Private Allocation of Public Goods

by

Pouya Kananian

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2022

Abstract

We study the problem of designing a truthful mechanism for fair allocation of divisible public goods. We consider a setting with n agents and m items. Each item is associated with a size, and the total size of the allocated items must not exceed the available capacity. All agents can access an allocated item, but agents might have different valuations for different items. To aggregate agents' preferences in a fair and efficient way, we focus on the notion of core, which incorporates Pareto efficiency and sharing inventive. In public good settings, agents might have the incentive to misreport their preferences and free ride on the items allocated by others. To address this issue, we present an approximately truthful mechanism. Our mechanism solves a convex optimization problem in a differentially private manner to find a fair allocation.

Acknowledgements

I am very grateful to my advisor, Dr. Seyed Majid Zahedi, for his guidance, support, and patience. I would like to thank Prof. Mahesh Tripunitara and Dr. Xi He for agreeing to review this thesis and for offering valuable feedback.

This work would not have been possible without my family's unconditional and loving support. They encouraged and supported me whenever I needed them. Many thanks to my friends who made my journey at the University of Waterloo an amazing one. They were there for me in my ups and downs.

Table of Contents

List of Figures

Chapter 1

Introduction

The problem of fair and efficient public good allocation comes up in various settings, such as the participatory budgeting problem (PB) [10, 1] and cache allocation in computer systems [61, 79, 30, 71, 47, 78]. When allocating private goods, each allocated good is assigned to a single user. However, for public goods, the allocated items benefit all users at the same time. Users might have different preferences for the goods, though.

We consider n agents and m public goods, each with a certain size. Given a capacity $c \in \mathbb{R}$, we are looking for a fair and efficient allocation such that the sum of the allocated items' sizes doesn't exceed c. We assume that the goods are divisible, meaning it is possible to allocate fractions of items. For $j \in \{1, \ldots, m\}$, if $z_j \in [0, 1]$ and $s_j \in \mathbb{R}$ denote the allocated fraction of item j and this item's size, respectively, we assume that this fraction consumes $z_j s_j$ of the available capacity. We furthermore, assume that each agent $i \in \{1, \ldots, n\}$, gains a utility equal to $u_{ij} z_j$ for the allocated item j where $u_{ij} \in \mathbb{R}$.

Pareto efficiency (PE) and Sharing incentive[1] (SI) are common desiderata for designing fair and efficient public good allocation mechanisms in computer systems[61, 79, 30, 47]. Envy-freeness is another classical notion of fairness considered in economic theory[47, 72]. The notion of envy, however, is ill-defined for public goods since the allocated items are shared between all users. For public goods, we can assume users will become envious if a large amount of the available resources are spent to make a small group of users happy. This assumption can be formalized using the notion of core[33, 66, 6, 14]. A core outcome is both Pareto efficient and proportional.

Designing a public good allocation mechanism is challenging since agents might be incentivized to lie about their preferences when resources are shared between users. To

[1]Sharing incentive is also referred to as proportionality or isolation guarantee.

illustrate this issue, we can consider the following example. Let us assume that a university's department of Electrical and Computer Engineering wants to decide whether to add a water fountain, a coffee vending machine, or a few couches to the department's lobby. Since they do not have enough budget to buy all these options, they plan to ask students about their preferences. Each student can pick one of the options as their preferred choice, and the department will divide the budget between options in proportion to the number of votes each option gets. Tom, who is a coffee lover, hopes that the department adds the coffee machine and a few couches to the lobby so that he can sit on the couches and enjoy a cup of coffee with his friends from time to time. The coffee machine is his favorite option, but he has observed that this is also the most favored option of many other students. Therefore, he has concluded that the coffee machine will have enough votes among students to collect a sufficient amount of budget. To increase the chances of a few couches being added to the lobby, he decides to vote for the couch option as his favored option instead of honestly reporting the coffee machine as his favorite choice. Suppose other students also make similar calculations in their minds to increase the chances of their favorite outcomes and do not report their preferences truthfully. In that case, the election's outcome will not truly reflect a fair aggregation of students' favorite options. If Tom succeeds in turning the result of the survey to his preferred outcome, we say that he is "free-riding" on the votes of those students who selected the vending machine as their preferred option. To avoid cheating and free-riding in public good allocation policies, designing strategy-proof mechanisms[2] for public good allocation has been a subject of study for decades[37, 55]. Dominant strategic truthful mechanisms are those that the participating agents don't have any incentive to lie about their preferences.

Our public good allocation is similar to the cache allocation problem considered in [61]. Pu et al. show that no cache allocation mechanism exists that satisfies Pareto efficiency, Sharing incentive, and truthfulness simultaneously. They, hence, try to design a truthful and proportional mechanism that is approximately Pareto efficient. Yu et al.[79] show that the mechanism proposed in [61] is not truthful. However, their proposed truthful method for cache allocation can be shown to not be truthful either[30].

Kunjir et al. [47] work on designing a cache allocation mechanism that its output lies in the core in expectation. As mentioned earlier, the notion of core is more general than Pareto efficiency and Sharing incentive and incorporates these two. However, Kunjir et al. do not tackle the problem of free-riding in public good allocation and assume that the tenants are not strategic. Fain et al. [25] work on designing an approximately truthful mechanism that achieves a solution that lies in the approximate core. The authors in [53] show that an ϵ-differentially private mechanism for $\epsilon \leq 1$ is also ϵ-approximately dominant

[2]Also referred to as truthful mechanisms.

strategy truthful. Fain et al. [25] use exponential mechanism[53] to achieve an differentially private and approximate asymptotic truthful mechanism.

In this work, we propose an approximately truthful mechanism for computing an approximate core solution for the problem of allocating divisible public goods based on differentially private convex optimization. Compared to using exponential mechanism, our method is more suitable when the space of the problem is exponentially large. Fain et al. [25] have to resort to the hit-and-run method[51] to sample from their exponential space in the polynomial time. There is no need to use such sampling methods in our proposed approach.

The authors in [41] and [42] work on jointly deferentially private convex optimization for private good allocation. Those methods cannot be applied to our problem since they use the dual decomposition method [24, 8] that cannot be applied to optimization problems with affine constraints. In this work, we propose a private version of the Alternating alternating direction method of multipliers (ADMM)[31, 28, 8] for achieving a fair, efficient, and approximately truthful public good allocation mechanism.

In §3, we formally define our public good allocation problem and explain how we can calculate a core outcome for this problem by solving a convex optimization problem. We propose a differentially private algorithm for solving this optimization problem in §4. §5 is dedicated to proving the privacy properties of the algorithm. In §6, we analyze the algorithm's convergence. Since our algorithm computes an approximately optimal solution to the optimization problem, it is necessary to analyze the violation of constraints in this approximate solution. We go through such feasibility analysis in §7. Finally, in §8, we empirically analyze the performance of our algorithm on synthetic data.

Chapter 2

Related Works

2.1 Truthful cache allocation

While caching can have a significant positive impact on applications' performances, cache memories are often limited by their size. In multi-tenant cloud environments, where many users share the cache memory, an efficient and fair caching policy can substantially affect the latency users experience when accessing their files. As mentioned by Pu et al.[61], traditional frequency-based and time-interval-based cache allocation methods such as LRU and LFU that focus on global hit ratio optimization often do not work well in multi-tenant cloud environments. When using such allocation policies, those users who access their files in long intervals or with low frequencies would experience their files getting evicted from the cache too often. Furthermore, these policies can be played with by the users who make excessive accesses to improve their access rate[61].

A naive approach to preserve fairness in these environments would be to divide the cache memory into equal partitions and give each user access to one of these isolated units. Users don't have any motivation for making excessive accesses or lying about their preferences in this setting. However, this might lead to unused space left in the cache memory. Moreover, in a multi-tenant environment, it is common for multiple users to share a dataset and send queries to the same files. Isolating the cache space of users from each other, therefore, is pretty inefficient in terms of cache utilization[61]. Pu et al. [61] observed that in an HDFS cluster, more than 30 percent of the files are shared between at least two agents in many workloads. A cached file can be accessed by multiple users when the caching policy doesn't limit users to access an isolated section of the cache. This non-exclusive sharing is, in fact, a key distinction between cache and resources such as CPU or communications links.

To come up with a fair and efficient cache allocation policy for a public good resource such as cache, Pu et al. [61] focused on the following desirable properties, which are commonly considered in recourse allocation policies: Isolation guarantee, Pareto-efficiency, and strategy proofness[11, 15, 32, 35, 52, 68, 69, 70, 74, 75, 80]. However, they showed that a cache allocation policy could not satisfy these three properties simultaneously. They discussed that, intuitively, there is a strong trade-off between truthfulness and Pareto-efficiency due to free-riding. They introduced a truthful allocation policy based on probabilistic blocking with isolation guarantee and approximate Pareto efficiency. This policy, named FairRide, probabilistically bans users from accessing files that are not cached by them.

Yu et al.[79] showed that FairRide[61] fails to eliminate strategic users' incentive to lie about their preferences. They proposed a policy for fair cache allocation in shared cloud environments. This policy, which is based on the VCG mechanism[13, 36, 73] provides isolation guarantee, near-optimal Pareto efficiency, and strategic proofness according to their claim. However, as mentioned by Friedman et al.[30], this mechanism is based on the wrong assumption that cheating is acceptable if it does not affect other users in a bad way. This assumption could be refuted by basic game-theoretic reasoning.

Friedman et al.[30] considered a setting in which every user has a few "private" files, and there might be a "public" file in the system that multiple users want to access. They worked on maximizing efficiency while preserving truthfulness and sharing inventiveness. In some resource allocation problems, users pay money based on the amount of resources they use. Since this is not the case in cache allocation, Friedman et al. resorted to money-burning and blocking mechanisms to address the inability to use monetary payments. They came up with an 83%-efficient mechanism for two users and a 63%-efficient one for an arbitrary users number. This work is limited by the assumption that only one "public" file exists in the system.

Tang et al. [71] worked on an allocation mechanism for Semi-External Memories (SEM), which are memory models that combine DRAM and SSD to overcome the limited size of DRAM[71]. To balance fairness and efficiency in shared cloud environments, they proposed partitioning the memory unit into two parts. The first part is split between users using the global max-min fairness policy. The second part is then allocated to optimize efficiency globally without caring about fairness. A tunable knob $\sigma \in [0, 1]$ in this approach determines what fraction of the memory unit should be dedicated for the first stage of the method (fairness-stage allocation) and how much will be left for the second stage (efficiency-stage allocation).

