Differential Privacy in Multi-Party Resource Sharing

Utku Karaca
Erasmus University Rotterdam, 3000 DR, Rotterdam P.O. Box 1738, The Netherlands
Ş. İlker Birbil
University of Amsterdam, 11018 TV, Amsterdam P.O. Box 15953, The Netherlands
Sinan Yıldırım
Sabancı University, Istanbul, Turkey
Nurşen Aydın
University of Warwick, Coventry CV4 7AL, United Kingdom
Gizem Mullaoğlu
Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands

Abstract: This study examines a resource-sharing problem involving multiple parties that agree to use a set of capacities together. We start with modeling the whole problem as a mathematical program, where all parties are required to exchange information to obtain the optimal objective function value. This information bears private data from each party in terms of coefficients used in the mathematical program. Moreover, the parties also consider the individual optimal solutions as private. In this setting, the concern for the parties is the privacy of their data and their optimal allocations. We propose a two-step approach to meet the privacy requirements of the parties. In the first step, we obtain a reformulated model that is amenable to a decomposition scheme. Although this scheme eliminates almost all data exchange, it does not provide a formal privacy guarantee. In the second step, we provide this guarantee with a differentially private algorithm at the expense of deviating slightly from the optimality. We provide bounds on this deviation and discuss the consequences of these theoretical results. We also propose a novel modification to increase the efficiency of the algorithm in terms of reducing the theoretical optimality gap. The study ends with a numerical experiment on a planning problem that demonstrates an application of the proposed approach. As we work with a general optimization model, our analysis and discussion can be used in different application areas including production planning, logistics, and network revenue management.

Keywords: collaboration; differential privacy; resource sharing; decomposition

1. Introduction. Efficient use of resources is one of the major objectives in every industry. Through collaboration, companies can coordinate their activities, increase their utilizations, and obtain significant savings. For instance, in freight logistics, companies share less-than-full vehicles to compound the problem of excess capacity (Speranza, 2018). This collaboration helps them to reduce transportation costs and improve their operational efficiencies. Similarly, airlines provide joint services to increase their flight capacity utilization (Topaloglu, 2012). In the automotive industry, Toyota and Fuji Heavy Industries had shared an existing manufacturing site to increase the efficiency in manufacturing (Toyota, 2006). In supply chains, collaborative planning improves production and replenishment decisions (Poundarikapuram and Veeramani, 2004). There are also other examples in lodging, telecommunication, and maritime transportation, where collaborations among companies occur by sharing resources, exchanging information, or providing joint services (Guo and Wu, 2018; Agarwal and Ergun, 2010; Chun et al., 2016).

Although coordinating activities and sharing resources can provide significant benefits to companies, it

also poses several challenges. These partnerships are mainly built around information exchange to coordinate the collective decision-making process. However, individual partners, though often working towards a common goal, can be competitors and may be unwilling or unable to fully disclose sensitive information about their operations (Hyndman et al., 2013; Albrecht and Stadtler, 2015). Sensitive information may involve demand forecasts, selling prices, operational costs, and available capacities. In air-cargo transportation, for example, airlines and freight forwarders collaborate to sell the flight capacity. In this partnership, the freight forwarders keep their demand information, operating costs, and reservation prices private to protect their interests (Amaruchkul et al., 2011). While centralized decision-making is desirable to achieve maximum gain from the collaboration, it may not be realistic for collaborative planning due to restrictions in information exchange. Therefore, decomposition and decentralization approaches have been studied to minimize information sharing among participants (Albrecht and Stadtler, 2015; Ding and Kaminsky, 2020). In decentralized systems, individual parties release only the required information at each step to obtain the optimal allocations of the shared resources with an iterative solution approach. This information can also be exchanged via a central planner. Although information exchange is reduced with a decentralization approach, the parties still have to share information about their operations with the others or the central planner. Specifically, in a network resource sharing setting, this shared information can be related to the optimal capacity allocations or the bid (dual) prices, which may reveal private information, like individual profits and capacity utilizations (Poundarikapuram and Veeramani, 2004; Albrecht and Stadtler, 2015). This data privacy concern in collaborations raises an important question: How can one mathematically guarantee data privacy while conducting optimization in a multi-party resource sharing setting?

