R_g &= \sum_{l=1}^{L} r_l \\
&\leqslant 2B\kappa + \sum_{l=2}^{L} 4\sqrt{2\kappa^2 \log(1/\beta)}\sqrt{2^{(2-\alpha)(l-1)}} \\
&+ \sum_{l=2}^{L} 8\sigma C\sqrt{2\gamma_T \log(1/\beta)}\sqrt{2^{(1-\alpha)(l-1)}} \\
&+ \sum_{l=2}^{L} 8\sigma BC\sqrt{\gamma_T 2^{l-1}} \\
&\leqslant 2B\kappa + 4\sqrt{2\kappa^2 \log(1/\beta)} \cdot 4\sqrt{2^{(L-1)(2-\alpha)}} \\
&+ 8\sigma C\sqrt{2\gamma_T \log(1/\beta)} \cdot C_1\sqrt{2^{(1-\alpha)(L-1)}} \qquad \left(C_1 = \sqrt{2^{1-\alpha}}/(\sqrt{2^{1-\alpha}} - 1)\right) \\
&+ 8\sigma BC\sqrt{\gamma_T} \cdot 4\sqrt{2^{L-1}} \\
&\overset{(a)}{\leqslant} 2B\kappa + 16\sqrt{2\kappa^2 \log(1/\beta)}T^{1-\alpha/2} + 8\sigma C_1 C\sqrt{2\gamma_T \log(1/\beta)T^{1-\alpha}} + 32\sigma BC\sqrt{\gamma_T T},
\end{aligned} \tag{D.42}$$

where the last step is due to $2^{L-1} \leqslant T$ and $L \leqslant \log(2T)$ since $\sum_{l=1}^{L-1} T_l + 1 \leqslant T$.

On the other hand, $R_b \leqslant 2B\kappa T$ since $|\max_{\mathbf{x}\in\mathcal{D}} f(\mathbf{x}) - f(\mathbf{x})| \leqslant 2B\kappa$ for all $\mathbf{x} \in \mathcal{D}$. Choose

$\beta = 1/(|\mathcal{D}|T)$ in Algorithm 6. Finally, we have the following results:

$$
\begin{aligned}
&\mathbb{E}[R(T)] \\
&= \mathbb{P}[\mathcal{E}_g]R_g + (1 - \mathbb{P}[\mathcal{E}_g])R_b \\
&\leqslant R_g + 4|\mathcal{D}|\beta L \cdot 2B\kappa T \\
&\leqslant 2B\kappa + 16\sqrt{2\kappa^2 \log(1/\beta)}T^{1-\alpha/2} + 8\sigma C_1 C\sqrt{2\gamma_T \log(1/\beta)T^{1-\alpha}} + 32\sigma BC\sqrt{\gamma_T T} \\
&+ 8B\kappa|\mathcal{D}|\beta LT \\
&= 2B\kappa + 16T^{1-\alpha/2}\sqrt{2\kappa^2 \log(|\mathcal{D}|T)} + 8\sigma C_1 C\sqrt{2\gamma_T T^{1-\alpha} \log(|\mathcal{D}|T)} \\
&+ 32\sigma BC\sqrt{\gamma_T T} + 8B\kappa \log(2T) \\
&= O(T^{1-\alpha/2}\sqrt{\log(|\mathcal{D}|T)}) + O(\sqrt{\gamma_T T^{1-\alpha} \log(|\mathcal{D})T)} + O(\sqrt{\gamma_T T}).
\end{aligned}
\tag{D.43}
$$

□

D.4 Proofs for Communication and Computation Results

The results regarding computation complexity and communication cost highly depend on the number of batches H_l in each phase l. Hence, we first provide the proof for Lemma 5.8.

Proof of Lemma 5.8. To bound the number of batches in the l-th phase, we follow a similar line to the proof of Lemma 4.3 in [23]. For any $1 \leqslant h \leqslant H_l$, we have

$$
\begin{aligned}
T_l(\mathbf{a}_h) = \left\lfloor \frac{C^2 - 1}{\Sigma_{h-1}^2(\mathbf{a}_h)} \right\rfloor &\geqslant \frac{C^2 - 1}{\Sigma_{h-1}^2(\mathbf{a}_h)} - 1 \\
\Rightarrow \quad \Sigma_{h-1}^2(\mathbf{a}_h)(T_l(\mathbf{a}_h) + 1) &\geqslant C^2 - 1 \\
\Rightarrow \quad 2\Sigma_{h-1}^2(\mathbf{a}_h)T_l(\mathbf{a}_h) &\geqslant C^2 - 1.
\end{aligned}
\tag{D.44}
$$

Recall that we use τ_h to denote the last within-phase time index in the h-th batch. Then, summing the above inequality across all batches up to H_l, we have

$$\begin{aligned}
H_l(C^2-1) &\leqslant \sum_{h=1}^{H_l} 2\Sigma_{h-1}^2(\mathbf{a}_h)T_l(\mathbf{a}_h) \\
&\leqslant 2\sum_{h=1}^{H_l}\sum_{\tau=\tau_{h-1}+1}^{\tau_{h-1}+T_l(\mathbf{a}_h)} \Sigma_{h-1}^2(\mathbf{a}_h) \\
&= 2\sum_{h=1}^{H_l}\sum_{\tau=\tau_{h-1}+1}^{\tau_{h-1}+T_l(\mathbf{a}_h)} \frac{\Sigma_{h-1}^2(\mathbf{a}_h)}{\sigma_\tau^2(\mathbf{a}_h)}\cdot\sigma_\tau^2(\mathbf{a}_h) \\
&\overset{(a)}{\leqslant} 2\sum_{h=1}^{H_l}\sum_{\tau=\tau_{h-1}+1}^{\tau_{h-1}+T_l(\mathbf{a}_h)} C^2\cdot\sigma_\tau^2(\mathbf{a}_h) \\
&\overset{(b)}{=} 2C^2\sum_{h=1}^{H_l}\sum_{\tau=\tau_{h-1}+1}^{\tau_{h-1}+T_l(\mathbf{a}_h)} \sigma_\tau^2(\mathbf{x}_{t_l+\tau}) \\
&= 2C^2\sum_{\tau=1}^{T_l}\sigma_\tau^2(\mathbf{x}_{t_l+\tau}) \\
&\overset{(c)}{\leqslant} 4\sigma^2C^2\gamma_{T_l},
\end{aligned} \tag{D.45}$$

where (a) is from Corollary D.4, (b) is based on our algorithm decision: $\mathbf{x}_{t_l+\tau}=\mathbf{a}_h$ for any $\tau\in[\tau_{h-1}+1,\tau_{h-1}+T_l(\mathbf{a}_h)]$, (c) is from Lemma D.1 where $\mathbf{X}_{T_l}=[\mathbf{x}_{t_l+1}^\top,\ldots,\mathbf{x}_{t_l+T_l}^\top]^\top$ for any phase l and $\lambda=\sigma^2$. Hence, we derive

$$H_l \leqslant \frac{4\sigma^2C^2}{C^2-1}\gamma_{T_l}. \tag{D.46}$$

□

We already analyze how to derive the computation complexity for `DPBE` in Remark 5.9. In the following, we prove Theorem 5.10, which tells the result regarding communication cost: $O(\gamma_T T^\alpha)$.

Proof of Theorem 5.10. Note that the communicating data in each phase between participants and the agent is the local average performance $y_l^u(\mathbf{a})$ for each action $\mathbf{a}$ chosen in the corresponding batch. That is, $N_{u,l} \leqslant H_l$ for every participant u. (Here, the inequality holds when merging batches as Remark 5.1). Combining the bound of H_l in Lemma 5.8, we derive the total communication cost satisfying

$$\sum_{l=1}^{L} |U_l| H_l \leqslant \sum_{l=1}^{L} \frac{4\sigma^2 C^2}{C^2-1} \gamma_{T_l} \cdot (2^{\alpha l}+1) = O\left(\frac{\sigma^2 C^2}{C^2-1} \cdot \gamma_T T^{\alpha}\right), \tag{D.47}$$

where the last step is due to $2^{L-1} \leqslant T$ and $L \leqslant \log(2T)$ since $\sum_{l=1}^{L-1} T_l + 1 \leqslant T$. □

D.5 Differential Private DPBE Extensions

In this section, we extend the differentially private DPBE in Section 5.6 to two other celebrated DP models: the local model and shuffle model.

To begin with, we present the details of the DP-DPBE in the central model mentioned in Section 5.6 in Algorithm 10. Recall that in the central DP model, with a trusted agent, data privacy is protected by privatizing the aggregated feedback so that the output of the algorithm is indistinguishable between any two users. In a particular phase l, the aggregated feedback for each chosen action becomes $\tilde{\mathbf{y}}_l = \bar{\mathbf{y}}_l + (\rho_1, \ldots, \rho_{H_l})$ (Eq. (5.15)), where $\rho_j \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma_{nc}^2)$ is the injected privacy noise. In addition, the confidence width function is updated accordingly (Eq. (5.17)) by taking this privacy noise into account.

Differentially Private DPBE in the Local DP Model. In the local model, the users do not trust the agent, and thus, each is equipped with a local randomizer $\mathcal{R}$ to protect its own local reward. Formally, a local randomizer $\mathcal{R}$ is (ε, δ)-local differentially private ((ε, δ)-LDP) if for any two user inputs, the probability that $\mathcal{R}$ outputs a value in any set Y is not different

by more than a multiplicative factor of e^{ε} and an additive factor of δ. To guarantee LDP, the the local randomizer $\mathcal{R}$ at each user u injects Gaussian noise before sending the local reward observations out to the central agent. That is,

$$\mathcal{R}(\mathbf{y}_l^u) = \mathbf{y}_l^u + (\rho_{u,1}, \ldots, \rho_{u,H_l}) \tag{D.51}$$

and then the *private* aggregated feedback for the chosen actions in the l-th phase in the local DP model becomes

$$\tilde{\mathbf{y}}_l = \frac{1}{|U_l|} \sum_{u \in U_l} \mathcal{R}(\mathbf{y}_l^u) = \frac{1}{|U_l|} \sum_{u \in U_l} (\mathbf{y}_l^u + (\rho_{u,1}, \ldots, \rho_{u,H_l})) \tag{D.52}$$

where $\rho_{u,j} \sim \mathcal{N}(0, \sigma_{nl}^2)$ is *i.i.d.* across both users and actions, and the variance σ_{nl}^2 is chosen according to the sensitivity of $\mathbf{y}_l^u$.

Differentially Private `DPBE` in the Shuffle DP Model. While local DP provides a more stringent privacy guarantee, it usually incurs larger regret cost [144]. The shuffle model is recently proposed to achieve a better tradeoff between regret and privacy [34]. In the shuffle model, between the users and the agent, there exists a shuffler that permutes the local feedback from the participants before they are observed by the agent so that the agent cannot distinguish between two users' feedback. Thus, an additional layer of randomness is introduced via shuffling, which can often be easily implemented using Cryptographic primitives (e.g., mixnets) due to its simple operation [14]. Specifically, the shuffle DP model consists of three components: a local randomizer $\mathcal{R}$ at each user side, a shuffler $\mathcal{S}$ between the users and the agent, and an analyzer $\mathcal{A}$ at the agent side. Let $\mathcal{U}_T \triangleq (U_l, \cdots, U_l)$ be the participants throughout the T rounds. Define the (composite) mechanism $\mathcal{M}_s(\mathcal{U}_T) \triangleq ((\mathcal{S} \circ \mathcal{R})(U_1), (\mathcal{S} \circ \mathcal{R})(U_2), \ldots, (\mathcal{S} \circ \mathcal{R})(U_L))$, where $(\mathcal{S} \circ \mathcal{R})(U_l) \triangleq \mathcal{S}(\{\mathcal{R}(\mathbf{y}_l^u)\}_{u \in U_l})$. Formally, We say the `DP-DPBE` algorithm satisfies the shuffle differential privacy (SDP) if the composite

mechanism $\mathcal{M}_s$ is DP, which leads to the following formal definition.