To overcome the problem of cheating in cache allocation, instead of trying to come up with a truthful cache allocation mechanism, Tang et al.[71] proposed a cheating detecting mechanism based on monitoring users' caching history. The downside of using such cheating detecting mechanisms is that an honest user who needs to increase her access frequency could be mistaken for a cheating user.

2.2 Cache allocation in the absence of strategic concerns

Kunjir et al.[47] argued that in a multi-tenant cluster where significant performance boosts can be experienced by the users who gain access to cache, low-priority tenants should not prevent high-priority ones from getting a fair amount of performance. They mentioned that the common practice in the industry at the time was to create groups of similar users. Each group could form a queue or pool and act as a tenant in the cluster.

Pareto-efficiency and envy-freeness are classical notions of fairness in economic theory. [47, 72] These notions, along with sharing incentive, have been considered in many resource allocation problems in computer science. Kunjir et al. [47] argued that since the optimizer can see the queries in their setting, they don't need to deal with strategy proofness. However, as we previously explained, cheat detection by monitoring the access requests of users is not trivial.

While envy freeness is ill-defined in public goods[47], a notion similar to envy can be considered for public goods based on proportionality and stability. We can say that users will become envious if a large amount of the available resources are spent to make a small group of users happy. This can be formalized using the notion of core. Allocations in core are both Pareto-efficient and proportional. Kunjir et al. showed that the solution that maximizes nash social welfare (NSW) lies in the core. They used dual multiplicative weight update [5, 29] to solve this maximization problem.

As mentioned by Fain et al.[26], unlike the utilitarian social welfare objective, which maximizes the sum of agents utilities without caring about fairness, and the egalitarian social welfare objective, which maximizes the minimum agent utility while being blind to efficiency, the nash social welfare encorporates both fairness and efficinecy. Fain et al. showed that the fractional relaxation of the nash social welfare program lies in the core. The fractional MNW outcome for public good allocation can be irrational despite rational inputs, though. [26, 4] This might result in finding approximate solutions rather than exact results.

Yu et al. [78] worked on an efficient caching policy for multi-tenant environments that enforces performance isolation between users. Their cache sharing scheme, LCAS, limits the cache accesses of users with excessive load to mitigate the negative impact of their high load on other users. This approach first tries to find an allocation with optimal efficiency without caring for isolation guarantee. In the second phase of the algorithm, in an attempt to obtain performance guarantee, LCAS searches this allocation for users with too much load and and throttles the cache usage of these users. There won't be any need to go through the second phase if the original allocation in the first phase already provides isolation guarantee.

2.3 Participatory budgeting and core

Participatory budgeting[10, 1] problems involve deciding how to spend tax dollars on public projects through public vote. Public goods make participatory budgeting different from other resource allocation problems. The notion of core[27, 55, 66] in public goods, which incorporates a group sharing inventiveness, is similar to envy freeness for private resources

If a solution is in core, no community of users should be envious about their share of the budget. Furthermore, the definition of core incorporates a weak notion of Pareto efficiency. Min-Max fairness concerns maximizing the utility of the agent with the least utility and results in Pareto-efficient solutions but tries to increase one user's utility in expense of others. In a core solution, in contrast, goods are funded in proportion to the number of agents that can benefit from them[25].

One way for computing a core solution is through the Lindahl equilibrium [50, 65]. The Lindahl equilibrium is proved to exist and be in the core under certain conditions in a mixed market of public and private goods[27]. Fain et al. [25] proposed a way for eliminating the price variables in the Lindahl equilibrium and based on that, they came up with an an efficient algorithm, using convex programming, for computing the core for non-saturating utility functions. They also pointed out that when utility functions are concave and homogeneous of degree 1, the solution that maximizes the proportional fair allocation, a more general version of the nash bargaining solution[57], computes the Lindahl equilibrium. There is a similar result for finding a Fisher equilibrium in markets consisting of private goods[46].

In the real world, utility functions are often saturating, which makes the algorithm for non-saturating utility functions not suitable for real applications. Fain et al., therefore, presented a heuristic algorithm for computing the approximate core for saturating utilities. The experimentally showed that this algorithm often computes the exact core on real data.

As mentioned by Fain et al.[25], the core always exists for participatory budgeting, however, free-riding should be considered as an important problem when allocating public goods and computing a core solution. Fain et al. studied whether in a public good allocation problem, the core outcome is truthful when the total number of agents goes to infinity. They answered this question in a negative way, and worked on an asymptotic truthful mechanism for finding an approximate core solution. Exponential mechanism [53] is used in this work to achieve approximate dominant-strategy truthfulness

2.4 Jointly differentially private convex programming

Hsu et al. [41] proposed a method for solving a family of convex optimization problems with two types of constraints under joint differential privacy. In the first type of constraints, only the variables of a single agent are involved. The second kind of constraints couple multiple agents and include variables from different agents, and therefore are called the coupling constraints. The approach in this paper is based on the dual decomposition[24, 8] method in distributed optimization and involves constructing a partial Lagrangian that couples the second type of constraints with the objective. The partial Lagrangian can be interpreted as the payoff of a zero-sum game between two players. The primal player tries to maximize the Lagrangian. The dual player, who controls the Lagrange variables and plays against the primal player, tries to minimize the partial Lagrangian when the constraints are violated by the primal player. A repeated play of primal and dual responses could lead to an approximate equilibrium in this game. Due to the form of the Lagrangian in this paper, each agent can compute their best response individually. The collection of agents' best responses forms a primal best response in each stage of the game. An agent's response when maximizing the partial Lagrangian is dependent on other agents' data only through the coupling constraints and dual player's actions. Therefore, by adding noise to the Lagrange variables and making the dual actions differentially private, the agents' responses would depend on their own and differentially private data. Hence, the agents' actions would satisfy joint differential privacy. This would guarantee joint differential privacy in the final primal solution of the method, which consists of the time-averaged responses of individual agents. To implement the repeated plays, [41] proposed a privacy-preserving gradient descent method that involves adding Gaussian noise to the Lagrangian's gradient at each iteration.

Hsu et al. [41] calculated bounds for their algorithm's accuracy and the total violation of coupling constraints under (ϵ, δ)-joint differential privacy. They also discussed a way for achieving exact feasibility by applying their algorithm, PrivDude, on a reduced problem with tighten coupling constraints.

To improve Hsu et al.'s method [41], Huang et al. [42] proposed a method for reducing the supply requirement with regards to the number of constraints. While they use the dual decomposition technique similar to [41], their method is based on a version of the multiplicative weight update algorithm with added Laplacian noise, instead of building a private algorithm by adding Guassian noise to the dual gradient descent method. While their method can be applied to the class of optimization problems studied in [41], they put their focus on the packing problem in their paper. To achieve exact feasibility, we can decrease the supply per constraint by a multiplicative factor and run this paper's algorithm.

To improve the running time of their algorithm, which depends cubically on n similar to [41], [42] worked on an algorithm that runs in $\mathcal{O}(n)$ and is a privacy-preserving version of the online packing algorithm[3] in the random-arrival model. Later on, Huang et al. [43] improved this scalable by coming up with another algorithm with $\mathcal{O}(n)$ running time and better minimum supply requirements. This algorithm is based on a noisy version of the dual multiplicitave weight update method with different step sizes and noise scales.

2.5 Deferentially private ADMM

Zhang and Zhu [81] introduced two differentially private algorithm, one based on perturbing the primal variables in ADMM, and another one involving adding noise to dual variables. They analyzed the impacts of adding privacy to only a single iteration of their algorithms. Zhang et al. [82] proposed method for perturbing the penalty parameter of ADMM to improve accuracy and analyzed their algorithm's total privacy loss. Zhang et al. [83] tried enhancing this penalty perturbation mehthod by reusing the results from the odd iterations in even iterations. Half of the updates don't leak private information in this way. Guo and Gong [38] worked on private asynchronous ADMM. In this paper, users train their models locally based on their datasets and only share part of the local model with the server.

[20, 19] used linearly decaying Gaussian noise to perturb the primal variables while maintaining utility. [20] focused on zero-concentrated differential privacy, which makes it possible to achieve a higher accuracy than (ϵ, δ)-differential privacy. [17] proposed a stochastic ADMM-based algorithm for distributed learning with higher accuracy and reduced computation complexity in each local iteration.

Huang et al. [45] proposed a deferentially private ADMM-based distributed learning algorithm that works with an approximate augmented Lagrangian function. Using a first-order approximation for the augmented Lagrangian function makes it possible to use this method for non-sooth objective functions. To get a better privacy accuracy trade-off, their algorithm adaptively changes the variance of the added Gaussian noise. They used the moments accountant method[2] for calculating the total privacy loss. Their method is suitable for a centralized network structure with star topology. [63] proposed a new differntially private algorithm based on an inexact alternating direction method of multipliers (IADMM) and objective perturbation and numerically showed their algorithm gets a better accuracy compared to [45].

Instead of relying on the requirement that each local problem can be perfectly optimized., [18] proposed an algorithm capable of releasing DP-preserving approximate and noisy solutions to the local problems. They improved their algorithm using the sparse vector technique[22], which checks if the current approximate solution is sufficiently different from the previous iteration's solution. When a solution barely differs from the previous iteration, it will be discarded to avoid unnecessary privacy loss accumulation. [77] proposed a two-phase approach for making ADMMs private. In this method, Gaussian noise is added to to local raw data and also the local uploaded model. While this approach strengthens privacy, it hurts the learning performance. [49] introduced a new privacy-preserving primal-dual convex optimization algorithm for federated learning while considering communication efficiency. It is possible to use this method on non-smooth objective functions. They discussed the tradeoff between privacy protection and communication efficiency as well as the trade off between ensuring privacy and learning performance.

As opposed to the previously mentioned works on differentially private ADMMs that focus on distributed ADMMs and federated learning, Shang et al.[67] worked on centralized optimization. Previous works in centralized ADMMs include [12], which focuses on stochastic ADMMs, and [76] about deterministic ADMMs.

Chapter 3

Problem Formulation

Let $c \in \mathbb{R}$ denote the total available capacity. We consider n agents in the system and m public goods. Let $s = (s_1, \ldots, s_m) \in \mathbb{R}^m$ be an m-dimensional vector, where s_j represents the size of each item j. A feasible allocation is a vector $z = (z_1, \ldots, z_m) \in [0, 1]^m$ and $s^T z = \sum_{j=1}^{m} z_j s_j \leq c$. For each j, z_j shows the fraction of item j allocated. We use $u_i = (u_{i1}, \ldots, u_{im}) \in \mathbb{R}_{\geq 0}^m$ to show agent i's utility for every unit of each item. The total utility agent i gains after a round of allocation, equals to $u_i^T z = \sum_j z_j u_{ij}$.