1.1 Contributions. We address the question above for a general linear optimization framework, where parties make use of the shared resources to manage their operations. Aside from the shared resources, the parties do not want to reveal directly their private information consisting of the coefficients in the objective function and the individual constraints. To this end, we first introduce a decomposition approach coupled with an iterative solution algorithm, where each party is only required to share only their allotments at each iteration. Then, we mask each allocation by adding a specifically adjusted noise to randomize the output. This randomization allows us to give a data privacy guarantee by using the mathematically rigorous definition of *differential privacy* (Dwork et al., 2006b). As we have an iterative algorithm, this theory dictates using only a finite number of iterations to preserve privacy at a given level. Considering the random output and the limited number of iterations, we observe that our differentially private decomposition algorithm may end up with a suboptimal solution. Therefore, we also provide theoretical bounds on the difference between the approximate and the optimal objective function values. To the best of our knowledge, this is the first formal treatment of data privacy *without any trusted party* in multi-party resource sharing via linear optimization. We further propose a novel modification to increase the efficiency of the private decomposition algorithm in terms of reducing the variance of the noise, and consequently, the optimality gap. We support our analysis with a set of numerical experiments on a production planning example and present the trade-off between

optimality and privacy guarantee. We specify our framework for linear programming as it is arguably the most frequently used optimization tool in practice (Ibaraki and Katoh, 1988; Topaloglu, 2012; Albrecht and Stadtler, 2015). However, we also remark that our work can easily be extended to general convex programs for related applications.

1.2 Review of Related Literature. Collaborative relationships between organizations have received considerable attention in the literature, and it is very popular in many industries including airlines (Wright et al., 2010), logistics and maritime shipping (Agarwal and Ergun, 2010; Verdonck et al., 2013; Lai et al., 2019), and retail (Guo and Wu, 2018). Most of these studies assume complete information exchange among partners, which may not be realistic in practice due to the regulations. Therefore, recent studies focus on the techniques to minimize information sharing in collaborative optimization problems. Poundarikapuram and Veeramani (2004) propose a decentralized decision-making framework based on the L-shaped method for a collaborative planning problem in a supply chain. The centralized collaborative planning problem is separated into a master problem that includes the common variables for all parties and several sub-problems that include private local objectives and variables. The authors present an iterative procedure to address this problem, where the parties can solve their local problems privately and disclose limited information to solve the master problem at each iteration. Topaloglu (2012) focuses on capacity sharing in airline alliances and proposes a decomposition approach to find booking limits for each alliance partner as well as bid prices for shared flights. Kovács et al. (2013) study collaboration in lot-sizing problems with two parties and investigate different solution approaches, including centralized and decentralized methods. To minimize information exchange, the lot-sizing problem is decomposed and solved by each party sequentially in the decentralization approach. Albrecht and Stadtler (2015) propose a scheme to coordinate the collaborative parties in a decentralized environment where parties' local private problems can be modeled as linear programs. In the proposed scheme, each party exchanges proposals by sharing information on the primal optimal solution of the linear programs.

Decomposition approaches have also attracted great attention in the machine learning community. *Federated learning* is initially proposed in Shokri and Shmatikov (2015); Konečnỳ et al. (2016); McMahan et al. (2017), and it focuses on learning tasks in a distributed environment. Commonly, machine learning models require a central training phase where all the data needs to be shared. The use of decomposition approaches enables training locally so that there is no need for data exchange. Even if the data is not shared, adversarial attacks are still possible through the shared model parameters during the training process. There are many studies in the literature to provide privacy guarantees by employing various methods. Geyer et al. (2017) design a differentially private federated learning algorithm. Truex et al. (2019); Yang et al. (2019) study privacy of the federated learning by employing both differential privacy and secure multi-party computation protocols. We refer the reader to Li et al. (2020) and Mothukuri et al. (2021) for extensive literature reviews on federated learning and its privacy, respectively.

One well-known approach to overcoming privacy breaches in data sharing through collaborations is using transformation-based methods. These methods allow reconstructing the given model with altered data while preserving the optimality. For instance, Vaidya (2009) proposes a transformation method in linear programming models where one party owns the objective function, and the other owns the constraints. In a follow up work, Bednarz et al. (2009) show that the proposed method by Vaidya (2009) is valid under specific restrictions. Later, Hong and Vaidya (2014) have revised the initial approach of Vaidya (2009) and improved the level of privacy. Mangasarian (2011) and Mangasarian (2012) present a transformation-based approach to preserve privacy in both vertically and horizontally partitioned linear models. In another study, Dreier and Kerschbaum (2011) show the computational inefficiency of the cryptographic methods in mathematical models and propose a transformation-based approach. Additionally, they provide a security analysis of their approach. All of these studies focus on hiding both input data and the primal decision variables in various settings. Recently, Karaca et al. (2022) study data-privacy in collaborative network revenue management problem using a transformation-based approach. The proposed method keeps both primal and dual variables private to each party. The transformation-based methods have a flaw that either they do not guarantee privacy, or they are computationally not feasible for solving even small-scale problems (Sweeney, 1997; Narayanan and Shmatikov, 2006; Toft, 2009). As a remedy, the notion of differential privacy has been widely studied in the literature.