Definition D.8. (Shuffle Differential Privacy (SDP)). For any $\varepsilon \geqslant 0$ and $\delta \in [0,1]$, the `DP-DPBE` is (ε, δ)-shuffle differential privacy (or (ε, δ)-SDP) if for any pair $\mathcal{U}_T$ and $\mathcal{U}_T^{'}$ that differ by one user, and for any $Z \in Range(\mathcal{M}_s)$[1]:

$$\mathbb{P}[\mathcal{M}_s(\mathcal{U}_T) \in Z] \leqslant e^{\varepsilon}\mathbb{P}[\mathcal{M}_s(\mathcal{U}_T^{'}) \in Z] + \delta. \tag{D.53}$$

In our case, we apply a shuffle model to the feedback from participants of every particular phase. That is, the *private* aggregated feedback for the chosen actions in the l-th phase in the shuffle DP model becomes

$$\tilde{\mathbf{y}}_l = \mathcal{A}\left(\mathcal{S}\left(\{\mathcal{R}(\mathbf{y}_l^u\}_{u \in U_l}\right)\right) \tag{D.54}$$

where the local randomizer injects a sub-Gaussian noise with variance σ_{ns}^2, which is *i.i.d.* across both users and actions. Thanks to our phase-then-batch strategy, the recently proposed vector summation protocol [35] can be extended to our algorithm as [89]. We present the concrete pseudocodes of $\mathcal{R}$, $\mathcal{S}$, and $\mathcal{A}$ in Algorithm 11.

D.5.1 Performance Guarantee

For the `DP-DPBE` algorithm incorporated with the above local and shuffle DP models, we provide the DP guarantee and regret in the following.

Theorem D.9 (DP guarantee)**.** *Suppose any user is not sampled in two different phases.*

i) Extend the `DPBE` *algorithm by employing a local randomizer* $\mathcal{R}$ *as Eq.* (D.52) *at each par-*

[1] $Rang(\mathcal{M})$ denotes the range of the output of the mechanism $\mathcal{M}$.

ticipant in the l-th phase, called `LDP-DPBE`*. With* $\sigma_{nl} = \frac{2\sqrt{2(\kappa^2+\sigma^2)H_l \log(2H_l/\delta_1)\ln(1.25/\delta_2))}}{\varepsilon}$, `LDP-DPBE` *guarantees* (ε, δ)*-LDP for where* $\delta = \delta_1 + \delta_2$.

ii) The `DPBE` *algorithm employing the shuffle protocol of Algorithm 11 in each phase l with input* $\{\mathbf{y}_l^u\}_{u \in U_l}$ *and* $\Delta_2 = B\kappa\sqrt{H_l} + \sqrt{2(\kappa^2+\sigma^2)H_l \log(1/\delta_1)}$*, guarantees* $(\varepsilon, \delta_1 + \delta_2)$*-SDP. In addition, the introduced error for privacy is sub-Gaussian with variance* $\sigma_{ns}^2 = O\left(\frac{\Delta_2^2 \ln(H_l/\delta_2)^2}{\varepsilon^2 |U_l|^2}\right)$ *and independent of the inputs.*

We achieve the above LDP guarantee in i) directly by employing the Gaussian mechanism given the sensitivity of $\mathbf{y}_l^u$. In the shuffle model, we follow the shuffle protocol for each phase in [89] and derive the corresponding SDP guarantee from Theorem A.2 therein.

From the above results, we derive that compared to the local model the shuffle model injects much less noise (σ_{ns}^2 vs. σ_{nl}^2) without requiring a trusted agent. In the following, we present the regret performance of `DP-DPBE` in these two DP models.

Theorem D.10 (LDP-DPBE)**.** *Apply the Gaussian mechanism in the local DP model with* $\sigma_{nl} = \frac{2\sqrt{2(\kappa^2+\sigma^2)H_l \log(2H_1/\delta_1)\ln(1.25/\delta_2))}}{\varepsilon}$ *. With* $\beta = \frac{1}{|\mathcal{D}|T}$ *and* $\sigma_n = \sqrt{\frac{2C^2\sigma_{nl}^2\gamma_T}{|U_l|}}$*, the* `DP-DPBE` *extension with a local DP model achieves the following expected regret*

$$\mathbb{E}[R(T)] = O(T^{1-\alpha/2}\sqrt{\log(|\mathcal{D}|T)}) + O\left(\frac{\ln(1/\delta)\gamma_T T^{1-\alpha/2}\sqrt{\log(|\mathcal{D}|T)}}{\varepsilon}\right). \tag{D.56}$$

Theorem D.11 (SDP-DPBE)**.** *With* $\sigma_n = \sigma_{ns}\sqrt{2C^2\gamma_T} = O\left(\frac{\gamma_T \ln(\gamma_T/\delta_2)\sqrt{(\kappa^2+\sigma^2)\log(H_l/\delta_1))}}{\varepsilon|U_l|}\right)$ *in the l-th phase and* $\beta = \frac{1}{|\mathcal{D}|T}$*, the* `DP-DPBE` *extension with the shuffle model in Algorithm 11achieves the following expected regret*

$$\mathbb{E}[R(T)] = O(T^{1-\alpha/2}\sqrt{\log(|\mathcal{D}|T)}) + O\left(\frac{\ln^{3/2}(\gamma_T/\delta)\gamma_T T^{1-\alpha}\sqrt{\log(|\mathcal{D}|T)}}{\varepsilon}\right). \tag{D.57}$$

Table D.2: Regret of DP-DPBE in Different DP Models

Algorithms	Regret
DPBE	$O(T^{1-\alpha/2}\sqrt{\log(\lvert\mathcal{D}\rvert T)})$
CDP-DPBE	$O(T^{1-\alpha/2}\sqrt{\log(\lvert\mathcal{D}\rvert T)}) + O\left(\frac{\ln(1/\delta)\gamma_T T^{1-\alpha}\sqrt{\log(\lvert\mathcal{D}\rvert T)}}{\varepsilon}\right)$
LDP-DPBE	$O(T^{1-\alpha/2}\sqrt{\log(\lvert\mathcal{D}\rvert T)}) + O\left(\frac{\ln(1/\delta)\gamma_T T^{1-\alpha/2}\sqrt{\log(\lvert\mathcal{D}\rvert T)}}{\varepsilon}\right)$
SDP-DPBE	$O(T^{1-\alpha/2}\sqrt{\log(\lvert\mathcal{D}\rvert T)}) + O\left(\frac{\ln^{3/2}(\gamma_T/\delta)\gamma_T T^{1-\alpha}\sqrt{\log(\lvert\mathcal{D}\rvert T)}}{\varepsilon}\right)$

Notes: CDP-DPBE, LDP-DPBE, and SDP-DPBE represent the DP-DPBE algorithm in the central, local, and shuffle models, respectively, which guarantee (ε, δ)-DP, (ε, δ)-LDP, and (ε, δ)-SDP, respectively.

We omit the proofs for the above two theorems because they can be derived by directly replacing σ_n of the central model with $\sigma_n = \sqrt{\frac{2C^2\sigma_{nl}^2\gamma_T}{|U_l|}}$ of the local model and $\sigma_n = \sigma_{ns}\sqrt{2C^2\gamma_T}$ of the shuffle model. See Appendix D.5.3.

D.5.2 Proofs for DP Guarantees

Before providing the DP guarantee of the DPBE algorithm in the three DP models, we first show the ℓ_2 sensitivity of $\bar{\mathbf{y}}_l$, which is a key parameter to decide the Gaussian noise.

Lemma D.12 (Sensitivity)**.** *Let $\mathcal{U}_T, \mathcal{U}_T' \subseteq \mathcal{U}$ be two sets of participants in DPBE differing on a single user that is participating in the l-th phase.*

$$\Delta_2 \triangleq \max |\bar{\mathbf{y}}_l' - \bar{\mathbf{y}}_l| \leqslant 2\frac{\sqrt{(\kappa^2+\sigma^2)H_l \log(2H_l/\delta_1)}}{|U_l|} \tag{D.58}$$

where H_l denotes the dimension of $\bar{\mathbf{y}}_l$ and σ^2 is the variance of the noisy observations.

Proof. Let U_l, U_l' be the sets of participating users in l-th phase corresponding to $\mathcal{U}_T$ and $\mathcal{U}_T'$ respectively. We have $|U_l| = |U_l'|$ and the sensitivity

$$\begin{aligned}\Delta_2 &\triangleq \max |\bar{\mathbf{y}}_l' - \bar{\mathbf{y}}_l| \\ &= \max_{\mathcal{U}_T, \mathcal{U}_T'} \left\| \frac{1}{|U_l|} \sum_{u \in U_l'} \mathbf{y}_l^u - \frac{1}{|U_l|} \sum_{u \in U_l} \mathbf{y}_l^u \right\|_2 \\ &\leqslant \frac{1}{|U_l|} \max_{u, u' \in \mathcal{U}} \|\mathbf{y}_l^{u'} - \mathbf{y}_l^u\|_2.\end{aligned} \tag{D.59}$$

For any chosen action $\mathbf{a} \in \mathbf{A}_{H_l}$, we have the following result

$$\begin{aligned}|y_l^{u'}(\mathbf{a}) - y_l^u(\mathbf{a})| &= \left| \frac{1}{T_l(\mathbf{a})} \sum_{t \in \mathcal{T}_l(\mathbf{a})} y_{u',t} - \frac{1}{T_l(\mathbf{a})} \sum_{t \in \mathcal{T}_l(\mathbf{a})} y_{u,t} \right| \\ &= \left| \frac{1}{T_l(\mathbf{a})} \sum_{t \in \mathcal{T}_l(\mathbf{a})} (y_{u',t} - y_{u,t}) \right| \\ &= \left| \frac{1}{T_l(\mathbf{a})} \sum_{t \in \mathcal{T}_l(\mathbf{a})} (f_{u'}(\mathbf{x}_t) + \eta_{u',t} - f_u(\mathbf{x}_t) - \eta_{u,t}) \right| \\ &\leqslant \frac{1}{T_l(\mathbf{a})} \sum_{t \in \mathcal{T}_l(\mathbf{a})} |f_{u'}(\mathbf{x}_t) + \eta_{u',t} - f_u(\mathbf{x}_t) - \eta_{u,t}| \,.\end{aligned}$$

Note that $f_u(\mathbf{x}) \sim \mathcal{N}(f(\mathbf{x}), k(\mathbf{x}, \mathbf{x}))$, $\eta_{u,t} \sim \mathcal{N}(0, \sigma^2)$, and the participating users are independent from each other. We have $(f_{u'}(\mathbf{x}_t) + \eta_{u',t} - f_u(\mathbf{x}_t) - \eta_{u,t}) \sim \mathcal{N}(0, 2(k(\mathbf{x}_t, \mathbf{x}_t) + \sigma^2))$. According to the concentration property of Gaussian distribution, we have with probability at least $1 - \delta_1$,

$$|f_{u'}(\mathbf{x}_t) + \eta_{u',t} - f_u(\mathbf{x}_t) + \eta_{u,t})| \leqslant 2\sqrt{(k(\mathbf{x}_t, \mathbf{x}_t) + \sigma^2) \log(2/\delta_1)} \leqslant 2\sqrt{(\kappa^2 + \sigma^2) \log(2/\delta_1)}$$

which results in $|y_l^{u'}(\mathbf{a}) - y_l^u(\mathbf{a})| \leqslant 2\sqrt{(\kappa^2 + \sigma^2) \log(2/\delta_1)}$ for any particular $\mathbf{a} \in \mathbf{A}_{H_l}$ with probability at least $1 - \delta_1$. Substituting the above result into Eq. (D.59) and combining a

union bound, we have with probability at least $1-\delta_1$ the ℓ_2 sensitivity of $\bar{\mathbf{y}}_l$ is bounded by

$$\Delta_2 \leqslant \frac{\max_{u,u'\in\mathcal{U}} \|\mathbf{y}_l^{u'} - \mathbf{y}_l^{u}\|_2}{|U_l|} \leqslant \frac{2\sqrt{H_l(\kappa^2+\sigma^2)\log(2H_l/\delta_1)}}{|U_l|} \tag{D.60}$$

where the last step is because H_l is the dimension of $\mathbf{y}_l^u$ and also the number of actions in $\mathbf{A}_{H_l}$. □

For both the central model and the local model, we employ the Gaussian mechanism in the differential privacy literature, which is described in the following.