Our goal is to come up with an approximately truthful mechanism that outputs a fair allocation z given agents' utilities $\{u_i\}_{i=1}^n$. To discuss the notion of fairness our allocation policy is based on, we will first define Pareto efficiency and Sharing incentiveness.

Definition 1 (Pareto Efficiency)**.** *An allocation z is Pareto efficient if there is no other allocation z' such that $u_i(z) \geq u_i(z')$ for every i with at least one strict inequality. In other words, an allocation is Pareto-efficient if no other allocation exists that improves the utility of at least one agent without decreasing the utility of others.*

Definition 2 (Sharing Incentive)**.** *An allocation satisfies sharing incentive if no agent would be better off if we divide the total capacity into N equal parts and give each agent a part.*

Envy-freeness is another fairness property commonly studied in private good allocation. Informally, it means that no agent should prefer another agent's share of allocated resources. Envy-freeness along with Pareto efficiency are the properties that make an allocation fair according to the classical economic theory[72]. Cake cutting is the classic context for envy-freeness when studying fractional allocations [60, 7]. The notion of envy is ill-defined when resources are shared between users. However a similar fairness notion can be defined for public goods based on the notion of proportionality and stablity[47]. The notion of core, which comes from cooperative game theory[33, 66, 6, 14], has been studied in the context of public goods[27, 55, 25, 26, 56] and cache allocation[47]. For an allocation z, a set of agents with size l form a blocking coalition if an allocation z' exists such that $(l/n)u_i(z') \geq u_i(z)$ for every agent in this set with at least one strict inequality. We say that an allocation lies in core if no subset of agents form a blocking coalition. This means that no subset of agents have the incentive to deviate from the outcome by putting together their proportion of resources.

Definition 3 (Core)**.** *An allocation z is a core solution if there is no subset T of agents, that given a $|T|/n$ fraction of the total capacity, can come up with an allocation z' such that $u_i(z') \geq u_i(z)$ for every user i in T and at least for one user j in T, $u_j(z') > u_j(z)$.*

An allocation that lies in the core satisfies both Pareto efficiency and sharing incentiveness. To see why, it is sufficient to set the size of T equal to n and 1, respectively. We can think of the core as a notion of group sharing-incentive[47, 26], meaning that if a proportion of all users, which form the subset T have identical preferences, they should at lest get $|T|/n$ of their maximum possible utility. We can say that a core outcome is sort of Pareto efficient for any subset of agents forming coalition, if we scale the [26] utilities in proportion to the subset's size.

[61] proved that no cache allocation policy can satisfy strategy-proofness, isolation guarantee, and Pareto efficiency at the same time. Since a core allocation satisfies isolation guarantee and Pareto efficiency, it is impossible to have a truthful cache allocation policy for finding a core outcome. Our public good allocation problem is similar to the cache allocation problem discussed in [61]. Our goal is to come up with an approximately truthful mechanism for finding an allocation that approximately lies in the core.

Definition 4 (Approximate Core[26])**.** *For $\delta, \alpha > 0$, an allocation z lies in the (δ, α)-approximate core if for any subset T of agents, there is no allocation z' that uses a $|T|/n$ fraction of the budget, $u_i(z') \geq (1 + \delta)u_i(z) + \alpha$ for every user i in T, and at least one inequality is strict.*

[53] showed that an ϵ-differentially private mechanism for $\epsilon \leq 1$ is also ϵ-approximately dominant strategy truthful. Therefore, we can search for a differentially private mechanism that finds an approximate core solution. [47] and [26] showed that maximizing the Nash welfare objective, which is product of agent's utilities, results in a core solution. Our algorithm, is therefore, based on solving a convex optimization program for finding the Nash welfare solution in a differentially private manner.

As mentioned earlier, $s = (s_1, \ldots, s_m) \in \mathbb{R}^m$ shows the size of each item j, and $u_i = (u_{i1}, \ldots, u_{im}) \in \mathbb{R}^m_{\geq 0}$ represents agent i's utility for every unit of each good. Given a capacity of c, to find an allocation, $z \in \mathbb{R}^m$, that lies in the core, we can solve the following optimization problem.

$$\begin{aligned} \text{Max.} \quad & \sum_{i=1}^{n} \ln(u_i^T z), \\ \text{s.t.} \quad & s^T z \leq c, \\ & 0 \leq z \leq 1. \end{aligned} \tag{3.1}$$

(3.1) is equal to the following optimization problem.

$$\begin{aligned} \text{Max.} \quad & \sum_{i=1}^{n} \ln(u_i^T x_i), \\ \text{s.t.} \quad & z = x_i && \forall i, \\ & s^T x_i \leq c && \forall i, \\ & 0 \leq x_i \leq 1 && \forall i. \end{aligned} \tag{3.2}$$

Compared to (3.1), auxiliary variables $x_i \in \mathbb{R}^m$ are introduced in this optimization problem. Similar to the global consensus problem in [8], the global variable z in this problem acts as a constraint for forcing local x_i variables to agree. Without z, this optimization problem can represent a private good allocation problem where x_i shows each agent's allocation. In this imaginary private good allocation problem, every agent has a budget for themselves, and x_i shows how they decide to allocate items using their capacity.Since we are in fact focusing on public goods and the budget is shared between users, we only need a variable $z \in \mathbb{R}$ to represent the allocation. However, by introducing an auxiliary variable per each agent, in an iterative algorithm for solving (3.2), each user can privately and locally update the allocation x_i. This allocation x_i is computed using agent i's private data and can reveal information about the agent's utility vector. To compute a allocation z in a private manner, we can come up with an algorithm for privately aggregating local variables x_i at each step of the algorithm.

One way that comes to mind for solving (3.2) is to consider z as the private variable of an imaginary agent $n+1$, and turn this optimization problem into a private good allocation problem with $n+1$ agents. This way, to solve this optimization problem privately, we might be able to use the methods proposed in [41] or [42] for jointly private convex programming.

However, since these approaches involve using the dual decomposition method[8] and partial Lagrangians, they cannot be applied to optimization problems with affine constraints. The primal players in these methods take a maximization step at each iteration of the algorithm. However, a partial Lagrangian with affine constraints is unbounded with respect to the primal variables, and hence, it is impossible to take an step that maximizes the partial Lagrangian. A similar problem is pointed out in [8] about using the dual ascent[62] methods on affine objective functions. To bypass this issue, we propose an optimization method for (3.2) that uses an augmented Lagrangian. Methods that use an augmented Lagrangian, converge under more general conditions than dual ascent, including cases when the objective function is unbounded[8]. For more information about dual ascent, dual decomposition, and augmented Lagrangian please refer to the comprehensive paper by Boyd et al. [8].

Let us define $C = \{x \in \mathbb{R}^m \mid s^T x \leq c,\ 0 \leq x \leq 1\}$. For the convex set C, we define a convex indicator function, I_C, as:

$$I_C(x) = \begin{cases} 0 & \text{if } x \in C, \\ \infty & \text{otherwise.} \end{cases}$$

For $x = (x_1, \ldots, x_n)$ and $\gamma = (\gamma_1, \ldots, \gamma_n)$, we can now write the augmented Lagrangian of (3.2) as:

$$\begin{aligned} L^\rho(x, z, \gamma) &= \sum_i L_i^\rho(x_i, z, \gamma_i) \\ &= \sum_i \left(\ln(u_i^T x_i) - I_C(x_i) - \gamma_i^T (x_i - z) - \frac{\rho}{2} \|x_i - z\|_2^2 \right). \end{aligned} \tag{3.3}$$

where for every $i \in \{1, 2, \cdots, n\}$, $\gamma_i \in \mathbb{R}^m$ is a dual variable corresponding to the constraint $z = x_i$, and ρ is the penalty parameter. Our algorithm for solving (3.2), which is based on the Alternating Direction Method of Multipliers (ADMM)[34, 31], is presented in the next section.

Chapter 4

Algorithm

The standard ADMM method solves (3.2) by maximizing the augmented Lagrangian $L^\rho(x, z, \gamma)$ (3.3) with respect to $\{x_i^{(k)}\}_{i=1}^n$ and z in an alternating fashion, followed by updating the dual variables $\{\gamma_i\}_{i=1}^n$. This method takes the following steps at each iteration $k \in \{1, \dots, K\}$, where the dual variable update in (4.1c) uses a step size equal to ρ, the augmented Lagrangian parameter.

$$x_i^{(k)} = \operatorname{argmax}_{x_i}(L_i^\rho(x_i, z^{(k-1)}, \gamma_i^{(k-1)})) \qquad \forall i, \tag{4.1a}$$

$$z^{(k)} = \operatorname{argmax}_{z}(L^\rho(x^{(k)}, z, \gamma^{(k-1)})), \tag{4.1b}$$

$$\gamma_i^{(k)} = \gamma_i^{(k-1)} + \rho(x_i^{(k)} - z^{(k)}) \qquad \forall i. \tag{4.1c}$$

By solving $\partial L^\rho(x^{(k)}, z, \gamma^{(k-1)})/\partial z_j = 0$, we can rewrite step (4.1b) as:

$$z_j^{(k)} = \frac{1}{n}\sum_i x_{ij}^{(k)} + \frac{1}{n\rho}\sum_i \gamma_{ij}^{(k-1)} \qquad \forall j. \tag{4.2}$$

Summing over $\{x_i^{(k)}\}_{i=1}^n$ in the above update involves aggregating private information of different agents. To make the algorithm differentially private, we can add a random noise vector $q^{(k)} = (q_1^{(k)}, \dots, q_m^{(k)})$ drawn from a Gaussian distribution to (4.1b). We leave the discussion on the magnitude of this noise vector for later. After adding noise, (4.2) becomes:

$$z^{(k)} = \frac{1}{n}\sum_i x_i^{(k)} + \frac{1}{n\rho}\sum_i \gamma_i^{(k-1)} + q^{(k)}. \tag{4.3}$$

According to step (4.1c) of the ADMM algorithm, we have:

$$\sum_i \gamma_i^{(k)} = \sum_i \left(\gamma_i^{(k-1)} + \rho(x_i^{(k)} - z^{(k)})\right). \tag{4.4}$$

By replacing $z^{(k)}$ from (4.3) into (4.4), we get $\sum_i \gamma_i^{(k)} = -\rho n q^{(k)}$. This would result in the following update rule for $z^{(k)}$.

$$z^{(k)} = \frac{1}{n}\sum_i x_i^{(k)} - q^{(k-1)} + q^{(k)} = \frac{1}{n}\sum_i x_i^{(k)} + v^{(k)}, \tag{4.5}$$

where $v^{(k)} = q^{(k)} - q^{(k-1)}$ is an m-dimensional Gaussian noise vector.

Compared to (4.3), (4.5) provides a more intuitive z-update rule. This update rule shows how $z^{(k)}$ can be calculated using a noisy average of $\{x_i^{(k)}\}_{i=1}^n$. Algorithm (1) shows the pseudocode of our proposed method. The algorithm sets the magnitude of the noise vector $v^{(k)} = (v_1^{(k)}, \ldots, v_m^{(k)})$ to achieve (ϵ, δ)-differential privacy. Please note that the variance of our Gaussian noise vector decays during the iterative process of the algorithm. The output of Algorithm (1) is determined using the average of $\{z^{(k)}\}_{k=1}^K$. We show in the privacy section that each z-update of the algorithm is differentially private, and hence, due to the robustness of differential privacy (DP) to post-processing, the output of the algorithm is private.

Algorithm 1: PriMA: Private Memory Allocation with ADMM

```
1  **requirements:** $c \geq 1$ (so that we can at least allocate one item),$m \geq c$
       (otherwise, the problem is trivial.), $u_{ij} \geq 0\ \forall i, j$, privacy
       parameters $\epsilon > 0$, $\delta \in (0, 1)$.
2  **input:** $n$, $m$, $s$, $u_i$ for every agent $i$.
3  **parameters:** $\epsilon'_k = ((r^{k-1} - r^k)/(1 - r^K))(\epsilon - \log(1/\delta)/(\alpha - 1))$, $\alpha > 1$, $r > 1$,
       $\gamma_{ij}^0 = 0\ \forall i, j$.
4  **for** $k = 1, \ldots, K$ **do**
5      $x_i^{(k)} = \text{argmax}_{x_i}(L_i^\rho(x_i, z^{(k-1)}, \gamma_i^{(k-1)}))\ \forall i$;
6      $v_j^{(k)} \sim \mathcal{N}(0, m\alpha/2n^2\epsilon'_k)\ \forall j$;
7      $z_j^{(k)} = (1/n)\sum_i x_{ij}^{(k)} + v_j^{(k)}\ \forall j$;
8      $\gamma_i^{(k)} = \gamma_i^{(k-1)} + \rho(x_i^{(k)} - z^{(k)})\ \forall i$;
9  **end**
10 $\bar{z}_j = (1/K)\sum_k z_j^{(k)}\ \forall j$;
11 **output:** $\hat{z}_j = \min(1, \max(0, \bar{z}_j))\ \forall j$.
```

Chapter 5

Privacy

To examine the privacy properties of Algorithm (1), we start by providing a definition for $(\epsilon, \delta)-$differential privacy. We can think of the private data of an agent as an element of a data universe U. A dataset $d \in U^n$ includes private data of n agents. Two datasets d, d' are adjacent if they differ only in one index, meaning that for some i, $d_i \neq d'_i$ and for all $j \neq i$, $d_j = d'_j$.

Definition 5 (Differential privacy [21])**.** *A mechanism $M : U^n \mapsto O$ is (ϵ, δ)-differential private if for every pair of adjacent inputs $x, y \in U^n$ and for every subset of outputs $o \in O$, it satisfies:*

$$\mathbb{P}(M(x) \in S) \leq e^{\epsilon}\mathbb{P}(M(y) \in S) + \delta.$$

We use Rényi differential privacy [54] to analyze the end-to-end privacy guarantee of our algorithm. Compared to other variants of differential privacy, Rényi differential privacy (RDP) provides tighter bounds on the privacy costs of iterative algorithms.

Definition 6 (Rényi differential privacy)**.** *A mechanism $M : U^n \mapsto O$ is (α, ϵ)-RDP with order $\alpha > 1$ if for any two adjacent inputs $x, y \in U^n$, it satisfies:*

$$D_\alpha(M(x) \| M(y)) \triangleq \frac{1}{\alpha - 1} \log \left(\mathbb{E}_{o \sim M(y)} \left[\left(\frac{\mathbb{P}(M(x) = o)}{\mathbb{P}(M(y) = o)} \right)^{\alpha} \right] \right) \leq \epsilon.$$

The following lemma, which is useful for analyzing the total privacy leakage of iterative algorithms, shows how adaptive composition of Rényi differential private subroutines degrades the privacy parameters.

Lemma 7 (Adaptive composition [54]). *Let $M_1 : U^n \mapsto O_1$ be (α, ϵ_1)-RDP and $M_2 : U^n \times O_1 \mapsto O_2$ be (α, ϵ_2)-RDP. The mechanism defined as (W, Z), where $W \sim M_1(x)$ and $Z \sim M_2(x, W)$, satisfies $(\alpha, \epsilon_1 + \epsilon_2)$-RDP.*

The next lemma shows a RDP mechanism also satisfies (ϵ, δ)-differential privacy.

Lemma 8 (RDP to DP [54]). *If M is an (α, ϵ)-RDP mechanism, then it is $(\epsilon+\log(1/\delta)/(\alpha-1), \delta)$-differentially private for any $0 < \delta < 1$.*

We can use Gaussian mechanism to achieve Rényi differential privacy. The magnitude of the Gaussian noise we need to apply to our algorithm is calibrated by the l_2-norm sensitivity.

Definition 9 (L2 sensitivity). *Let $f : U^n \mapsto \mathbb{R}^m$ be an m-dimensional function. The ℓ_2 sensitivity of f is defined as:*

$$\Delta_2(f) = \max_{adjacent\ x,y \in U^n} \|f(x) - f(y)\|_2.$$

Lemma 10 (Gaussian mechanism [54]). *Let $f : U^n \mapsto \mathbb{R}^m$ be a vector-valued function with ℓ_2 sensitivity of $\Delta_2(f)$. Let M be a mechanism that outputs $\mathcal{N}(f(x), \sigma^2 \mathbf{1}_m)$. If $\sigma^2 = \alpha \Delta_2^2(f)/2\epsilon$, then M is (α, ϵ)-RDP.*

Our algorithm is based on injecting a Gaussian mechanism with a linearly decaying variance to perturb the process of updating vector z, which involves aggregating private data of different agents. The following theorem shows the privacy guarantee of Algorithm (1).

Theorem 11. *Given $\epsilon, \delta \in (0, 1)$, the sequence of variables $z^{(1)}, \dots z^{(k)}$ produced by Algorithm (1) satisfies (ϵ, δ)-differential privacy.*

Proof. At each iteration, we use the Gaussian mechanism to add noise to the z vector. Since $x_{ij}^{(k)} \in [0, 1]$, the ℓ_2 sensitivity of the z-update step in Algorithm (1) is $\sqrt{m}/n$. Due to Lemma (10), each z-update step is $(\alpha, \epsilon^{(k)})$-RDP. Therefore, according to Lemma (7), Algorithm (1) is $(\alpha, \bar{\epsilon})$-RDP, where

$$\bar{\epsilon} = \sum_{k=1}^{K} \epsilon'_t = \epsilon'_1(1 - r^K)/(1 - r) = \epsilon - \log(1/\delta)/(\alpha - 1).$$

Finally, it follows from Lemma (8) that Algorithm (1) is (ϵ, δ)-DP. □

Chapter 6

Convergence

6.1 Preliminaries

Consider the following convex optimization problem.

$$\begin{aligned} \text{Min.} \quad & f(w), \\ \text{s.t.} \quad & w \in D, \end{aligned} \tag{6.1}$$

where $f : \mathbb{R}^n \mapsto \mathbb{R}$ is a convex function and $D \in \mathbb{R}^n$ is a convex set. The *subdifferential* of f at $w \in \mathbb{R}^n$ is defined as:

$$\partial f(w) = \{g \in \mathbb{R}^n \mid f(y) \geq f(w) + g^T(y - w) \quad \forall y \in \mathbb{R}^n\}.$$

Consider now the following variational inequalities (VI) problem.

$$\begin{aligned} \text{Find.} \quad & w \in D, \\ \text{s.t.} \quad & g^T(y - w) \geq 0 \quad \forall y \in D, g \in \partial f(w). \end{aligned} \tag{6.2}$$

It is known that the solution sets of (6.1) and (6.2) are equal.

Next, we define $w \in W = \mathbb{R}^{nm} \times \mathbb{R}^m \times \mathbb{R}^{nm}$ and $F(w)$ as follows.

$$w = \begin{pmatrix} x \\ z \\ \gamma \end{pmatrix}, \quad F(w) = \begin{pmatrix} -\gamma \\ \sum_i \gamma_i \\ x - G_{n,m} z \end{pmatrix},$$

where, $G_{n,m}$ is an $(nm \times m)$-dimensional matrix defined as $G_{n,m}^T = (I_m, \dots, I_m)$, where I_m is an identity matrix of size m. Now, consider the following VI problem.

$$\begin{aligned} &\text{Find.} \quad w^* = (x^*, z^*, \gamma^*) \in W, \\ &\text{s.t.} \qquad \theta(x) - \theta(x^*) + (w - w^*)^T F(w^*) \le 0 \quad \forall w \in W. \end{aligned} \tag{6.3}$$

where $\theta(x) = \sum_i \left(\ln(u_i^T x_i) - I_C(x_i) \right)$. Since θ is a concave function, we have $\theta(x) - \theta(x') \le g^T(x - x')$ for all x, x', and $g \in \partial\theta(x')$. Therefore, it can be easily shown that the solution set of (6.3) is equal to that of (3.2).

Similar to [40], we next characterize (6.3) in the following lemma. The proof of this lemma is an incremental extension of Theorem 2.1 in [40] and is deferred to Appendix A.1.1.