Theorem D.13. *(Gaussian Mechanism [50]). Given any vector-valued function*[2] $f:\mathcal{U}^* \to \mathbb{R}^s$*, define* $\Delta_2 \triangleq \max_{\mathcal{U}_1,\mathcal{U}_2'\subseteq\mathcal{U}} \|f(\mathcal{U}_1)-f(\mathcal{U}_2)\|_2$*. Let* $\sigma = \Delta_2\sqrt{2\ln(1.25/\delta)}/\varepsilon$*. The Gaussian mechanism, which adds independently drawn random noise from* $\mathcal{N}(0,\sigma^2)$ *to each output of* $f(\cdot)$*, i.e. returning* $f(\mathcal{U})+(\rho_1,\ldots,\rho_s)$ *with* $\rho_j \overset{i.i.d.}{\sim} \mathcal{N}(0,\sigma^2)$*, ensures* (ε,δ)*-DP.*

Proof of Theorem 5.13. Let E denote the event that the global sensitivity bound of Theorem D.12 holds. Thus, $\mathbb{P}[E] \geqslant 1-\delta_1$. If E holds, adding independently drawn noise from $\mathcal{N}(0, \frac{2\Delta_2^2\ln(1.25/\delta_2)}{\varepsilon})$ to each element of $\bar{\mathbf{y}}_l$, i.e., returning $\bar{\mathbf{y}}_l + (\rho_1,\cdots,\rho_{H_l})$ with $\rho_j \overset{i.i.d.}{\sim} \mathcal{N}(0, \frac{2\Delta_2^2\ln(1.25/\delta_2)}{\varepsilon})$, ensures (ε,δ_2)-DP according to Theorem D.13. Specifically, the following inequality holds

$$\mathbb{P}[\mathcal{M}(\mathcal{U}_T)\in Z|E] \leqslant e^{\varepsilon}\mathcal{P}[\mathcal{M}(\mathcal{U}_T')\in Z|E] + \delta_2. \tag{D.61}$$

[2] We use the superscript * to indicate that the length could be varying.

Then, we have

$$\begin{aligned}
\mathbb{P}[\mathcal{M}(\mathcal{U}_T) \in Z] &\leqslant \mathbb{P}[\mathcal{M}(\mathcal{U}_T) \in Z|E]\mathbb{P}[E] + 1 - \mathbb{P}[E] \\
&\leqslant (e^{\varepsilon}\mathcal{P}[\mathcal{M}(\mathcal{U}'_T) \in Z|E] + \delta_2)\mathbb{P}[E] + \delta_1 \\
&\leqslant e^{\varepsilon}\mathcal{P}[\mathcal{M}(\mathcal{U}'_T) \in Z|E]\mathbb{P}[E] + \delta_2 + \delta_1 \\
&\leqslant e^{\varepsilon}\mathcal{P}[\mathcal{M}(\mathcal{U}'_T) \in Z|E]\mathbb{P}[E] + \delta_2 + \delta_1 \\
&\leqslant e^{\varepsilon}\mathcal{P}[\mathcal{M}(\mathcal{U}'_T) \in Z, E] + \delta_2 + \delta_1 \\
&\leqslant e^{\varepsilon}\mathcal{P}[\mathcal{M}(\mathcal{U}'_T) \in Z] + \delta
\end{aligned} \tag{D.62}$$

where $\delta = \delta_1 + \delta_2$. □

Similarly, we can derive the (ε, δ)-LDP. Meanwhile, we can achieve (ε, δ)-SDP by combining the analysis in Eq. (D.62) and the proof for Theorem A.2 in [89].

D.5.3 Proof of Theorem 4.11

Following a similar line to the proof for Theorem 5.3, we first provide the key concentration inequality under `DP-DPBE` in Theorem D.14.

Theorem D.14. *For any particular phase l, with probability at least $1 - 6\beta$, the following holds*

$$|f(\mathbf{x}) - \tilde{\mu}_l(\mathbf{x})| \leqslant \tilde{w}_l(\mathbf{x}), \tag{D.63}$$

where mean function $\tilde{\mu}_l(\mathbf{x})$ and confidence width function $\tilde{w}_l(\mathbf{x})$ are defined in Eq. (5.16) and Eq. (5.17).

Proof. In this proof, we will show the following concentration inequality holds for any $\mathbf{x} \in \mathcal{D}$

$$\mathbb{P}[|f(\mathbf{x}) - \tilde{\mu}_l(\mathbf{x})| \geqslant \tilde{w}_l(\mathbf{x})] \leqslant 6\beta. \tag{D.64}$$

Let $\boldsymbol{\rho} \triangleq (\rho_1, \ldots, \rho_{H_l})$. Note that

$$\begin{aligned}\tilde{\mu}_l(\mathbf{x}) &= \mathbf{k}(\mathbf{x}, \mathbf{A}_{H_l})^\top (\mathbf{K}_{\mathbf{A}_{H_l}\mathbf{A}_{H_l}} + \lambda \mathbf{W}_{H_l}^{-1})^{-1} \tilde{\mathbf{y}}_l \\ &= \mathbf{k}(\mathbf{x}, \mathbf{A}_{H_l})^\top (\mathbf{K}_{\mathbf{A}_{H_l}\mathbf{A}_{H_l}} + \lambda \mathbf{W}_{H_l}^{-1})^{-1} (\bar{y}_l + \boldsymbol{\rho}) \\ &= \bar{\mu}_l(x) + \mathbf{k}(\mathbf{x}, \mathbf{A}_{H_l})^\top (\mathbf{K}_{\mathbf{A}_{H_l}\mathbf{A}_{H_l}} + \lambda \mathbf{W}_{H_l}^{-1})^{-1} \boldsymbol{\rho}.\end{aligned} \tag{D.65}$$

Then, we have

$$|f(\mathbf{x}) - \tilde{\mu}_l(\mathbf{x})| \leqslant |f(\mathbf{x}) - \bar{\mu}_l(\mathbf{x})| + |\mathbf{k}(\mathbf{x}, \mathbf{A}_{H_l})^\top (\mathbf{K}_{\mathbf{A}_{H_l}\mathbf{A}_{H_l}} + \lambda \mathbf{W}_{H_l}^{-1})^{-1} \boldsymbol{\rho}|. \tag{D.66}$$

For any $\mathbf{x} \in \mathcal{D}$, we have

$$\begin{aligned}&\mathbb{P}\left[|f(\mathbf{x}) - \tilde{\mu}_l(\mathbf{x})| \geqslant \tilde{w}_l(\mathbf{x})\right] \\ \leqslant &\mathbb{P}\left[|f(\mathbf{x}) - \bar{\mu}_l(x)| + \left|\mathbf{k}(\mathbf{x}, \mathbf{A}_{H_l})^\top (\mathbf{K}_{\mathbf{A}_{H_l}\mathbf{A}_{H_l}} + \lambda \mathbf{W}_{H_l}^{-1})^{-1} \boldsymbol{\rho}\right| \geqslant w_l(\mathbf{x}) + 2C\sqrt{\gamma_T \sigma_n^2 \log(1/\beta)}\right] \\ \leqslant &\mathbb{P}\left[|f(\mathbf{x}) - \bar{\mu}_l(x)| \geqslant w_l(\mathbf{x})\right] + \mathbb{P}\left[\left|\mathbf{k}(\mathbf{x}, \mathbf{A}_{H_l})^\top (\mathbf{K}_{\mathbf{A}_{H_l}\mathbf{A}_{H_l}} + \lambda \mathbf{W}_{H_l}^{-1})^{-1} \boldsymbol{\rho}\right| \geqslant 2C\sqrt{\gamma_T \sigma_n^2 \log(1/\beta)}\right] \\ \leqslant &4\beta + \mathbb{P}\left[\left|\mathbf{k}(\mathbf{x}, \mathbf{A}_{H_l})^\top (\mathbf{K}_{\mathbf{A}_{H_l}\mathbf{A}_{H_l}} + \lambda \mathbf{W}_{H_l}^{-1})^{-1} \boldsymbol{\rho}\right| \geqslant 2C\sqrt{\gamma_T \sigma_n^2 \log(1/\beta)}\right],\end{aligned} \tag{D.67}$$

where the first inequality is due to $\tilde{w}_l(x) = w_l(x) + \sqrt{2\sigma_n^2 \log(1/\beta)}$ from Eq. (5.17), the second inequality is from union bound, and the last one is from Theorem D.7. Hence, it remains to bound the second probability in Eq. (D.67).

Recall that $\boldsymbol{\rho} = (\rho_1, \ldots, \rho_{H_l})$ where $\rho_j \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma_{nc}^2)$. Then, $\mathbf{k}(\mathbf{x}, \mathbf{A}_{H_l})^\top (\mathbf{K}_{\mathbf{A}_{H_l}\mathbf{A}_{H_l}} + \lambda \mathbf{W}_{H_l}^{-1})^{-1} \boldsymbol{\rho}$ is the sum of H_l *i.i.d.* Gaussian variables, and the total variance (denoted by σ_{sum}^2) is

$$\sigma_{\text{sum}}^2 = \mathbf{k}(\mathbf{x}, \mathbf{A}_{H_l})^\top (\mathbf{K}_{\mathbf{A}_{H_l}\mathbf{A}_{H_l}} + \lambda \mathbf{W}_{H_l}^{-1})^{-1} (\mathbf{K}_{\mathbf{A}_{H_l}\mathbf{A}_{H_l}} + \lambda \mathbf{W}_{H_l}^{-1})^{-1} \mathbf{k}(\mathbf{x}, \mathbf{A}_{H_l}) \sigma_{nc}^2. \tag{D.68}$$

Notice that

$$
\begin{aligned}
&\mathbf{k}(\mathbf{x}, \mathbf{A}_{H_l})^\top (\mathbf{K}_{\mathbf{A}_{H_l}\mathbf{A}_{H_l}} + \lambda \mathbf{W}_{H_l}^{-1})^{-1} (\mathbf{K}_{\mathbf{A}_{H_l}\mathbf{A}_{H_l}} + \lambda \mathbf{W}_{H_l}^{-1})^{-1} \mathbf{k}(\mathbf{x}, \mathbf{A}_{H_l}) \\
=&\varphi(\mathbf{x})^\top \Phi_{H_l}^\top (\Phi_{H_l}\Phi_{H_l}^\top + \lambda \mathbf{W}_{H_l}^{-1})^{-1} (\Phi_{H_l}\Phi_{H_l}^\top + \lambda \mathbf{W}_{H_l}^{-1})^{-1} \Phi_{H_l}\varphi(\mathbf{x}) \\
=&\varphi(\mathbf{x})^\top \Phi_{H_l}^\top \mathbf{W}_{H_l}^{1/2} (\mathbf{W}_{H_l}^{1/2}\Phi_{H_l}\Phi_{H_l}^\top \mathbf{W}_{H_l}^{1/2} + \lambda \mathbf{I})^{-1} \mathbf{W}_{H_l}^{1/2} \cdot \mathbf{W}_{H_l}^{1/2} (\mathbf{W}_{H_l}^{1/2}\Phi_{H_l}\Phi_{H_l}^\top \mathbf{W}_{H_l}^{1/2} + \lambda \mathbf{I})^{-1} \mathbf{W}_{H_l}^{1/2} \Phi_{H_l}\varphi(\mathbf{x}) \\
=&\varphi(\mathbf{x})^\top (\Phi_{H_l}^\top \mathbf{W}_{H_l}^{1/2}\mathbf{W}_{H_l}^{1/2}\Phi_{H_l} + \lambda \mathbf{I})^{-1} \Phi_{H_l}^\top \mathbf{W}_{H_l}^{1/2} \cdot \mathbf{W}_{H_l} \cdot \mathbf{W}_{H_l}^{1/2} \Phi_{H_l} (\Phi_{H_l}^\top \mathbf{W}_{H_l}^{1/2}\mathbf{W}_{H_l}^{1/2}\Phi_{H_l} + \lambda \mathbf{I})^{-1} \varphi(\mathbf{x}) \\
=&\varphi(\mathbf{x})^\top (\Phi_{H_l}^\top \mathbf{W}_{H_l}\Phi_{H_l} + \lambda \mathbf{I})^{-1} \Phi_{H_l}^\top \mathbf{W}_{H_l}^2 \Phi_{H_l} (\Phi_{H_l}^\top \mathbf{W}_{H_l}\Phi_{H_l} + \lambda \mathbf{I})^{-1} \varphi(\mathbf{x}) \\
\overset{(a)}{\leqslant}&T_l \varphi(\mathbf{x})^\top (\Phi_{H_l}^\top \mathbf{W}_{H_l}\Phi_{H_l} + \lambda \mathbf{I})^{-1} \Phi_{H_l}^\top \mathbf{W}_{H_l} \Phi_{H_l} (\Phi_{H_l}^\top \mathbf{W}_{H_l}\Phi_{H_l} + \lambda \mathbf{I})^{-1} \varphi(\mathbf{x}) \\
=&T_l \varphi(\mathbf{x})^\top (\Phi_{H_l}^\top \mathbf{W}_{H_l}\Phi_{H_l} + \lambda \mathbf{I})^{-1} (\Phi_{H_l}^\top \mathbf{W}_{H_l}\Phi_{H_l} + \sigma_n^2 \mathbf{I}) (\Phi_{H_l}^\top \mathbf{W}_{H_l}\Phi_{H_l} + \lambda \mathbf{I})^{-1} \varphi(\mathbf{x}) \\
&- \lambda T_l \varphi(\mathbf{x})^\top (\Phi_{H_l}^\top \mathbf{W}_{H_l}\Phi_{H_l} + \lambda \mathbf{I})^{-1} (\Phi_{H_l}^\top \mathbf{W}_{H_l}\Phi_{H_l} + \lambda \mathbf{I})^{-1} \varphi(\mathbf{x}) \\
\leqslant&T_l \varphi(\mathbf{x})^\top (\Phi_{H_l}^\top \mathbf{W}_{H_l}\Phi_{H_l} + \lambda \mathbf{I})^{-1} \varphi(\mathbf{x}) \\
\overset{(b)}{=}&T_l \varphi(\mathbf{x})^\top (\Phi_{\tau_{H_l}}^\top \Phi_{\tau_{H_l}} + \lambda \mathbf{I})^{-1} \varphi(\mathbf{x}) \\
\overset{(c)}{=}&\frac{T_l \sigma_{\tau_{H_l}}^2(\mathbf{x})}{\lambda} \\
\overset{(d)}{=}&\frac{T_l \Sigma_{H_l}^2(\mathbf{x})}{\lambda} = \frac{T_l \Sigma_{H_l}^2(\mathbf{x})}{\sigma^2} \leqslant 2C^2 \gamma_{T_l}
\end{aligned}
\tag{D.69}
$$