Lemma 12. *The solution set of (6.3) is convex and is characterized as:*

$$W^* = \bigcap_{w \in W} \{ w^* \mid \theta(x) - \theta(x^*) + (w - w^*)^T F(w) \le 0 \}.$$

Lemma (12) implies that $\tilde{w} \in W$ is an α-approximate solution of (6.3) for $\alpha \ge 0$ if it satisfies:

$$\theta(x) - \theta(\tilde{x}) + (w - \tilde{w})^T F(w) \le \alpha \quad \forall w \in W. \tag{6.4}$$

6.2 Convergence of Algorithm (1)

In the rest of this section, we show that after K steps, the output of Algorithm (1) satisfies (6.4) in expectation with $\alpha = O(1/K)$, which proves an $O(1/K)$ convergence rate in expectation for Algorithm (1). Our convergence proof closely follows the general convergence proof of the Douglas-Rachford alternating direction method in [40].

Let us start our convergence poof by defining the following matrices.

$$M = \begin{pmatrix} I_{nm} & 0 & 0 \\ 0 & I_m & 0 \\ 0 & -\rho G_{n,m} & I_{nm} \end{pmatrix}, \quad H = \begin{pmatrix} 0 & 0 & 0 \\ 0 & n\rho I_m & 0 \\ 0 & 0 & I_{nm}/\rho \end{pmatrix},$$
$$Q = \begin{pmatrix} 0 & 0 & 0 \\ 0 & n\rho I_m & 0 \\ 0 & -G_{n,m} & I_{nm}/\rho \end{pmatrix}. \tag{6.5}$$

It can be easily verified that $Q = HM$. Moreover, since H is symmetric and positive semidefinite, we use the following notation.

$$\|w - w'\|_H^2 \coloneqq (w - w')^T H (w - w').$$

Next, we define a sequence $\{\tilde{w}^{(k)}\}$ based on the sequence $\{w^{(k)}\}$ generated by Algorithm (1) as:

$$\tilde{w}^{(k)} = (\tilde{x}^{(k)}, \tilde{z}^{(k)}, \tilde{\gamma}^{(k)}) = (x^{(k+1)}, z^{(k+1)}, \gamma^{(k)} + \rho(x^{(k+1)} - G_{n,m} z^{(k)})). \tag{6.6}$$

It can be easily verified that:

$$w^{(k+1)} = w^{(k)} - M(w^{(k)} - \tilde{w}^{(k)}) \quad \forall i.$$

Our goal is to use the sequence $\{\tilde{w}^{(k)}\}$ to analyze the convergence rate. The following lemma shows how much the point $\tilde{w}_i^{(k)}$ deviates from a point in the solution set of (6.3). The proof of this lemma is deferred to Appendix A.1.2.

Lemma 13. *Let $v^{(k)} = (v_1^{(k)}, \ldots, v_m^{(k)})$ be the random noise vector generated by Algorithm (1) and $\{\tilde{w}^{(k)}\}$ and Q be defined according to (6.6) and (6.2), respectively. Then the following inequality holds for all i and $w \in W$.*

$$\theta(x) - \theta(\tilde{x}^{(k)}) + (w - \tilde{w}^{(k)})^T F(w) \le \frac{1}{2}\|w - w^{(k)}\|_H^2 - \frac{1}{2}\|w - w^{(k+1)}\|_H^2 - n\rho(z - \tilde{z}^{(k)})^T \sum_{k'=1}^{k+1} v^{(k')}. \tag{6.7}$$

Theorem 14. *Let $\bar{w}$ be defined as:*

$$\bar{w} = \frac{1}{K}\sum_{k=0}^{K-1} \tilde{w}^{(k)}. \tag{6.8}$$

where $\tilde{w}^{(k)}$ is defined according to (6.6). Consider sequences $\{w^{(k)}\}$ and $\{v^{(k)}\}$ generated by Algorithm (1), and let H be defined according to (6.2). Then we have:

$$\mathbb{E}[\theta(x^*) - \theta(\bar{x}) + (w^* - \bar{w})^T F(w^*)] \le \frac{1}{2K}\|w^* - w^{(0)}\|_H^2 + \frac{\alpha\rho m^2(\alpha - 1)}{2Knr^{K-1}(\epsilon - \log(1/\delta))}\left(\frac{1 - r^K}{1 - r}\right)^2. \tag{6.9}$$

Proof. Summing (6.7) over $k = 0, 1, \ldots, K-1$, we have:

$$\theta(x) - \frac{1}{K}\sum_{k=0}^{K-1}\theta(\tilde{x}^{(k)}) + (w-\bar{w})^T F(w) \leq \frac{1}{2K}\|w - w^{(0)}\|_H^2 \\ - \frac{n\rho}{K}\sum_{k=0}^{K-1}(z - \tilde{z}^{(k)})^T v^{(k+1)}. \tag{6.10}$$

Since $\theta(x)$ is concave, we have:

$$\theta(\frac{1}{K}\sum_{k=0}^{K-1}\tilde{x}^{(k)}) \geq \frac{1}{K}\sum_{k=0}^{K-1}\theta(\tilde{x}^{(k)}). \tag{6.11}$$

Combining (6.8), (6.10), and (6.11), we have:

$$\theta(x) - \theta(\bar{x}) + (w-\bar{w})^T F(w) \leq \frac{1}{2K}\|w - w^{(0)}\|_H^2 \\ - \frac{np}{K}\sum_{k=0}^{K-1}(z - \tilde{z}^{(k)})^T v^{(k+1)}. \tag{6.12}$$

Since $\mathbb{E}\left[v^{(k)}\right] = 0$ for every $k = 1, \dots, K$, we have the following:

$$\begin{aligned}
&-\mathbb{E}\left[\frac{1}{K}\sum_{k=0}^{K-1}(z^* - \tilde{z}^{(k)})^T v^{(k+1)}\right] \\
&= \mathbb{E}\left[\frac{1}{K}\sum_{k=0}^{K-1}(\tilde{z}^{(k)})^T v^{(k+1)}\right] - \mathbb{E}\left[\frac{1}{K}\sum_{k=0}^{K-1}(z^*)^T v^{(k+1)}\right] \\
&= \mathbb{E}\left[\frac{1}{K}\sum_{k=0}^{K-1}(z^{(k+1)})^T v^{(k+1)}\right] \\
&= \mathbb{E}\left[\frac{1}{K}\sum_{k=0}^{K-1}(\frac{1}{n}\sum_i x_i^{(k+1)} + v^{(k+1)})^T v^{(k+1)}\right] \\
&= \frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[\|v^{(k+1)}\|^2\right] = \frac{1}{K}\sum_{k=0}^{K-1} Var\left[v^{(k+1)}\right] \\
&= \frac{1}{K}\sum_{k=0}^{K-1}(m^2\alpha/2n^2\epsilon'_{k+1}) \\
&= \frac{\alpha m^2(\alpha-1)}{2Kn^2r^{K-1}(\epsilon - \log(1/\delta))}\left(\frac{1-r^K}{1-r}\right)^2. \qquad (6.13)
\end{aligned}$$

Given (6.13), if we take expectation on both sides of (6.12) we will have (6.9).

□

Chapter 7

Feasibility

For the sake of simplicity, we assume in this section that $s_j = 1$ for every attribute of the vector s in (3.2). In other words, we assume the items' sizes are equal. If items have different sizes, since we assume that the goods are divisible, we can easily divide the goods into equal partitions.

Theorem 15. *Let the sequences $\{x^{(k)}\}$, $\{z^{(k)}\}$, and $\{v^{(k)}\}$ be generated by Algorithm (1). With probability at least $1 - 2\exp(-\xi^2)$ this algorithm produces an output $\hat{z}$ such that*

$$s^T \hat{z} \leq c + \mathcal{O}\left(\frac{\xi m \sqrt{\sum_j s_j^2}}{n}\right) \tag{7.1}$$

Proof. According to the definition of $\bar{z}$ in Line (10) of Algorithm (1),

$$\begin{aligned}\bar{z} &= \frac{1}{K}\sum_k z^{(k)} \\ &= \frac{1}{K}\sum_k \Big(\frac{1}{n}\sum_i x_i^{(k)} + v^{(k)}\Big) \\ &= \frac{1}{n}\sum_i \bar{x}_i + \bar{v}.\end{aligned}$$

where $\bar{v} = \frac{1}{K}\sum_{k=1}^{K} v^{(k)}$. Based on the relationship between $\bar{z}$ and $\hat{z}$ in Lines (10)–(11) of Algorithm (1),

$$\begin{aligned}\hat{z} &= \min\left(1, \max\left(0, \bar{z}\right)\right) \\ &= \min(1, \max(0, \frac{1}{n}\sum_i \bar{x_i} + \bar{v})).\end{aligned}$$

Since $x_i^{(k)} \in C$ for every i and k, $\bar{x_i} \in C$ too, which means $\bar{x_i} \in [0, 1]$. Hence, we can rewrite $\hat{z}$ as follows

$$\hat{z} = \frac{1}{n}\sum_i \bar{x_i} + \hat{v}. \tag{7.2}$$

where the clipped noise $\hat{v}$ is defined as follows.

$$\hat{v} = \min(1 - \frac{1}{n}\sum_i \bar{x_i}, \max(-\frac{1}{n}\sum_i \bar{x_i}, \bar{v})).$$

During the iterative process of Algorithm (1), for every i and k: $s^T x_i^{(k)} \leq c$. Therefore, for every i, $(1/n)\sum_{i=1}^{n} s^T x_i^{(k)} \leq c$ and hence, $s^T(z^{(k)} - v^{(k)}) \leq c$.

This implies that $(1/K)\sum_{k=1}^{K}(s^T(z^{(k)} - v^{(k)})) \leq c$. Which means $s^T(\bar{z} - \bar{v}) \leq c$. However, according to (7.2), $\bar{z} - \bar{v} = \hat{z} - \hat{v} = (1/n)\sum_i \bar{x_i}$. Therefore, $s^T(\hat{z} - \hat{v}) \leq c$. Finally, we have $s^T\hat{z} \leq c + s^T\hat{v}$, and since $|\hat{v}| \leq |\bar{v}|$:

$$s^T\hat{z} \leq c + s^T\hat{v} \leq c + s^T|\hat{v}| \leq c + s^T|\bar{v}|.$$

Since $s = (1, \cdots, 1) \in \mathbb{R}^m$, $s^T|\bar{v}| = |s^T\bar{v}|$. Therefore:

$$s^T\hat{z} \leq c + |s^T\bar{v}|.$$

Based on the Hoeffding bound, since $s^T\bar{v} = \sum_j s_j\bar{v_j} = \frac{1}{K}\sum_{j,k} s_j\bar{v_j}^{(k)}$,

$$\begin{aligned}
\mathbb{P}[|s^T\bar{v}| \geq \beta] &\leq 2\exp\left(-\frac{\beta^2}{2\sum_{j,k} Var[s_j^2\bar{v_j}^{(k)}]}\right)\\
&= 2\exp\left(-\frac{\beta^2}{2m(\sum_j s_j^2)(\sum_k Var[\bar{v_1}^{(k)}])}\right)\\
&= 2\exp\left(-\frac{\beta^2}{2m(\sum_j s_j^2)(\sum_k m\alpha/2n^2\epsilon_k')}\right)\\
&= 2\exp\left(-\frac{\beta^2 n^2}{\alpha m^2(\sum_j s_j^2)(\epsilon - \log(1/\delta)/(\alpha-1))}\right)
\end{aligned}$$

We can complete the proof by setting $\beta^2 = \xi^2\left(\frac{\alpha m^2(\sum_j s_j^2)(\epsilon-\log(1/\delta)/(\alpha-1))}{n^2}\right)$.

$\square$

Chapter 8

Experiments

To empirically test our method, we implement our algorithm using Python and CVXPY[16], which is a Python-based modeling language for convex optimization problems. We test our method on synthetic data generated in three scenarios. In all scenarios, we set the number of agents n equal to 10000 and considered $m = 10$ public goods. In these scenarios, $c = m/2 = 5$, $\rho = 0.01$, and $s_j = 1$ for all j.

In Line (5) of Algorithm (1), each agent takes an x-update step by finding an x_i that maximizes $L_i^{\rho}(x_i, z^{(k-1)}, \gamma_i^{(k-1)})$. This step of the algorithm could be executed in a distributed way. While we run our experiments on a single machine, to speed up the execution time of the algorithm, we use multi-processing to run the x-update step of the algorithm in a parallel manner on multiple CPU cores. The number of parallel processes we created for this step of the algorithm are equal to the number of CPU cores. Since the machine we use for running the algorithm has 256 cores, this parallel implementation of the algorithm significantly improves the performance. In each process, the CVXPY package is used for calculating $x_i^{(k)}$ of $10000/p$ agents, where p denotes the number of processes. To calculate $z^{(k)}$ at the k^{th} step of the algorithm, we aggregate the results calculated in the x-update step and use the NumPy[58] package to add Gaussian noise to the result. We implement the rest of our method according to Algorithm (1).

For all the test scenarios, we set the privacy parameters $\epsilon = 0.1$, $\delta = 10^{-3}$, $r = 0.01$, and we calculate α as follows:

$$\alpha = \frac{log(\delta^{-1})}{(1-\mu)\epsilon + 1},$$

where $\mu = 0.5$.

8.1 Synthetic Scenario 1

In the first scenario, we assume each agent i has a type t_i, and agents' types determine their utility functions. For agent i, $u_i(x) = u_{t_i}^T x$ and t_i is a random number between 1 and 20 selected using a uniform distribution. For each type $t \in [1, \ldots, 20]$, $u_t \in \mathbb{R}^m$ is a random vector that its attributes are random numbers between 0 and 1, generated using a uniform distribution.

For each scenario, we generate random utilities once, and run the algorithm on that data 100 times. Let $f(z^{(k)})_q$ denote the value of $f(z)$ at the k^{th} step of the q^{th} run of the algorithm. To evaluate the convergence properties of our algorithm, we plot the trajectory of $(1/q)\sum_{q=1}^{100} f(z^{(k)})_q$ and $(1/q)\sum_{q=1}^{100} f(x^{(k)})_q$.

The following figures show good convergence properties considering the added noise to z. We can observe in Figure 8.2 that the value of $f(z)$ is increasing during the execution of the algorithm and the difference between $f(z^{(k)})$ and $f(z^{(k-1)})$ seems to decrease as k gets larger. This shows that despite the added noise to z, $f(z)$ is getting closer to $f(z^*)$. Please note that the value of $f(z^{(k)})$ is large in Figure 8.2 since $f(z^{(k)})$ is the sum of $f_i(z^{(k)})$ over i and $n = 10000$ is a large number.

Figure 8.1 evaluates the convergence of $f(x) = \sum_i f_i(x_i)$. This plot shows that as k increases, we reach an steady state. Since the algorithm is trying to make different agents and their x_i to agree with each other, the value of $f(x)$ decreases before reaching the steady state. At the initial steps of the algorithm, when x_i of different agents do not agree with each other, it is easier for $f_i(x_i)$ to be bigger.

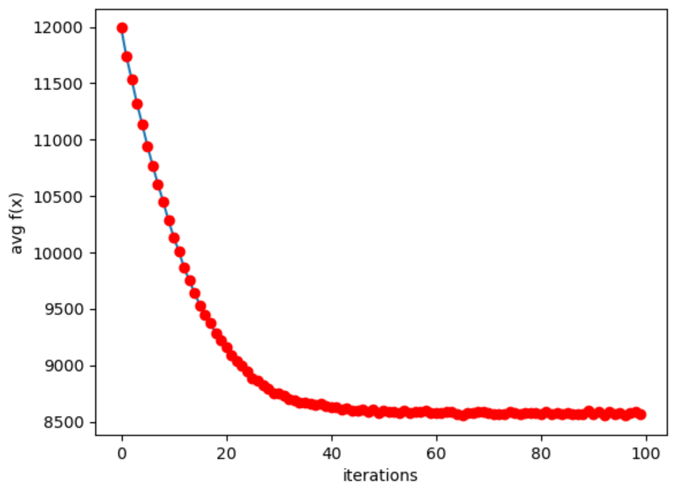

Figure 8.1: Convergence of $f(z^{(x)})$ in the first scenario.

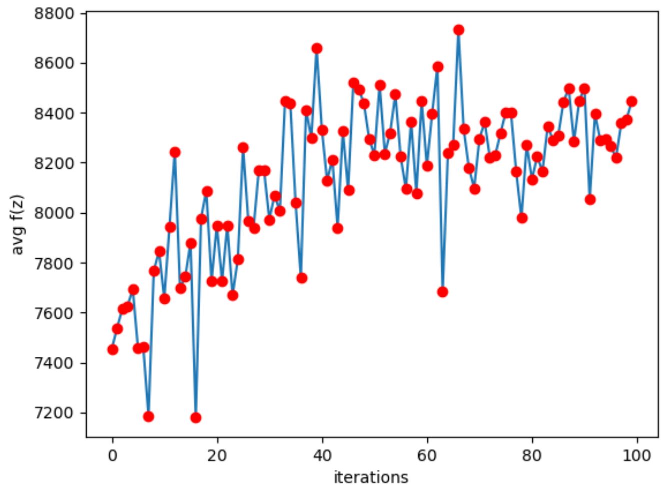

Figure 8.2: Convergence of $f(z^{(k)})$ in the first scenario.

8.2 Synthetic Scenario 2

We consider three types of agents in the second scenario, where for each agent i, $\mathbb{P}(t_i = 1) = 1/2, \mathbb{P}(t_i = 2) = 1/3$, and $\mathbb{P}(t_i = 3) = 1/6$. Similar to the first scenario, $u_i(x) = u_{t_i}^T x$ for each i, where $(u_t)_j \in [0, 1]$ is selected using a uniform distribution for every $t \in 1, 2, 3$.

Figure 8.3 and Figure 8.4 show similar convergence properties as Figure 8.1 and Figure 8.2.

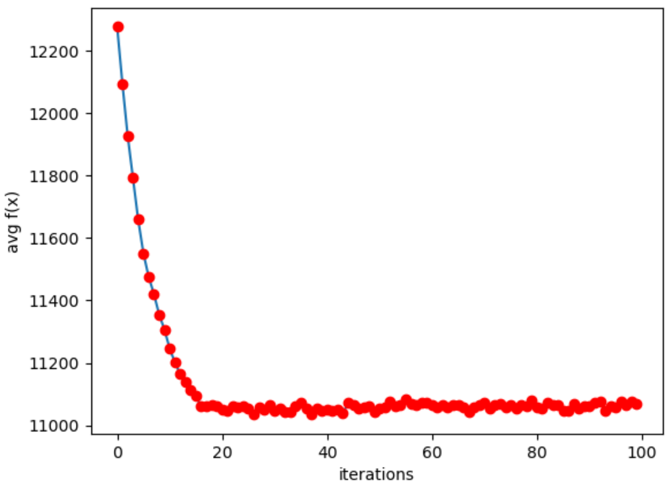

Figure 8.3: Convergence of $f(x^{(k)})$ in the second scenario.

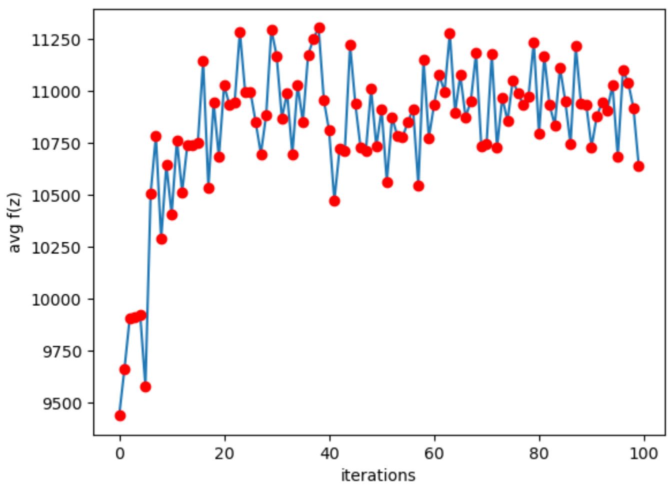

Figure 8.4: Convergence of $f(z^{(k)})$ in the second scenario.

8.3 Synthetic Scenario 3

In the third scenario, agents do not have types. We assume agents have linear utilities $u_i(x) = u_i^T x$, there is an item that every agent is interested in, and each agent's utility for other goods is randomly generated. In this scenario, the random variable $u_{ij} \in [0, 1]$ comes from a uniform distribution for $j \in [2, \ldots, m]$, and $u_{i0} = 1$ for every i.

Figure 8.5 and Figure 8.6 shows that it is harder for the algorithm to converge when each of the 10000 agent has a different random utility vector and agents do not have a specific type. This happens since the number of agents, 10000, is way larger than the number of public goods they try to agree on.