where (a) is from $\Phi_{H_l}^\top \mathbf{W}_{H_l}^2 \Phi_{H_l} \leqslant \Phi_{H_l}^\top (T_l \mathbf{I}) \mathbf{W}_{H_l} \Phi_{H_l} = T_l \Phi_{H_l}^\top \mathbf{W}_{H_l} \Phi_{H_l}$ because each diagonal entry of W_{H_l} satisfies $[W_{H_l}]_{hh} = T_l(\mathbf{a}_h) \leqslant T_l$, (b) is based on Eq. (D.10), (c) is from Eq. (D.5), and (d) is according to the equivalence representation in Lemma D.3. The last step is from the result in Lemma D.5.

Substituting the above result into Eq. (D.68), we have

$$
\sigma_{\text{sum}}^2 \leqslant 2C^2 \gamma_{T_l} \sigma_{nc}^2 = \sigma_n^2. \tag{D.70}
$$

According to the tail bound of Gaussian variables, we have

$$\mathbb{P}\left[\left|\mathbf{k}(\mathbf{x}, \mathbf{A}_{H_l})^\top (\mathbf{K}_{\mathbf{A}_{H_l}\mathbf{A}_{H_l}} + \lambda \mathbf{W}_{H_l}^{-1})^{-1} \boldsymbol{\rho}\right| \geqslant \sqrt{2\sigma_n^2 \log(1/\beta)}\right] \leqslant 2\exp\left\{-\frac{4C^2\gamma_T\sigma_{nc}^2\log(1/\beta)}{2\sigma_{\text{sum}}^2}\right\} \leqslant 2\beta$$

□

Proof of Theorem 4.11. Similar to the proof of Theorem 5.3, we, to prove Theorem 4.11, first present three results when the concentration inequality in Theorem D.14 holds, then obtain an upper bound for the regret incurred in a particular phase $l > 2$ with high probability, and finally sum up the regret over all phases.

1) Three observations when Eq. (D.63) holds

Define a "good" event when Eq. (D.63) holds in the l-th phase as:

$$\tilde{\mathcal{E}}_l \triangleq \{\forall \mathbf{x} \in \mathcal{D}_l, |f(\mathbf{x}) - \tilde{\mu}_l(\mathbf{x})| \leqslant \tilde{w}_l(\mathbf{x})\}.$$

We have $\mathbb{P}[\tilde{\mathcal{E}}_l] \geqslant 1 - 6|\mathcal{D}|\beta$ via the union bound. Then, similar to the non-private case, under event $\tilde{\mathcal{E}}_l$ in the l-th phase, we have the following three observations:

1. For any optimal action $\mathbf{x}^* \in \operatorname{argmax}_{\mathbf{x}\in\mathcal{D}} f(\mathbf{x})$, if $\mathbf{x}^* \in \mathcal{D}_l$, then $\mathbf{x}^* \in \mathcal{D}_{l+1}$.

2. Let $f^* = \max_{\mathbf{x}\in\mathcal{D}} f(\mathbf{x})$. Supposed that $x^* \in \mathcal{D}_l$. For any $\mathbf{x} \in \mathcal{D}_{l+1}$, its reward gap from the optimal reward is bounded by $4\max_{\mathbf{x}\in\mathcal{D}_l} \tilde{w}_l(\mathbf{x})$, i.e.,

$$f^* - f(\mathbf{x}) \leqslant 4\max_{\mathbf{x}\in\mathcal{D}_l} \tilde{w}_l(\mathbf{x}).$$

3. The confidence width function in the private setting satisfies

$$\max_{\mathbf{x}\in\mathcal{D}_l} \tilde{w}_l(\mathbf{x}) \leqslant \max_{\mathbf{x}\in\mathcal{D}_l} w_l(\mathbf{x}) + \frac{G_1\gamma_T\sqrt{2\log(1/\beta)}}{|U_l|}, \tag{D.71}$$

where $G_1 \triangleq \frac{8C^2\sqrt{2(\kappa^2+\sigma^2)\sigma^2\log(1/\delta_1)\ln(1.25/\delta_2)}}{\varepsilon\sqrt{C^2-1}}$.

The first two observations can be derived similar to the non-private case. Regarding the third observation, we have the confidence width function in the private setting $\tilde{w}_l(\mathbf{x}) = w_l(\mathbf{x}) + \sqrt{2\sigma_n^2\log(1/\beta)}$ and

$$\begin{aligned}
&\sqrt{2\sigma_n^2\log(1/\beta)} \\
&= 2C\sqrt{\gamma_T\sigma_{nc}^2\log(1/\beta)} \\
&= \frac{4C\sqrt{2(\kappa^2+\sigma^2)H_l\gamma_T\log(1/\delta_1)\ln(1.25/\delta_2)\log(1/\beta)}}{\varepsilon|U_l|} \\
&\leqslant \frac{8C^2\gamma_T\sqrt{2(\kappa^2+\sigma^2)\sigma^2\log(1/\delta_1)\ln(1.25/\delta_2)\log(1/\beta)}}{\varepsilon|U_l|\sqrt{C^2-1}} && \text{(Lemma 5.8)} \\
&\leqslant \underbrace{\frac{8C^2\sqrt{2(\kappa^2+\sigma^2)\sigma^2\log(1/\delta_1)\ln(1.25/\delta_2)}}{\varepsilon\sqrt{C^2-1}}}_{G_1}\cdot\frac{\gamma_T\sqrt{2\log(1/\beta)}}{|U_l|}.
\end{aligned}$$

2) Regret in a specific phase $l > 2$.

Under event $\tilde{\mathcal{E}}_{l-1}$, the regret incurred in the l-th phase is

$$
\begin{aligned}
&\sum_{t\in\mathcal{T}_l} f^* - f(\mathbf{x}_t) \\
\leqslant &\sum_{t\in\mathcal{T}_l} 4 \max_{\mathbf{x}\in\mathcal{D}_{l-1}} \tilde{w}_{l-1}(\mathbf{x}) \\
\leqslant &4T_l \max_{\mathbf{x}\in\mathcal{D}_{l-1}} w_{l-1}(\mathbf{x}) \\
\leqslant &4T_l \max_{\mathbf{x}\in\mathcal{D}_{l-1}} w_{l-1}(\mathbf{x}) + 4T_l \cdot \frac{G_1\gamma_T\sqrt{2\log(1/\beta)}}{|U_{l-1}|} \qquad \text{(Observation 3)} \\
\leqslant &4T_l \max_{\mathbf{x}\in\mathcal{D}_{l-1}} w_{l-1}(\mathbf{x}) + 4G_1\gamma_T\sqrt{2\log(1/\beta)}2^{(1-\alpha)(l-1)} \\
\leqslant &4\sqrt{2\kappa^2\log(1/\beta)}\sqrt{2^{(2-\alpha)(l-1)}} + 8\sigma C\sqrt{2\gamma_T\log(1/\beta)}\sqrt{2^{(1-\alpha)(l-1)}} + 8\sigma BC\sqrt{\gamma_T 2^{l-1}} \\
&+ 4G_1\gamma_T\sqrt{2\log(1/\beta)}2^{(1-\alpha)(l-1)},
\end{aligned}
$$

where the last step is from Eq. (D.41).

3) Total regret.

Define $\tilde{\mathcal{E}}_g$ as the event where the "good" event occurs in every phase in the private setting, i.e., $\tilde{\mathcal{E}}_g \triangleq \bigcap_{l=1}^{L} \tilde{\mathcal{E}}_l$. It is not difficult to obtain $\mathbb{P}[\mathcal{E}_g] \geqslant 1 - 6|\mathcal{D}|\beta L$ by applying union bound.

At the same time, the total regret under event $\tilde{\mathcal{E}}_g$ becomes

$$
\begin{aligned}
R_g &= \sum_{l=1}^{L} \sum_{t \in \mathcal{T}_l} (f^* - f(\mathbf{x}_t)) \\
&\leqslant 2B\kappa + \sum_{l=2}^{L} 4\sqrt{2\kappa^2 \log(1/\beta)} \sqrt{2^{(2-\alpha)(l-1)}} \\
&+ \sum_{l=2}^{L} 8\sigma C \sqrt{2\gamma_T \log(1/\beta)} \sqrt{2^{(1-\alpha)(l-1)}} \\
&+ \sum_{l=2}^{L} 8\sigma BC \sqrt{\gamma_T 2^{l-1}} \\
&+ \sum_{l=2}^{L} 4G_1 \gamma_T \sqrt{2\log(1/\beta)} 2^{(1-\alpha)(l-1)} \\
&\leqslant 2B\kappa + 4\sqrt{2\kappa^2 \log(1/\beta)} \cdot 4\sqrt{2^{(L-1)(2-\alpha)}} \\
&+ 8\sigma C \sqrt{2\gamma_T \log(1/\beta)} \cdot C_1 \sqrt{2^{(1-\alpha)(L-1)}} \qquad \left(C_1 \triangleq \frac{\sqrt{2^{1-\alpha}}}{\sqrt{2^{1-\alpha}} - 1} \right) \\
&+ 8\sigma BC \sqrt{\gamma_T} \cdot 4\sqrt{2^{L-1}} \\
&+ 4G_1 \gamma_T \sqrt{2\log(1/\beta)} \cdot C_2 2^{(1-\alpha)(L-1)} \qquad \left(C_2 \triangleq \frac{2^{1-\alpha}}{2^{1-\alpha} - 1} \right) \\
&\leqslant 2B\kappa + 16\sqrt{2\kappa^2 \log(1/\beta)} T^{1-\alpha/2} + 8\sigma C_1 C \sqrt{2\gamma_T \log(1/\beta) T^{1-\alpha}} \\
&+ 32\sigma BC \sqrt{\gamma_T T} + 4C_2 G_1 \gamma_T \sqrt{2\log(1/\beta)} T^{1-\alpha},
\end{aligned}
\tag{D.72}
$$

where the last step is is due to $2^{L-1} \leqslant T$ and $L \leqslant \log(2T)$ since $\sum_{l=1}^{L-1} T_l + 1 \leqslant T$. On the other hand, $R_b \leqslant 2B\kappa T$ since $|\max_{\mathbf{x} \in \mathcal{D}} f(\mathbf{x}) - f(\mathbf{x})| \leqslant 2B\kappa$ for all $\mathbf{x} \in \mathcal{D}$. Choose