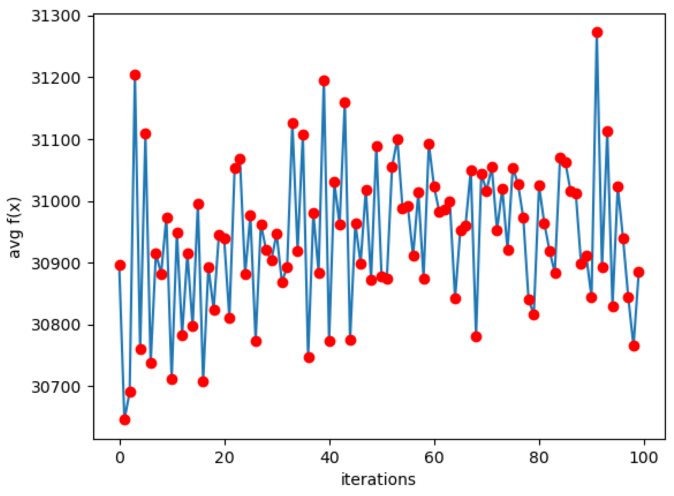

Figure 8.5: Convergence of $f(x^{(k)})$ in the third scenario.

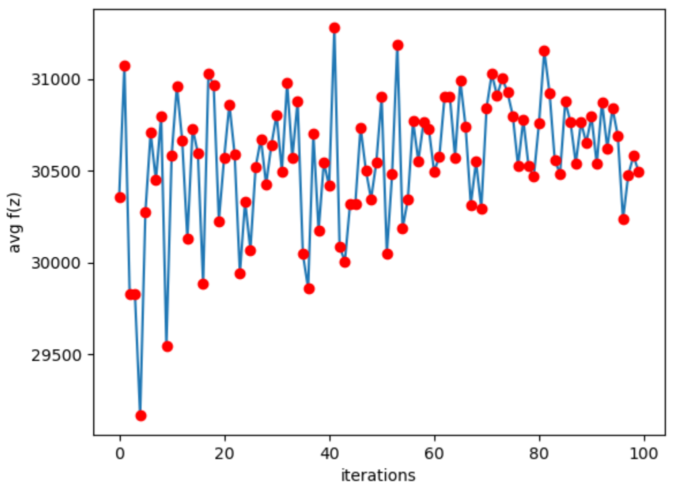

Figure 8.6: Convergence of $f(z^{(k)})$ in the third scenario.

References

[1] What is pb? https://www.participatorybudgeting.org/what-is-pb/, 2022.

[2] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 308–318, 2016.

[3] Shipra Agrawal and Nikhil R Devanur. Fast algorithms for online stochastic convex programming. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1405–1424, 2014.

[4] S Airiau, H Aziz, I Caragiannis, J Kruger, and J Lang. Positional social decision schemes: Fair and efficient portioning. In *Proceedings of the 7th International Workshop on Computational Social Choice (COMSOC)*, 2018.

[5] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: A meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012.

[6] Robert J Aumann. Markets with a continuum of traders. *Econometrica: Journal of the Econometric Society*, pages 39–50, 1964.

[7] Haris Aziz and Simon Mackenzie. A discrete and bounded envy-free cake cutting protocol for any number of agents. In *Proceedings of the 57th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 416–427, 2016.

[8] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[9] Eric Budish, Yeon-Koo Che, Fuhito Kojima, and Paul Milgrom. Designing random allocation mechanisms: Theory and applications. *American Economic Review*, 103(2):585–623, 2013.

[10] Yves Cabannes. Participatory budgeting: A significant contribution to participatory democracy. *Environment and urbanization*, 16(1):27–46, 2004.

[11] Bogdan Caprita, Wong Chun Chan, Jason Nieh, Clifford Stein, and Haoqiang Zheng. Group ratio round-robin: O(1) proportional share scheduling for uniprocessor and multiprocessor systems. In *Proceedings of the USENIX Annual Technical Conference (ATC), General Track*, pages 337–352, 2005.

[12] Chen Chen and Jaewoo Lee. Rényi differentially private ADMM for non-smooth regularized optimization. In *Proceedings of the 10th ACM Conference on Data and Application Security and Privacy (CODASPY)*, pages 319–328, 2020.

[13] Edward H Clarke. Multipart pricing of public goods. *Public choice*, pages 17–33, 1971.

[14] Gerard Debreu and Herbert Scarf. A limit theorem on the core of an economy. *International Economic Review*, 4(3):235–246, 1963.

[15] Alan Demers, Srinivasan Keshav, and Scott Shenker. Analysis and simulation of a fair queueing algorithm. *ACM SIGCOMM Computer Communication Review*, 19(4):1–12, 1989.

[16] Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *The Journal of Machine Learning Research*, 17(1):2909–2913, 2016.

[17] Jiahao Ding, Sai Mounika Errapotu, Haijun Zhang, Yanmin Gong, Miao Pan, and Zhu Han. Stochastic admm based distributed machine learning with differential privacy. In *Proceedings of the 15th International conference on security and privacy in communication systems (SecureComm)*, pages 257–277, 2019.

[18] Jiahao Ding, Jingyi Wang, Guannan Liang, Jinbo Bi, and Miao Pan. Towards plausible differentially private ADMM based distributed machine learning. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management (CIKM)*, pages 285–294, 2020.

[19] Jiahao Ding, Xinyue Zhang, Mingsong Chen, Kaiping Xue, Chi Zhang, and Miao Pan. Differentially private robust ADMM for distributed machine learning. In *Proceedings of the IEEE International Conference on Big Data (Big Data)*.

[20] Jiahao Ding, Xinyue Zhang, Mingsong Chen, Kaiping Xue, Chi Zhang, and Miao Pan. Differentially private robust ADMM for distributed machine learning. pages 1302–1311, 2019.

[21] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the 3rd Conference on Theory of Cryptography (TCC)*, pages 265–284, 2006.

[22] Cynthia Dwork, Moni Naor, Omer Reingold, Guy N Rothblum, and Salil Vadhan. On the complexity of differentially private data release: Efficient algorithms and hardness results. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC)*, pages 381–390, 2009.

[23] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Theoretical Computer Science*, 9(3-4):211–407, 2014.

[24] Hugh Everett III. Generalized lagrange multiplier method for solving problems of optimum allocation of resources. *Operations Research*, 11(3):399–417, 1963.

[25] Brandon Fain, Ashish Goel, and Kamesh Munagala. The core of the participatory budgeting problem. In *Proceedings of the 12th International Conference on Web and Internet Economics (WINE)*, pages 384–399, 2016.

[26] Brandon Fain, Kamesh Munagala, and Nisarg Shah. Fair allocation of indivisible public goods. In *Proceedings of the 19th ACM Conference on Economics and Computation (EC)*, pages 575–592, 2018.

[27] Duncan K Foley. Lindahl's solution and the core of an economy with public goods. *Econometrica: Journal of the Econometric Society*, 38(1):66–72, 1970.

[28] Michel Fortin and Roland Glowinski. *Augmented Lagrangian methods: Applications to the numerical solution of boundary-value problems.* Elsevier, 2000.

[29] Yoav Freund and Robert E Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29(1-2):79–103, 1999.

[30] Eric J Friedman, Vasilis Gkatzelis, Christos-Alexandros Psomas, and Scott Shenker. Fair and efficient memory sharing: Confronting free riders. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, number 01, pages 1965–1972, 2019.

[31] Daniel Gabay and Bertrand Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & Mathematics with Applications*, 2(1):17–40, 1976.

[32] Ali Ghodsi, Matei Zaharia, Benjamin Hindman, Andy Konwinski, Scott Shenker, and Ion Stoica. Dominant resource fairness: Fair allocation of multiple resource types. In *Proceedings of the 8th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2011.

[33] Donald Bruce Gillies. *Some theorems on n-person games*. Princeton University, 1953.

[34] Roland Glowinski and Americo Marroco. Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de dirichlet non linéaires. *Revue française d'automatique, informatique, recherche opérationnelle. Analyse numérique*, 9(R2):41–76, 1975.

[35] Pawan Goyal, Harrick M Vin, and Haichen Chen. Start-time fair queueing: A scheduling algorithm for integrated services packet switching networks. In *Proceedings on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOM)*, pages 157–168, 1996.

[36] Theodore Groves. Incentives in teams. *Econometrica: Journal of the Econometric Society*, pages 617–631, 1973.

[37] Theodore Groves and John Ledyard. Optimal allocation of public goods: A solution to the "free rider" problem. *Econometrica: Journal of the Econometric Society*, pages 783–809, 1977.

[38] Yuanxiong Guo and Yanmin Gong. Practical collaborative learning for crowdsensing in the internet of things with differential privacy. In *Proceedings of the IEEE Conference on Communications and Network Security (CNS)*, pages 1–9, 2018.

[39] MT Hale and M Egerstedty. Differentially private cloud-based multi-agent optimization with constraints. In *Proceedings of the American Control Conference (ACC)*, pages 1235–1240, 2015.

[40] Bingsheng He and Xiaoming Yuan. On the O(1/n) convergence rate of the Douglas-Rachford alternating direction method. *SIAM Journal on Numerical Analysis*, 50(2):700–709, 2012.

[41] Justin Hsu, Zhiyi Huang, Aaron Roth, and Zhiwei Steven Wu. Jointly private convex programming. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 580–599, 2016.

[42] Zhiyi Huang and Xue Zhu. Near optimal jointly private packing algorithms via dual multiplicative weight update. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 343–357, 2018.

[43] Zhiyi Huang and Xue Zhu. Scalable and jointly differentially private packing. *arXiv preprint arXiv:1905.00767*, 2019.

[44] Zonghao Huang and Yanmin Gong. Differentially private ADMM for convex distributed learning: Improved accuracy via multi-step approximation. *arXiv preprint arXiv:2005.07890*, 2020.

[45] Zonghao Huang, Rui Hu, Yuanxiong Guo, Eric Chan-Tin, and Yanmin Gong. DP-ADMM: ADMM-based distributed learning with differential privacy. *IEEE Transactions on Information Forensics and Security*, 15:1002–1012, 2019.

[46] Kamal Jain and Vijay V. Vazirani. Eisenberg-Gale markets: Algorithms and structural properties. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC)*, pages 364–373, 2007.

[47] Mayuresh Kunjir, Brandon Fain, Kamesh Munagala, and Shivnath Babu. ROBUS: Fair cache allocation for data-parallel workloads. In *Proceedings of the 2017 ACM International Conference on Management of Data (SIGMOD)*, page 219–234, 2017.

[48] Beatrice Laurent and Pascal Massart. Adaptive estimation of a quadratic functional by model selection. *Annals of Statistics*, pages 1302–1338, 2000.