$\beta = 1/(|\mathcal{D}|T)$ in Algorithm 10. Then, the expected regret is:

$$
\begin{aligned}
&\mathbb{E}[R(T)] = \mathbb{P}[\tilde{\mathcal{E}}_g]R_g + (1 - \mathbb{P}[\tilde{\mathcal{E}}_g])R_b \\
&\leqslant R_g + 6|\mathcal{D}|\beta L \cdot 2B\kappa T \\
&\leqslant 2B\kappa + 16\sqrt{2\kappa^2 \log(1/\beta)}T^{1-\alpha/2} + 8\sigma C_1 C\sqrt{2\gamma_T \log(1/\beta)T^{1-\alpha}} + 32\sigma BC\sqrt{\gamma_T T} \\
&+ 4C_2 G_1 \gamma_T \sqrt{2\log(1/\beta)}T^{1-\alpha} + 12B\kappa|\mathcal{D}|\beta LT \\
&= 2B\kappa + 16T^{1-\alpha/2}\sqrt{2\kappa^2 \log(|\mathcal{D}|T)} + 8\sigma C_1 C\sqrt{2\gamma_T T^{1-\alpha}\log(|\mathcal{D}|T)} + 32\sigma BC\sqrt{\gamma_T T} \\
&+ 4C_2 G_1 \gamma_T \sqrt{2\log(|\mathcal{D}|T)}T^{1-\alpha} + 12B\kappa \log(2T) \\
&= O(T^{1-\alpha/2}\sqrt{\log(|\mathcal{D}|T)}) + O(\sqrt{\gamma_T T^{1-\alpha}\log(|\mathcal{D})T)} + O(G_1\gamma_T T^{1-\alpha}\sqrt{\log(|\mathcal{D}|T)}) + O(\sqrt{\gamma_T T}).
\end{aligned}
\tag{D.73}
$$

Finally, substituting G_1 with $\delta_1 = \delta_2 = \delta/2$, we have the total expected regret under the `DP-DPBE` with the central model is

$$
\begin{aligned}
\mathbb{E}[R(T)] = O(T^{1-\alpha/2}\sqrt{\log(|\mathcal{D}|T)}) + O\left(\frac{\ln(1/\delta)\gamma_T T^{1-\alpha}\sqrt{\log(kT)}}{\varepsilon}\right) \\
+ O(\sqrt{\gamma_T T^{1-\alpha}\log(|\mathcal{D})T)} + O(\sqrt{\gamma_T T}).
\end{aligned}
\tag{D.74}
$$

□

While the `DPBE` algorithm uses GP tools to define and manage the uncertainty in estimating the unknown function f, the analysis of `DPBE` algorithm does not rely on any *Bayesian* assumption about f being actually drawn from the prior $\mathcal{GP}(0, k)$, and it only requires f to be bounded in the kernel norm associated with the RKHS $\mathcal{H}_k$.

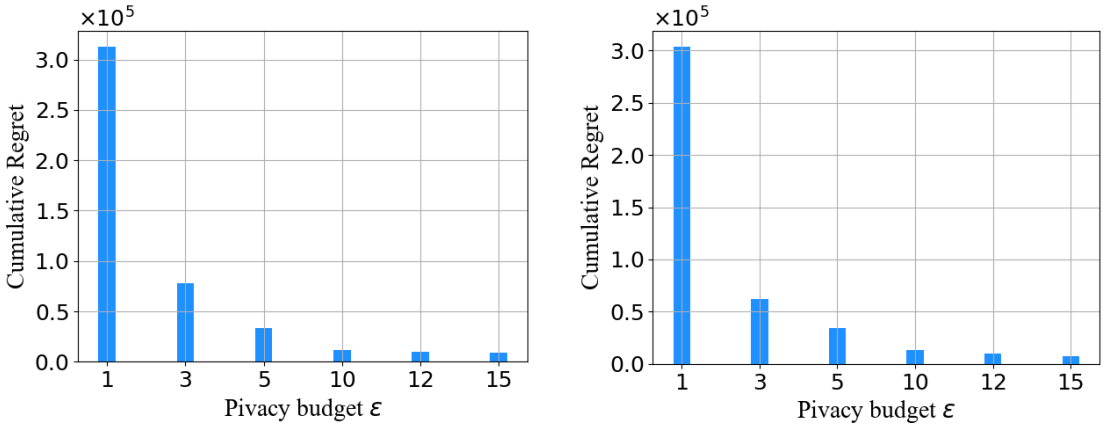
D.6 Additional Numerical Results

D.6.1 Evaluations of DP-DPBE

In Sections 5.7, we evaluated DP-DPBE on the synthetic function. In this subsection, we present additional numerical results of DP-DPBE on the standard benchmark functions and the function from real-world data. For these simulations, we run $T = 10^6$ rounds as the simulations for the synthetic function, present how the cumulative regret at the end of T varies with privacy budget ε in Figure D.1 and compare DP-DPBE with DPBE in Figure D.2. We perform 20 runs for each function. From the figures, we can derive the same results regarding privacy-regret tradeoff and the message of achieving privacy "for free" as for the synthetic function.

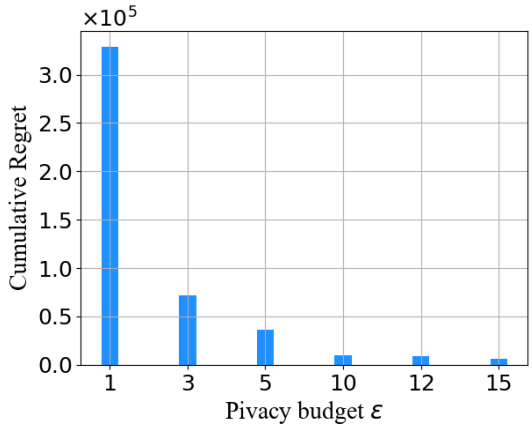
D.6.2 Comparison with State-of-the-Art

In Section 5.8, we provide simulation results on the regret performance and running time of GP-UCB, BPE, and our algorithm DPBE with different values of α on the synthetic data generated in Section 5.7.1 In this section, we add additional numerical results on three benchmark functions (Sphere, Six-hump Camel, Michalewicz) and one function from real-world data– Light sensor data [118]. The parameters of the problem setting and the algorithms are as follows: $T = 4 \times 10^4$, $|\mathcal{D}| = 100$, and $k = k_{SE}$ with $l_{SE} = 0.2$; (a) Sphere function. Settings: $d = 3, C = 1.5, \sigma = 0.01, v^2 = 0.001, \lambda = \sigma^2/v^2$; (b) Six-Hump Camel function. Settings: $d = 2, C = 1.5, \sigma = 0.01, v^2 = 0.01, \lambda = \sigma^2/v^2$; (c) Michalewicz function. Settings: $d = 2, C = 1.5, \sigma = 0.01, v^2 = 0.01, \lambda = \sigma^2/v^2$; (d) Functions from real-world data. Settings: $d = 2, C = 1.42, \sigma = 0.01, v^2 = 0.01, \lambda = \sigma^2/v^2$. We plot the cumulative regret for all the algorithms in Figure D.3 and present the running time in Table D.3.

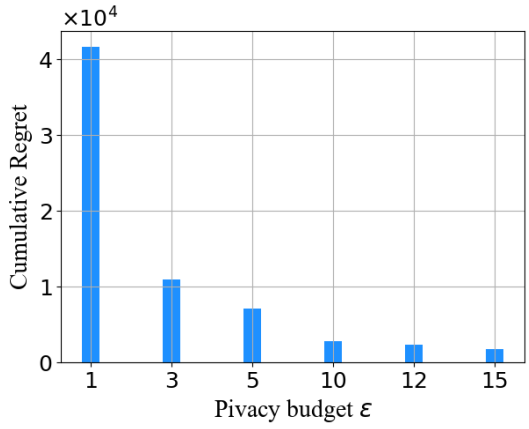


(a) Sphere function

(b) Six-Hump Camel function

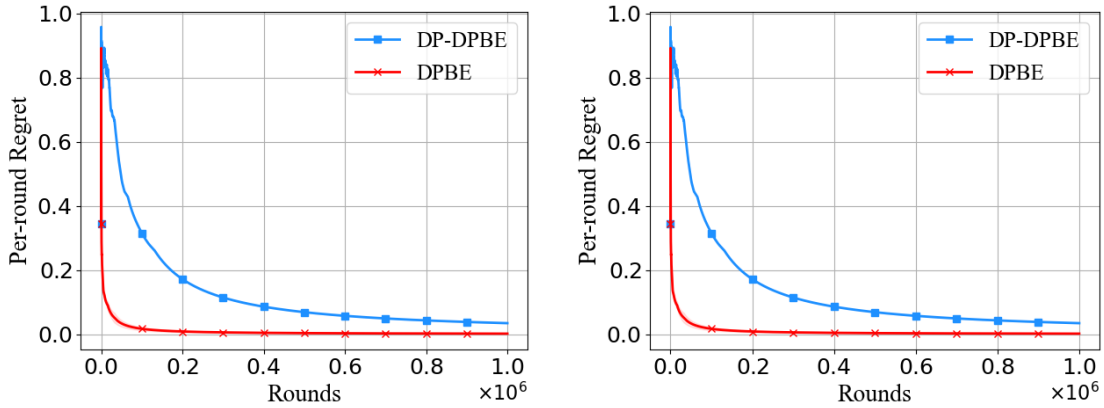


(c) Michalewicz function



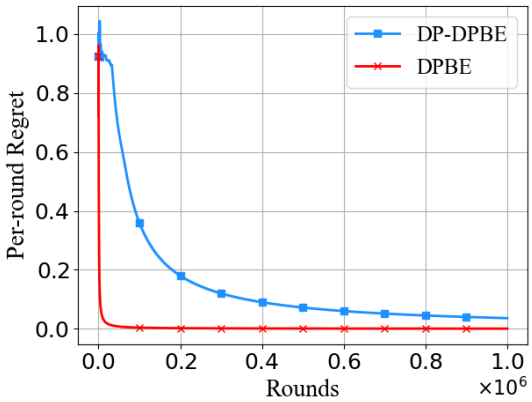
(d) Function from light sensor data

Figure D.1: Performance of `DP-DPBE`: Final cumulative regret vs. privacy budget ε.

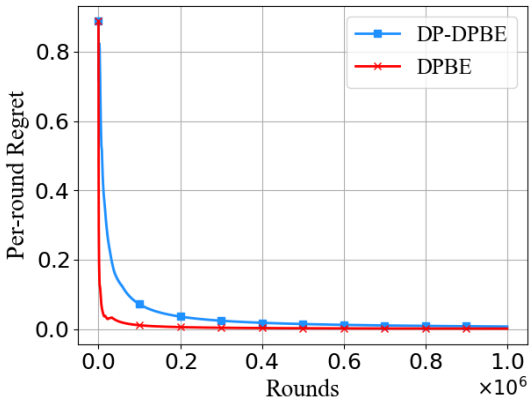


(a) Sphere function

(b) Six-Hump Camel function

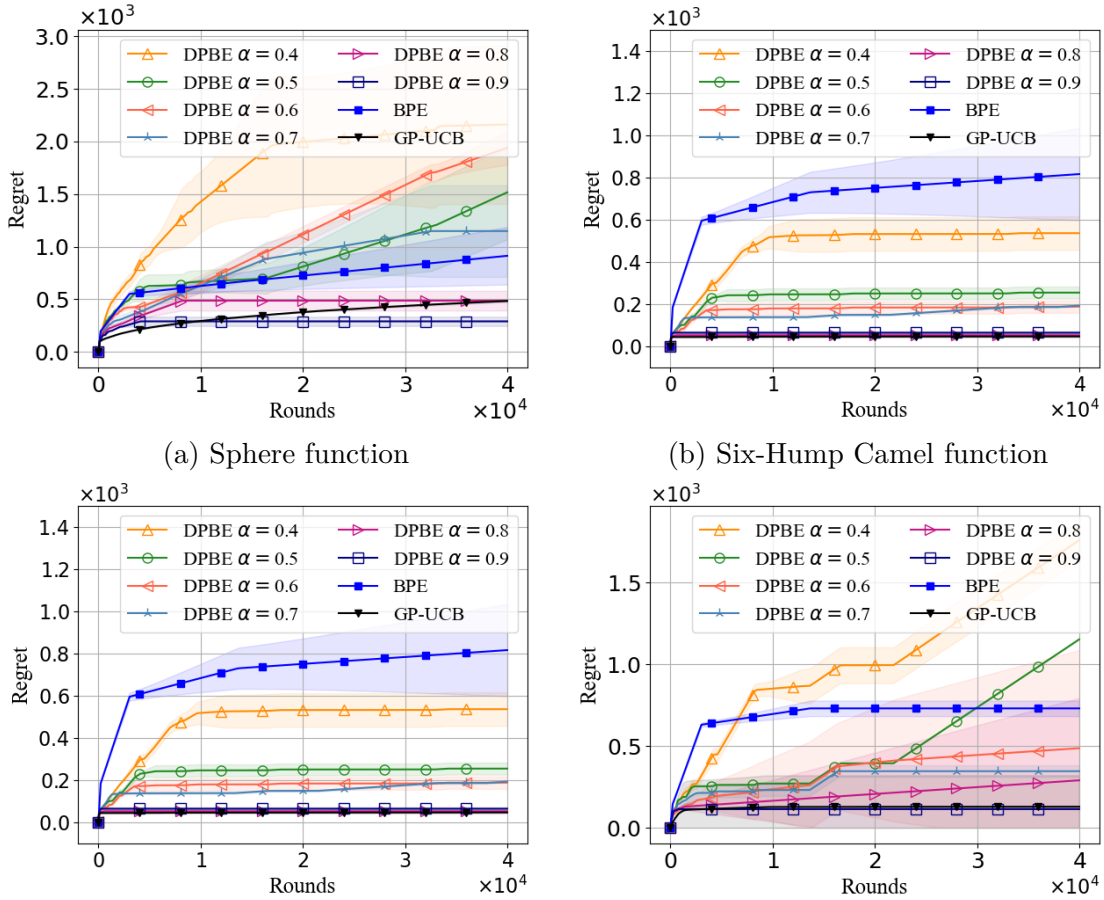


(c) Michalewicz function



(d) Function from light sensor data

Figure D.2: Performance of `DP-DPBE`: Final cumulative regret vs. privacy budget ε.



(a) Sphere function

(b) Six-Hump Camel function

(c) Michalewicz function

(d) Function from light sensor data

Figure D.3: Comparison of regret performance under `DPBE`, `GP-UCB`, and `BPE` on three benchmark functions and one function from real-world dataset. The shaded area represents the standard deviation.

Table D.3: Comparison of running time (seconds) under GP-UCB, BPE, and DPBE with different values of α.

Algorithms	DPBE $\alpha = 0.4$	DPBE $\alpha = 0.5$	DPBE $\alpha = 0.6$	DPBE $\alpha = 0.7$	DPBE $\alpha = 0.8$	DPBE $\alpha = 0.9$	GP-UCB	BPE
Sphere	0.08	0.07	0.07	0.07	0.09	0.13	4.68	37.87
Six-Hump Camel	0.04	0.03	0.04	0.03	0.04	0.04	4.79	10.43
Michalewicz	0.04	0.04	0.05	0.06	0.07	0.11	4.95	4.48
Light Sensor Data	0.04	0.06	0.07	0.03	0.06	0.05	3.22	82.08

Algorithm 10 Differentially Private DPBE (DP-DPBE)

Input: $\mathcal{D} \subseteq \mathbb{R}^d$, $\alpha \in (0,1)$, $\beta \in (0,1)$, C, σ^2, σ_n, and privacy parameters ε, δ
Initialization: $l = 1$, $\mathcal{D}_1 = \mathcal{D}$, $t_1 = 0$, and $T_1 = 1$
while $t_l < T$ **do**
  Set $\tau = 1$, $h = 0$, $\tau_1 = 0$ and $\Sigma_0^2(\mathbf{x}) = k(\mathbf{x}, \mathbf{x})$, for all $\mathbf{x} \in \mathcal{D}_l$
  while $\tau \leqslant T_l$ **do**
    $h = h + 1$
    Choose

$$\mathbf{a}_h \in \operatorname*{argmax}_{\mathbf{x} \in \mathcal{D}_l} \Sigma_{h-1}^2(\mathbf{x}) \tag{D.48}$$

    Play action $\mathbf{a}_h$ for $T_l(\mathbf{a}_h) \triangleq \lfloor (C^2 - 1)/\Sigma_{h-1}^2(\mathbf{a}_h) \rfloor$ times if not reaching $\min\{T, t_l + T_l\}$
    Update $\tau = \tau + T_l(\mathbf{a}_h)$ and $\Sigma_h^2(\cdot)$ by including $\mathbf{a}_h$ according to Eq. (5.7).
  end while
  Let $H_l = h$ denote the total number of batches in this phase.
  Randomly select $\lceil 2^{\alpha l} \rceil$ participants U_l
  for each participant $u \in U_l$ **do**
    Collect and compute local average reward for every chosen action $\mathbf{a} \in \mathbf{A}_{H_l}$: $y_l^u(\mathbf{a}) = \frac{1}{T_l(\mathbf{a})} \sum_{t \in \mathcal{T}_l(\mathbf{a})} y_{u,t}$
    Send the local average reward for every chosen action $\mathbf{y}_l^u \triangleq [y_l^u(\mathbf{a})]_{\mathbf{a} \in \mathbf{A}_{H_l}}$ to the agent
  end for
  Aggregate local observations for each chosen action $\mathbf{a} \in \mathbf{A}_{H_l}$: $y_l(\mathbf{a}) = \frac{1}{|U_l|} \sum_{u \in U_l} y_l^u(\mathbf{a})$
  Let $\bar{\mathbf{y}}_l = [y_l(\mathbf{a}_1), \ldots, y_l(\mathbf{a}_{H_l})]$ and

$$\tilde{\mathbf{y}}_l = \bar{\mathbf{y}}_l + (\rho_1, \ldots, \rho_{H_l}), \text{ where } \rho_j \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma_n^2)$$

  Update $\tilde{\mu}_l(\cdot)$:

$$\tilde{\mu}_l(\mathbf{x}) \triangleq \mathbf{k}(\mathbf{x}, \mathbf{A}_{H_l})^\top (\mathbf{K}_{\mathbf{A}_{H_l}\mathbf{A}_{H_l}} + \lambda \mathbf{W}_{H_l}^{-1})^{-1} \tilde{\mathbf{y}}_l \tag{D.49}$$

  Eliminate low-rewarding actions from $\mathcal{D}_l$ based on $\tilde{w}_l(\cdot)$ in Eq. (5.17):

$$\mathcal{D}_{l+1} = \left\{ \mathbf{x} \in \mathcal{D}_l : \tilde{\mu}_l(\mathbf{x}) + \tilde{w}_l(\mathbf{x}) \geqslant \max_{\mathbf{b} \in \mathcal{D}_l} (\tilde{\mu}_l(\mathbf{b}) - \tilde{w}_l(\mathbf{b})) \right\}. \tag{D.50}$$

  $T_{l+1} = 2T_l$, $t = t + T_l$; $l = l + 1$
end while

Algorithm 11 $\mathcal{M}$: Shuffle Protocol for a Set of Vectors with Users U [89]

Input: $\{\mathbf{y}^u\}_{u\in U}$, where each $\mathbf{y}^u \in \mathbb{R}^s$, $\|\mathbf{y}^u\|_2 \leqslant \Delta_2$
Parameters: ε, δ_2, g, b, p
Let

$$\begin{cases} \hat{\varepsilon} = \frac{\varepsilon}{18\sqrt{\log(2/\delta_2)}} \\ g = \max\{\hat{\varepsilon}\sqrt{|U|}/(6\sqrt{5\ln((4s)/\delta_2)}), \sqrt{s}, 10\} \\ b = \lceil \frac{180g^2 \ln(4s/\delta_2)}{\hat{\varepsilon}^2|U|} \rceil \\ p = \frac{90g^2 \ln(4s/\delta_2)}{b\hat{\varepsilon}^2|U|} \end{cases} \tag{D.55}$$

// Local Randomizer
function $\mathcal{R}(\mathbf{y}^u)$
  for coordinate $j \in [s]$ **do**
    Shift data to enforce non-negativity: $w_{u,j} = (\mathbf{y}^u)_j + \Delta_2, \forall u \in U$
    //randomizer for each entry
    Set $\bar{w}_{u,j} \leftarrow \lfloor w_{u,j} g/(2\Delta_2) \rfloor$ $\quad$ //$\max|(\mathbf{y}^u)_j + \Delta_2| \leqslant 2\Delta_2$
    Sample rounding value $\gamma_1 \sim \mathbf{Ber}(w_{u,j}g/(2\Delta_2) - \bar{w}_{u,j})$
    Sample privacy noise value $\gamma_2 \sim \mathbf{Bin}(b, p)$
    Let ϕ_j^u be a multi-set of $(g+b)$ bits associated with the j-th coordinate of user u, where ϕ_j^u consists of $\bar{w}_{u,j} + \gamma_1 + \gamma_2$ copies of 1 and $g + b - (\bar{w}_{i,j} + \gamma_1 + \gamma_2)$ copies of 0
  end for
  Report $\{(j, \phi_j^u)\}_{j\in[s]}$ to the shuffler
end function
// Shuffler
function $\mathcal{S}(\{(j, \boldsymbol{\phi}_j)\}_{j\in[s]})$ $\quad$ //$\boldsymbol{\phi}_j = (\phi_j^u)_{u\in U}$
  for each coordinate $j \in [s]$ **do**
    Shuffle and output all $(g+b)|U|$ bits in $\boldsymbol{\phi}_j$
  end for
end function
// Analyzer
function $\mathcal{A}(\mathcal{S}(\{(j, \boldsymbol{\phi}_j)\}_{j\in[s]})$
  for coordinate $j \in [s]$ **do**
    Compute $z_j \leftarrow \frac{2\Delta_2}{g|U|}((\sum_{i=1}^{(g+b)|U|}(\boldsymbol{\phi}_j)_i) - b|U|p)$ $\quad$ // $(\boldsymbol{\phi}_j)_i$ denotes the i-th bit in $\boldsymbol{\phi}_j$
    Re-center: $o_j \leftarrow z_j - \Delta_2$
  end for
  Output the estimator of vector average $o = (o_j)_{j\in[s]}$
end function

Bibliography

[1] Hundreds of millions of facebook user records were exposed on amazon cloud server. URL https://www.cbsnews.com/news/millions-facebook-user-records-exposed-amazon-cloud-server/. Accessed: 2021-05-06.

[2] Federated learning: Collaborative machine learning without centralized training data. URL https://ai.googleblog.com/2017/04/federated-learning-collaborative.html. Accessed: 2021-05-06.

[3] Adspeed ad server. https://www.adspeed.com/, 2018. [Accessed: 2018-06-30].

[4] Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. In *NIPS*, volume 11, pages 2312–2320, 2011.

[5] Mridul Agarwal, Vaneet Aggarwal, and Kamyar Azizzadenesheli. Multi-agent multi-armed bandits with limited communication. *arXiv preprint arXiv:2102.08462*, 2021.

[6] Alexander A Ageev and Maxim I Sviridenko. Pipage rounding: A new method of constructing algorithms with proven performance guarantee. *Journal of Combinatorial Optimization*, 8(3):307–328, 2004.

[7] Tafseer Akhtar, Christos Tselios, and Ilias Politis. Radio resource management: approaches and implementations from 4g to 5g and beyond. *Wireless Networks*, 27(1): 693–734, 2021.

[8] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multi-armed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

[9] Ashwinkumar Badanidiyuru and Jan Vondrák. Fast algorithms for maximizing submodular functions. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 1497–1514. SIAM, 2014.

[10] Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. Streaming submodular maximization: Massive data summarization on the fly. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 671–680, 2014.

[11] Ravikumar Balakrishnan, Tian Li, Tianyi Zhou, Nageen Himayat, Virginia Smith, and Jeff Bilmes. Diverse client selection for federated learning: Submodularity and convergence analysis. In *International Workshop on Federated Learning for User Privacy and Data Confidentiality in Conjunction with ICML 2020*. International Machine Learning Society (IMLS), 2021.

[12] Fuat Basık, Bugra Gedik, Hakan Ferhatosmanoglu, and Kun-Lung Wu. Fair task allocation in crowdsourced delivery. *IEEE Transactions on Services Computing*, 2018.

[13] Raef Bassily, Vitaly Feldman, Kunal Talwar, and Abhradeep Thakurta. Private stochastic convex optimization with optimal rates. *arXiv preprint arXiv:1908.09970*, 2019.

[14] Andrea Bittau, Úlfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnes, and Bernhard Seefeld. Prochlo: Strong privacy for analytics in the crowd. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 441–459, 2017.

[15] Ilija Bogunovic and Andreas Krause. Misspecified gaussian process bandit optimization. *Advances in Neural Information Processing Systems*, 34, 2021.

[16] Ilija Bogunovic, Zihan Li, Andreas Krause, and Jonathan Scarlett. A robust phased elimination algorithm for corruption-tolerant gaussian process bandits. *arXiv preprint arXiv:2202.01850*, 2022.

[17] Djallel Bouneffouf, Irina Rish, and Charu Aggarwal. Survey on applications of multi-armed and contextual bandits. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2020.

[18] Sébastien Bubeck and Nicolo Cesa-Bianchi. Regret analysis of stochastic and non-stochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1):1–122, 2012.