[49] Yiwei Li, Shuai Wang, Tsung-Hui Chang, and Chong-Yung Chi. Federated stochastic primal-dual learning with differential privacy. *arXiv preprint arXiv:2204.12284*, 2022.

[50] Erik Lindahl. Just taxation—a positive solution. In *Classics in the Theory of Public Finance*, pages 168–176. 1958.

[51] László Lovász and Santosh Vempala. The geometry of logconcave functions and sampling algorithms. *Random Structures & Algorithms*, 30(3):307–358, 2007.

[52] Laurent Massoulié and James Roberts. Bandwidth sharing: Objectives and algorithms. In *Proceedings of the 18th Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (INFOCOM).*

[53] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 94–103, 2007.

[54] Ilya Mironov. Rényi differential privacy. In *Proceedings of the 30th IEEE Computer Security Foundations Symposium (CSF)*, pages 263–275, 2017.

[55] Thomas J Muench. The core and the Lindahl equilibrium of an economy with a public good: An example. *Journal of Economic Theory*, 4(2):241–255, 1972.

[56] Kamesh Munagala, Yiheng Shen, Kangning Wang, and Zhiyi Wang. Approximate core for committee selection via multilinear extension and market clearing. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2229–2252, 2022.

[57] John F Nash Jr. The bargaining problem. *Econometrica: Journal of the Econometric Society*, pages 155–162, 1950.

[58] Travis E Oliphant. *A Guide to NumPy*, volume 1. 2006.

[59] Hua Ouyang, Niao He, Long Tran, and Alexander Gray. Stochastic alternating direction method of multipliers. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pages 80–88, 2013.

[60] Ariel D Procaccia. Cake cutting: Not just child's play. *Communications of the ACM*, 56(7):78–87, 2013.

[61] Qifan Pu, Haoyuan Li, Matei Zaharia, Ali Ghodsi, and Ion Stoica. FairRide: Near-optimal, fair cache sharing. In *Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 393–406, 2016.

[62] R Tyrrell Rockafellar. *Convex analysis*, volume 18. 1970.

[63] Minseok Ryu and Kibaek Kim. Differentially private federated learning via inexact ADMM. *arXiv preprint arXiv:2106.06127*, 2021.

[64] Minseok Ryu and Kibaek Kim. Differentially private federated learning via inexact ADMM with multiple local updates. *arXiv preprint arXiv:2202.09409*, 2022.

[65] Paul A Samuelson. The pure theory of public expenditure. *The Review of Economics and Statistics*, pages 387–389, 1954.

[66] Herbert E Scarf. The core of an N person game. *Econometrica: Journal of the Econometric Society*, pages 50–69, 1967.

[67] Fanhua Shang, Tao Xu, Yuanyuan Liu, Hongying Liu, Longjie Shen, and Maoguo Gong. Differentially private ADMM algorithms for machine learning. *IEEE Transactions on Information Forensics and Security*, 16.

[68] Madhavapeddi Shreedhar and George Varghese. Efficient fair queuing using deficit round-robin. *IEEE/ACM Transactions on networking*, 4(3):375–385, 1996.

[69] Ion Stoica, Hussein Abdel-Wahab, Kevin Jeffay, Sanjoy K Baruah, Johannes E Gehrke, and C Greg Plaxton. A proportional share resource allocation algorithm for real-time, time-shared systems. In *Proceedings of the 17th IEEE Real-Time Systems Symposium*, pages 288–299, 1996.

[70] Ion Stoica, Scott Shenker, and Hui Zhang. Core-stateless fair queueing: Achieving approximately fair bandwidth allocations in high speed networks. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pages 118–130, 1998.

[71] Shanjiang Tang, Qifei Chai, Ce Yu, Yusen Li, and Chao Sun. Balancing fairness and efficiency for cache sharing in semi-external memory system. In *Proceedings of the 49th International Conference on Parallel Processing (ICPP)*, pages 1–11, 2020.

[72] Hal R Varian. Two problems in the theory of fairness. *Journal of Public Economics*, 5(3-4):249–260, 1976.

[73] William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of finance*, 16(1):8–37, 1961.

[74] Carl A Waldspurger and William E Weihl. Lottery scheduling: Flexible proportional-share resource management. In *Proceedings of the 1st USENIX conference on Operating Systems Design and Implementation (OSDI)*, 1994.

[75] Carl A Waldspurger and William E Weihl. *Stride scheduling: Deterministic proportional share resource management*. Massachusetts Institute of Technology. Laboratory for Computer Science, 1995.

[76] Puyu Wang and Hai Zhang. Differential privacy for sparse classification learning. *Neurocomputing*, 375:91–101, 2020.

[77] Xin Wang, Hideaki Ishii, Linkang Du, Peng Cheng, and Jiming Chen. Privacy-preserving distributed machine learning via local randomization and ADMM perturbation. *IEEE Transactions on Signal Processing*, 68:4226–4241, 2020.

[78] Yinghao Yu, Wei Wang, Jun Zhang, and Khaled Ben Letaief. LACS: Load-aware cache sharing with isolation guarantee. In *Proceedings of the 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 207–217, 2019.

[79] Yinghao Yu, Wei Wang, Jun Zhang, Qizhen Weng, and Khaled Ben Letaief. OpuS: Fair and efficient cache sharing for in-memory data analytics. In *Proceedings of the 38th IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 154–164, 2018.

[80] Hui Zhang and Jon CR Bennett. WF2Q: Worst-case fair weighted fair queueing. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, volume 96, pages 120–128, 1996.

[81] Tao Zhang and Quanyan Zhu. Dynamic differential privacy for ADMM-based distributed classification learning. *IEEE Transactions on Information Forensics and Security*, 12(1):172–187, 2016.

[82] Xueru Zhang, Mohammad Mahdi Khalili, and Mingyan Liu. Improving the privacy and accuracy of ADMM-based distributed algorithms. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 5796–5805, 2018.

[83] Xueru Zhang, Mohammad Mahdi Khalili, and Mingyan Liu. Recycled ADMM: Improve privacy and accuracy with less computation in distributed algorithms. In *Proceedings of the 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 959–965, 2018.

APPENDICES

Appendix A

Lemmas

A.1 Proof of Lemmas

A.1.1 Proof of Lemma (12)

Proof. First, note that $w^T F(w) = 0$ for all w. Therefore, $(w - w')^T F(w - w') = 0$ for any w and w'. Given that F is a linear function, we have $(w-w')^T F(w) = (w-w')^T F(w')$. For any $w^* \in W^*$, we have $\theta(x) - \theta(x^*) + (w - w^*)^T F(w) \leq 0 \; \forall w$. Replacing $(w - w^*)^T F(w) = (w - w^*)^T F(w^*)$, we can conclude that w^* is a solution to (6.2). Next, suppose that w^* is a solution to (6.2). For such w^*, we have $\theta(x) - \theta(x^*) + (w - w^*)^T F(w^*) \leq 0$ for all w. We can replace $(w - w^*)^T F(w^*) = (w - w^*)^T F(w)$ to conclude that $w^* \in W^*$. Finally, we prove the convexity of W^*. Since θ is a concave function, the following set is a convex set for any w. $\{w^* \mid \theta(x) - \theta(x^*) + (w - w^*)^T F(w) \leq 0\}$. Convexity of W^* follows from the fact that the intersection of convex sets is a convex set. $\square$

A.1.2 Proof of Lemma (13)

Proof. By the VI reformulation of Line (5) of Algorithm (1), we have:

$$\theta(x) - \theta(x^{(k+1)}) + (x - x^{(k+1)})^T(-\gamma^{(k)} - \rho(x^{(k+1)} - G_{n,m} z^{(k)})) \leq 0 \quad \forall x. \tag{A.1}$$

Given (4.3), Line (7) of Algorithm (1) implies that $z^{(k+1)}$ is the solution to:

$$\underset{z}{\text{maximize}} \quad -\sum_i \left((\gamma_i^{(k)})^T (x_i^{(k+1)} - z + q^{(k+1)}) - \frac{\rho}{2} \|x_i^{(k+1)} - z + q^{(k+1)}\|_2^2 \right),$$

The VI reformulation of the above optimization is:

$$(z - z^{(k+1)})^T \left(\sum_i \left(\gamma_i^{(k)} + \rho(x_i^{(k+1)} - z^{(k+1)} + q^{(k+1)}) \right) \right) \leq 0 \quad \forall z. \tag{A.2}$$

Using the notation of $\tilde{w}^{(k)}$, we can respectively rewrite (A.1) and (A.2) as:

$$\theta(x) - \theta(\tilde{x}^{(k)}) + (x - \tilde{x}^{(k)})^T(-\tilde{\gamma}^{(k)}) \leq 0 \quad \forall x, \tag{A.3}$$

and

$$(z - \tilde{z}^{(k)})^T \left(\sum_i \tilde{\gamma}_i^{(k)} + n\rho(z^{(k)} - \tilde{z}^{(k)}) + n\rho q^{(k+1)} \right) \leq 0 \quad \forall z. \tag{A.4}$$

Given (6.6) and Line (5) of Algorithm (1), we further have:

$$(\tilde{x}^{(k)} - G_{n,m}\tilde{z}^{(k)}) - G_{n,m}(z^{(k)} - \tilde{z}^{(k)}) + (\gamma^{(k)} - \tilde{\gamma}^{(k)})/\rho = 0. \tag{A.5}$$

Given that $q^{(k+1)} = \sum_{k'=1}^{k+1} v^{(k')}$, combining (A.3)–(A.5), we have:

$$\begin{aligned} &\theta(x) - \theta(\tilde{x}^{(k)}) + (w - \tilde{w}^{(k)})^T F(\tilde{w}) \leq \\ &- (w - \tilde{w}^{(k)})^T Q(w^{(k)} - \tilde{w}^{(k)}) - n\rho(z - \tilde{z}^{(k)})^T \sum_{k'=1}^{k+1} v^{(k')} \quad \forall w. \end{aligned} \tag{A.6}$$

Given Lemma 3.2 in [40], we have:

$$-(w - \tilde{w}^{(k)})^T Q(w^{(k)} - \tilde{w}^{(k)}) \leq \frac{1}{2}\|w - w^{(k)}\|_H^2 - \frac{1}{2}\|w - w^{(k+1)}\|_H^2 \quad \forall w.$$

Setting $(w - \tilde{w}(k))^T F(\tilde{w}(k)) = (w - \tilde{w}(k))^T F(w)$ and substituting the above inequality in (A.6), we get (6.7), which completes the proof. □