[19] Niv Buchbinder and Moran Feldman. Submodular functions maximization problems. *Handbook of Approximation Algorithms and Metaheuristics*, 1:753–788, 2017.

[20] Peter Bühlmann and Sara Van De Geer. *Statistics for high-dimensional data: methods, theory and applications*. Springer Science & Business Media, 2011.

[21] K. Cai, X. Liu, Y. J. Chen, and John C. S. Lui. An online learning approach to network application optimization with guarantee. In *Proceedings of IEEE INFOCOM*, 2018. in press.

[22] Daniele Calandriello, Luigi Carratino, Alessandro Lazaric, Michal Valko, and Lorenzo Rosasco. Near-linear time gaussian process optimization with adaptive batching and resparsification. In *International Conference on Machine Learning*, pages 1295–1305. PMLR, 2020.

[23] Daniele Calandriello, Luigi Carratino, Alessandro Lazaric, Michal Valko, and Lorenzo Rosasco. Scaling gaussian process optimization by evaluating a few unique candidates multiple times. *arXiv preprint arXiv:2201.12909*, 2022.

[24] Gruia Calinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. Maximizing a submodular set function subject to a matroid constraint. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 182–196. Springer, 2007.

[25] Romain Camilleri, Kevin Jamieson, and Julian Katz-Samuels. High-dimensional experimental design and kernel bandits. In *International Conference on Machine Learning*, pages 1227–1237. PMLR, 2021.

[26] Clément L Canonne, Gautam Kamath, and Thomas Steinke. The discrete gaussian for differential privacy. *arXiv preprint arXiv:2004.00010*, 2020.

[27] Nicol'o Cesa-Bianchi, Claudio Gentile, Yishay Mansour, and Alberto Minora. Delay and cooperation in nonstochastic bandits. In *Conference on Learning Theory*, pages 605–622. PMLR, 2016.

[28] Chandra Chekuri and Jan Vondrák. Randomized pipage rounding for matroid polytopes and applications. *CoRR, abs/0909.4348*, 4, 2009.

[29] Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. Dependent randomized rounding via exchange properties of combinatorial structures. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 575–584. IEEE, 2010.

[30] Kun Chen, Kechao Cai, Longbo Huang, and John Lui. Beyond the click-through rate: Web link selection with multi-level feedback. *arXiv preprint arXiv:1805.01702*, 2018.

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

[32] Mingzhe Chen, Nir Shlezinger, H Vincent Poor, Yonina C Eldar, and Shuguang Cui. Communication-efficient federated learning. *Proceedings of the National Academy of Sciences*, 118(17), 2021.

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

[34] Albert Cheu, Adam Smith, Jonathan Ullman, David Zeber, and Maxim Zhilyaev. Distributed differential privacy via shuffling. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 375–403. Springer, 2019.

[35] Albert Cheu, Matthew Joseph, Jieming Mao, and Binghui Peng. Shuffle private stochastic convex optimization. *arXiv preprint arXiv:2106.09805*, 2021.

[36] Alexandra Chouldechova and Aaron Roth. The frontiers of fairness in machine learning. *arXiv preprint arXiv:1810.08810*, 2018.

[37] Sayak Ray Chowdhury and Aditya Gopalan. On kernelized multi-armed bandits. In *International Conference on Machine Learning*, pages 844–853. PMLR, 2017.

[38] Sayak Ray Chowdhury and Xingyu Zhou. Shuffle private linear contextual bandits. *arXiv preprint arXiv:2202.05567*, 2022.

[39] Wei Chu, Lihong Li, Lev Reyzin, and Robert Schapire. Contextual bandits with linear payoff functions. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 208–214. JMLR Workshop and Conference Proceedings, 2011.

[40] Zhongxiang Dai, Kian Hsiang Low, and Patrick Jaillet. Federated Bayesian optimization via Thompson sampling. *arXiv preprint arXiv:2010.10154*, 2020.

[41] Zhongxiang Dai, Bryan Kian Hsiang Low, and Patrick Jaillet. Differentially private federated bayesian optimization with distributed exploration. *Advances in Neural Information Processing Systems*, 34:9125–9139, 2021.

[42] Varsha Dani, Thomas P Hayes, and Sham M Kakade. Stochastic linear optimization under bandit feedback. In *In Proceedings of the 21st Annual Conference on Learning Theory (COLT)*, pages 355–366, 2008.

[43] Eric V Denardo, Eugene A Feinberg, and Uriel G Rothblum. The multi-armed bandit, with constraints. *Annals of Operations Research*, 208(1):37–62, 2013.

[44] Yihan Du, Wei Chen, Yuko Yuroki, and Longbo Huang. Collaborative pure exploration in kernel bandit. *arXiv preprint arXiv:2110.15771*, 2021.

[45] Abhimanyu Dubey. No-regret algorithms for private gaussian process bandit optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 2062–2070. PMLR, 2021.

[46] Abhimanyu Dubey and Alex Pentland. Differentially-private federated linear bandits. *arXiv preprint arXiv:2010.11425*, 2020.

[47] Abhimanyu Dubey et al. Cooperative multi-agent bandits with heavy tails. In *International Conference on Machine Learning*, pages 2730–2739. PMLR, 2020.

[48] Abhimanyu Dubey et al. Kernel methods for cooperative multi-agent contextual bandits. In *International Conference on Machine Learning*, pages 2740–2750. PMLR, 2020.

[49] John C Duchi, Michael I Jordan, and Martin J Wainwright. Minimax optimal procedures for locally private estimation. *Journal of the American Statistical Association*, 113(521):182–201, 2018.

[50] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014.

[51] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[52] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*, pages 214–226, 2012.

[53] Cynthia Dwork, Kunal Talwar, Abhradeep Thakurta, and Li Zhang. Analyze gauss: optimal bounds for privacy-preserving principal component analysis. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 11–20, 2014.

[54] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. Amplification by shuffling: From local to central differential privacy via anonymity. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2468–2479. SIAM, 2019.

[55] Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.

[56] Rosa L Figueroa, Qing Zeng-Treitler, Sasikiran Kandula, and Long H Ngo. Predicting sample size required for classification performance. *BMC medical informatics and decision making*, 12(1):1–10, 2012.

[57] Yi Gai, Bhaskar Krishnamachari, and Rahul Jain. Combinatorial network optimization with unknown variables: Multi-armed bandits with linear rewards and individual observations. *IEEE/ACM Transactions on Networking (TON)*, 20(5):1466–1478, 2012.

[58] Rajiv Gandhi, Samir Khuller, Srinivasan Parthasarathy, and Aravind Srinivasan. Dependent rounding and its applications to approximation algorithms. *Journal of the ACM (JACM)*, 53(3):324–360, 2006.

[59] Lin Gao, Fen Hou, and Jianwei Huang. Providing long-term participation incentive in participatory sensing. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 2803–2811. IEEE, 2015.

[60] Evrard Garcelon, Kamalika Chaudhuri, Vianney Perchet, and Matteo Pirotta. Privacy amplification via shuffling for linear contextual bandits. *arXiv preprint arXiv:2112.06008*, 2021.

[61] Robin C Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*, 2017.

[62] Antonious Girgis, Deepesh Data, Suhas Diggavi, Peter Kairouz, and Ananda Theertha Suresh. Shuffled model of differential privacy in federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 2521–2529. PMLR, 2021.

[63] Vijay Gupta, Timothy H Chung, Babak Hassibi, and Richard M Murray. On a stochastic sensor selection algorithm with applications in sensor scheduling and sensor coverage. *Automatica*, 42(2):251–260, 2006.

[64] Zhu Han and KJ Ray Liu. *Resource allocation for wireless networks: basics, techniques, and applications*. Cambridge university press, 2008.

[65] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. In *The Collected Works of Wassily Hoeffding*, pages 409–426. Springer, 1994.

[66] I. H. Hou, V. Borkar, and P. R. Kumar. A theory of qos for wireless. In *Proceedings of IEEE INFOCOM*, pages 486–494, April 2009. doi: 10.1109/INFCOM.2009.5061954.

[67] Wei-Kang Hsu, Jiaming Xu, Xiaojun Lin, and Mark R Bell. Integrate learning and control in queueing systems with uncertain payoffs. Technical report, Purdue University, available at https://engineering.purdue.edu/%7elinx/papers.html, 2018.

[68] Ruiquan Huang, Weiqiang Wu, Jing Yang, and Cong Shen. Federated linear contextual bandits. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[69] Prateek Jain, Pravesh Kothari, and Abhradeep Thakurta. Differentially private online learning. In *Conference on Learning Theory*, pages 24–1. JMLR Workshop and Conference Proceedings, 2012.

[70] David Janz, David Burt, and Javier González. Bandit optimisation of functions in the matérn kernel RKHS. In *International Conference on Artificial Intelligence and Statistics*, pages 2486–2495. PMLR, 2020.

[71] Syed Talha Jawaid and Stephen L Smith. Submodularity and greedy algorithms in sensor scheduling for linear dynamical systems. *Automatica*, 61:282–288, 2015.

[72] Matthew Joseph, Michael Kearns, Jamie Morgenstern, Seth Neel, and Aaron Roth. Fair algorithms for infinite and contextual bandits. *arXiv preprint arXiv:1610.09559*, 2016.

[73] Matthew Joseph, Michael Kearns, Jamie H Morgenstern, and Aaron Roth. Fairness in learning: Classic and contextual bandits. In *Advances in Neural Information Processing Systems*, pages 325–333, 2016.

[74] Peter Kairouz, Ziyu Liu, and Thomas Steinke. The distributed discrete gaussian mechanism for federated learning with secure aggregation. *arXiv preprint arXiv:2102.06387*, 2021.

[75] Motonobu Kanagawa, Philipp Hennig, Dino Sejdinovic, and Bharath K Sriperumbudur. Gaussian processes and kernel methods: A review on connections and equivalences. *arXiv preprint arXiv:1807.02582*, 2018.

[76] Shiva Prasad Kasiviswanathan, Homin K Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? *SIAM Journal on Computing*, 40(3): 793–826, 2011.

[77] Ehsan Kazemi, Morteza Zadimoghaddam, and Amin Karbasi. Scalable deletion-robust submodular maximization: Data summarization with privacy and fairness constraints. In *International conference on machine learning*, pages 2544–2553, 2018.

[78] Jack Kiefer and Jacob Wolfowitz. The equivalence of two extremum problems. *Canadian Journal of Mathematics*, 12:363–366, 1960.

[79] Katrin Kirchhoff and Jeff Bilmes. Submodularity for data selection in machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 131–141, 2014.

[80] Andreas Krause and Daniel Golovin. Submodular function maximization. In *Tractability: Practical Approaches to Hard Problems*. Cambridge University Press, February 2014.

[81] Andreas Krause and Daniel Golovin. Submodular function maximization., 2014.

[82] Tze Leung Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

[83] T. Lan, D. Kao, M. Chiang, and A. Sabharwal. An axiomatic theory of fairness in network resource allocation. In *2010 Proceedings IEEE INFOCOM*, pages 1–9, March 2010.

[84] Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

[85] Tor Lattimore, Csaba Szepesvari, and Gellert Weisz. Learning with good feature representations in bandits and in RL with a generative model. In *International Conference on Machine Learning*, pages 5662–5670. PMLR, 2020.

[86] Siyu Lei, Silviu Maniu, Luyi Mo, Reynold Cheng, and Pierre Senellart. Online influence maximization. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 645–654. ACM, 2015.

[87] F. Li, J. Liu, and B. Ji. Combinatorial sleeping bandits with fairness constraints. In *Proceedings of IEEE INFOCOM*, pages 1–9, 2019.

[88] Fengjiao Li, Jia Liu, and Bo Ji. Combinatorial sleeping bandits with fairness constraints. *IEEE Transactions on Network Science and Engineering*, 2019.

[89] Fengjiao Li, Xingyu Zhou, and Bo Ji. Differentially private linear bandits with partial distributed feedback. *arXiv preprint arXiv:2207.05827*, 2022.

[90] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670, 2010.

[91] Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, Yuan Li, Xu Liu, and Bingsheng He. A survey on federated learning systems: vision, hype and reality for data privacy and protection. *arXiv preprint arXiv:1907.09693*, 2019.

[92] Tian Li, Maziar Sanjabi, and Virginia Smith. Fair resource allocation in federated learning. *arXiv preprint arXiv:1905.10497*, 2019.

[93] Zihan Li and Jonathan Scarlett. Gaussian process bandit optimization with few batches. In *International Conference on Artificial Intelligence and Statistics*, pages 92–107. PMLR, 2022.

[94] Wei Yang Bryan Lim, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit Niyato, and Chunyan Miao. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 2020.

[95] Xin Liu, Edwin KP Chong, and Ness B Shroff. A framework for opportunistic scheduling in wireless networks. *Computer networks*, 41(4):451–474, 2003.

[96] Lorenzo Maggi, Alvaro Valcarce, and Jakob Hoydis. Bayesian optimization for radio resource management: Open loop power control. *IEEE Journal on Selected Areas in Communications*, 39(7):1858–1871, 2021.

[97] Ajay Mahimkar, Ashiwan Sivakumar, Zihui Ge, Shomik Pathak, and Karunasish Biswas. Auric: using data-driven recommendation to automatically generate cellular configuration. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, pages 807–820, 2021.

[98] David Martínez-Rubio, Varun Kanade, and Patrick Rebeschini. Decentralized cooperative stochastic bandits. *Advances in Neural Information Processing Systems*, 32, 2019.

[99] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera

y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.

[100] Christopher Meek, Bo Thiesson, and David Heckerman. The learning-curve sampling method applied to model-based clustering. *Journal of Machine Learning Research*, 2 (Feb):397–418, 2002.

[101] Nikita Mishra and Abhradeep Thakurta. (nearly) optimal differentially private stochastic multi-arm bandits. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, pages 592–601, 2015.

[102] Kanishka Misra, Eric M Schwartz, and Jacob Abernethy. Dynamic online pricing with incomplete information using multiarmed bandit experiments. *Marketing Science*, 38 (2):226–252, 2019.

[103] Mehryar Mohri, Gary Sivek, and Ananda Theertha Suresh. Agnostic federated learning. *arXiv preprint arXiv:1902.00146*, 2019.

[104] Michael J Neely. Stochastic network optimization with application to communication and queueing systems. *Synthesis Lectures on Communication Networks*, 3(1):1–211, 2010.

[105] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14(1):265–294, 1978.

[106] Takayuki Nishio and Ryo Yonetani. Client selection for federated learning with heterogeneous resources in mobile edge. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE, 2019.

[107] Daniel Pérez Palomar and Mung Chiang. A tutorial on decomposition methods for network utility maximization. *IEEE Journal on Selected Areas in Communications*, 24(8):1439–1451, 2006.

[108] Vishakha Patil, Ganesh Ghalme, Vineet Nair, and Y Narahari. Stochastic multi-armed bandits with arm-specific fairness guarantees. *arXiv preprint arXiv:1905.11260*, 2019.

[109] Petar Popovski, Kasper Fløe Trillingsgaard, Osvaldo Simeone, and Giuseppe Durisi. 5g wireless network slicing for embb, urllc, and mmtc: A communication-theoretic view. *Ieee Access*, 6:55765–55779, 2018.

[110] Friedrich Pukelsheim. *Optimal design of experiments*. SIAM, 2006.

[111] Theodore Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition, 2001. ISBN 0130422320.

[112] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer, 2003.

[113] Carl Edward Rasmussen and Christopher KI Williams. *Gaussian processes for machine learning*, volume 1. MIT press Cambridge, MA, 2006.

[114] Sayak Ray Chowdhury and Aditya Gopalan. Bayesian optimization under heavy-tailed payoffs. *Advances in Neural Information Processing Systems*, 32, 2019.

[115] Wenbo Ren, Xingyu Zhou, Jia Liu, and Ness B Shroff. Multi-armed bandits with local differential privacy. *arXiv preprint arXiv:2007.03121*, 2020.

[116] Paat Rusmevichientong and John N Tsitsiklis. Linearly parameterized bandits. *Mathematics of Operations Research*, 35(2):395–411, 2010.

[117] Touqir Sajed and Or Sheffet. An optimal private stochastic-mab algorithm based on optimal private stopping rule. In *International Conference on Machine Learning*, pages 5579–5588. PMLR, 2019.

[118] School of Computer Science, Carnegie Mellon University. Light sensor data. Retrieved October 05, 2022, from http://www.cs.cmu.edu/~guestrin/Class/10708-F08/projects/lightsensor.zip.

[119] Manohar Shamaiah, Siddhartha Banerjee, and Haris Vikalo. Greedy sensor selection: Leveraging submodularity. In *49th IEEE conference on decision and control (CDC)*, pages 2572–2577. IEEE, 2010.

[120] Roshan Shariff and Or Sheffet. Differentially private contextual linear bandits. *arXiv preprint arXiv:1810.00068*, 2018.

[121] Chengshuai Shi and Cong Shen. Federated multi-armed bandits. In *35th AAAI Conference on Artificial Intelligence*, 2021.

[122] Chengshuai Shi, Cong Shen, and Jing Yang. Federated multi-armed bandits with personalization. In *International Conference on Artificial Intelligence and Statistics*, pages 2917–2925. PMLR, 2021.

[123] Rachael Hwee Ling Sim, Yehong Zhang, Bryan Kian Hsiang Low, and Patrick Jaillet. Collaborative bayesian optimization with fair regret. In *International Conference on Machine Learning*, pages 9691–9701. PMLR, 2021.

[124] Aleksandrs Slivkins. Introduction to multi-armed bandits. *arXiv preprint arXiv:1904.07272*, 2019.

[125] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian

process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

[126] Lichao Sun, Weiran Huang, Philip S Yu, and Wei Chen. Multi-round influence maximization. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2249–2258. ACM, 2018.

[127] S. Surjanovic and D. Bingham. Virtual library of simulation experiments: Test functions and datasets. Retrieved July 29, 2022, from http://www.sfu.ca/~ssurjano.

[128] Mohammad Sadegh Talebi and Alexandre Proutiere. Learning proportionally fair allocations with low regret. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 2(2):36:1–36:31, 2018.

[129] Jay Tenenbaum, Haim Kaplan, Yishay Mansour, and Uri Stemmer. Differentially private multi-armed bandits in the shuffle model. *arXiv preprint arXiv:2106.02900*, 2021.

[130] Hien To, Cyrus Shahabi, and Leyla Kazemi. A server-assigned spatial crowdsourcing framework. *ACM Transactions on Spatial Algorithms and Systems*, 1(1):2, 2015.

[131] Aristide CY Tossou and Christos Dimitrakakis. Algorithms for differentially private multi-armed bandits. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[132] Vasileios Tzoumas. *Resilient Submodular Maximization for Control and Sensing*. PhD thesis, University of Pennsylvania, 2018.

[133] Sattar Vakili, Nacime Bouziani, Sepehr Jalali, Alberto Bernacchia, and Da-shan Shiu. Optimal order simple regret for gaussian process bandits. *Advances in Neural Information Processing Systems*, 34, 2021.

[134] Sattar Vakili, Kia Khezeli, and Victor Picheny. On information gain and regret bounds in gaussian process bandits. In *International Conference on Artificial Intelligence and Statistics*, pages 82–90. PMLR, 2021.

[135] Harish Viswanathan and Preben E Mogensen. Communications in the 6g era. *IEEE Access*, 8:57063–57074, 2020.

[136] Jan Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 67–74. ACM, 2008.

[137] Yuanhao Wang, Jiachen Hu, Xiaoyu Chen, and Liwei Wang. Distributed bandit learning: Near-optimal regret with efficient communication. *arXiv preprint arXiv:1904.06309*, 2019.

[138] Kai Wei, Rishabh Iyer, and Jeff Bilmes. Submodularity in data subset selection and active learning. In *International Conference on Machine Learning*, pages 1954–1963, 2015.

[139] Howard H. Yang, Zuozhu Liu, Tony Q. S. Quek, and H. Vincent Poor. Scheduling policies for federated learning in wireless networks. *IEEE Transactions on Communications*, 68(1):317–333, 2020. doi: 10.1109/TCOMM.2019.2944169.

[140] Peng Yang, Ning Zhang, Shan Zhang, Kan Yang, Li Yu, and Xuemin Shen. Identifying the most valuable workers in fog-assisted spatial crowdsourcing. *IEEE Internet of Things Journal*, 4(5):1193–1203, 2017.

[141] Chi Zhang, Kalyan P Subbu, Jun Luo, and Jianxin Wu. Groping: Geomagnetism and crowdsensing powered indoor navigation. *IEEE Transactions on Mobile Computing*, 14(2):387–400, 2014.

[142] Kai Zheng, Tianle Cai, Weiran Huang, Zhenguo Li, and Liwei Wang. Locally differentially private (contextual) bandits learning. *arXiv preprint arXiv:2006.00701*, 2020.

[143] Xingyu Zhou and Bo Ji. On kernelized multi-armed bandits with constraints. *arXiv preprint arXiv:2203.15589*, 2022.

[144] Xingyu Zhou and Jian Tan. Local differential privacy for bayesian optimization. *arXiv preprint arXiv:2010.06709*, 2020.

[145] Yinglun Zhu, Dongruo Zhou, Ruoxi Jiang, Quanquan Gu, Rebecca Willett, and Robert Nowak. Pure exploration in kernel and neural bandits. *Advances in Neural Information Processing Systems*, 34:11618–11630, 2021.

[146] Zhaowei Zhu, Jingxuan Zhu, Ji Liu, and Yang Liu. Federated bandit: A gossiping approach. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 5(1):1–29, 2021.

Vita

Fengjiao Li was born in Changzhi, Shanxi, China, in 1991. Currently, she is a Ph.D. candidate in the Department of Computer Science at Virginia Tech, Blacksburg Virginia, USA. She received her M.S. degree in Communications and Information System at Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 2015. Prior to that, she received her B.S. degree in Electronical and Information Engineering from Taiyuan University of Technology, Taiyuan, China, in 2012.

While as a Ph.D. student, Fengjiao was a Graduate Teaching Assistant (GTA) for the undergraduate courses: CS 1114 Introduction to Software Design, CS 1044 Introduction to C++ Programming, and CS 3114 Data Structures and Algorithms at Virginia Tech, and CIS 3223 Data Structures and Algorithms at Temple University. Her current research interests include machine learning, networking, modeling, optimization, and analysis. During her doctoral study, she is working on online resource allocation under uncertainty, while studying several key issues, such as, fairness, communication efficiency, and data privacy protection in distributed networks. Her work has appeared in top-tier IEEE/ACM conferences and journals.

During her Ph.D. study, she won the IEEE INFOCOM 2019 Best Paper Award, the IEEE INFOCOM 2019 Best In-Session Presentation Award, and the WiOpt 2022 Best Student Paper Award. She is a recipient of Student Travel Grants of IEEE INFOCOM 2019. She also received N2Women Young Researcher Fellowship in 2020. During her Ph.D. study, she received Dr. Dennis G. Kakura Graduate Fellowship in CS at Virginia Tech in 2022 and Scott Hibbs Future of Computing Award in CIS at Temple University in 2020.

Journal Publications

1. **Fengjiao Li**, Jia Liu, and Bo Ji, "Combinatorial Sleeping Bandits with Fairness Constraints," *IEEE Transactions on Network Science and Engineering (TNSE)*, vol. 7, no. 3, pp. 1799-1813, 1 July-Sept. 2020.

2. **Fengjiao Li**, Yu Sang, Zhongdong Liu, Bin Li, Huasen Wu, and Bo Ji, "Waiting but not Aging: Optimizing Information Freshness Under the Pull Model," *IEEE/ACM Transactions on Networking (ToN)*, vol. 29, no. 1, pp. 465-478, February 2021.

3. **Fengjiao Li**, Xingyu Zhou, and Bo Ji, "Differentially Private Linear Bandits with Partial Distributed Feedback," in preparation for submission to *IEEE Transactions on Network Science and Engineering (TNSE)*. (Invited fast track submission)

Conference Publications

1. **Fengjiao Li**, Jia Liu, and Bo Ji, "Combinatorial Sleeping Bandits with Fairness Constraints," Proceedings of *IEEE INFOCOM 2019*, Paris, France, May 2019. (Best Paper Award; Best-in-Session Presentation Award)

2. **Fengjiao Li**, Jia Liu, and Bo Ji, "Federated Learning with Fair Worker Selection: A Multi-Round Submodular Maximization Approach," Proceedings of *IEEE MASS 2021*, Virtual Event, October 2021.

3. **Fengjiao Li**, Xingyu Zhou, and Bo Ji, "Differentially Private Linear Bandits with Partial Distributed Feedback," Proceedings of *IEEE WiOpt 2022*, Turin, Italy, September 2022. (Best Student Paper Award)

4. **Fengjiao Li**, Xingyu Zhou, and Bo Ji, "(Private) Kernelized Bandits with Distributed Biased Feedback," submitted to *ACM SIGMETRICS 2023* (One-shot revision).