Differential privacy is a probabilistic definition of privacy that suggests using random perturbations to minimize the probability of revealing specific records from neighboring databases (Dwork et al., 2006b). Zhang and Zhu (2017) present two differentially private algorithms using the alternating direction method of multipliers for the empirical risk minimization problem. Recently, its applications are extended to constrained and distributed optimization algorithms. Huang et al. (2015) consider a cost minimization problem among parties with an additional privacy requirement for the cost function. They employ differential privacy and present a private distributed optimization scheme. Hale and Egerstedty (2015) study a multi-party optimization problem in a cloud structure, where the aim is to optimize global objective function with global inequality constraints. In the study, the parties send all the necessary data to the cloud, whose role is to keep each party's data private. Han et al. (2016) study a differentially private distributed constrained optimization problem in resource allocation where they assume a convex and Lipschitz objective function. They consider only shared resources in a specific structure, called the server-client network structure. Even though this study is conducted for a resource allocation setting, it differs from our study by the direct application of the stochastic subgradient method and its lack of rigorous bounds on approximate optimality.

Hsu et al. (2014) study differential privacy in linear programs. They define variations of neighboring datasets and analyze solution methods for each of the definitions. A study that is closely related to ours is given by Hsu et al. (2016). Similar to our approach, they present a jointly differential private algorithm by applying Lagrangian relaxation to the linking constraints. Unlike our study, however, they focus on a special case where an aggregator collects data from all parties. In our study, we do not assume the existence of a

trusted data aggregator, and we design a differentially private algorithm that also ensures each party's data privacy against the other parties.

2. Methodology. In this section, we introduce the mathematical model for multi-party resource-sharing and our initial attempt to obtain a *data-hiding* solution approach with decomposition. Suppose we have $K > 1$ parties that agree to collectively use $m \geq 1$ *shared resources* with capacities given in the vector $\boldsymbol{c} \in [0, \infty)^m$. In addition, each party k has its m_k *private constraints*; such as individual resources, demand restrictions, production requirements, and flow conservation. We let $\boldsymbol{b}_k \in \mathbb{R}^{m_k}$ be the right-hand-sides of the private constraints of party k. Next, each party k has matrices $\boldsymbol{A}_k \in \mathbb{R}^{m \times n_k}$ and $\boldsymbol{B}_k \in \mathbb{R}^{m_k \times n_k}$ that include the technology coefficients related to the shared and private constraints, respectively. The objective of the model is to find the optimal allocations the parties such that the total utility is maximized. (Without loss of generality, we give a maximization model, and all our discussion can be trivially extended to a minimization setting.) For each k, we let $\boldsymbol{x}_k \in \mathbb{R}^{n_k}$ be the vector of party k's individual variables and $\boldsymbol{u}_k \in \mathbb{R}^{n_k}$ be its utility vector. The canonical multi-party resource-sharing model then becomes

$$
\begin{aligned}
Z_P \;=\; \text{maximize} \quad & \sum_{k=1}^{K} \boldsymbol{u}_k^{\intercal} \boldsymbol{x}_k, && (1)\\
\text{subject to} \quad & \sum_{k=1}^{K} \boldsymbol{A}_k \boldsymbol{x}_k \leq \boldsymbol{c}, && (2)\\
& \boldsymbol{B}_k \boldsymbol{x}_k \leq \boldsymbol{b}_k, \quad k = 1, \ldots, K, && (3)
\end{aligned}
$$

The set of constraints (2) guarantee that the total consumptions of the parties do not exceed the shared capacities, whereas the set of constraints (3) expresses the private constraints of each party k.

We note that our methodological discussion in the current and the next sections can be extended to a general setting, where the objective function and the constraint functions are nonlinear *but* define a convex programming model. We have decided to focus on a linear programming model to simplify the exposition. We also note that the linear model presented in the form of (1)-(3) is very general and used in many real-world applications from production planning to portfolio selection, from logistics to network revenue management (Ibaraki and Katoh, 1988; Topaloglu, 2012; Birbil et al., 2014; Albrecht and Stadtler, 2015). In its current form, the model (1)-(3) can be solved when all input is available. Before raising the privacy concerns, let us first define formally what constitutes the (private) dataset for each party.

DEFINITION 2.1 *(Dataset). In the multi-party resource-sharing problem* (1)-(3)*, for each* $k \in \{1, \ldots, K\}$*, the dataset of party* k *is the collection* $\mathcal{D}_k = \{\boldsymbol{A}_k, \boldsymbol{B}_k, \boldsymbol{b}_k, \boldsymbol{u}_k\}$.

Although the parties agree to solve the resource-sharing problem collectively, they are unwilling to share their datasets with their fellow parties. This is because a dataset consists of sensitive information for its corresponding party. For instance, in a network revenue management problem, a dataset of a party may

consist of private information about its products, capacities, and revenues (Karaca et al., 2022). Given this privacy concern, the main question becomes: How can a party join the others to solve the resource-sharing problem collectively, but keep its dataset private? To address this question, we first propose a decomposition approach, where each party ends up solving a sequence of subproblems without sharing any data with the other parties.

2.1 Data-hiding with Decomposition. Our decomposition approach stems from a natural observation: If the shared capacities are divided among the parties *apriori*, then there is no need to solve the overall problem as each party can independently solve its problem with the allocated capacity. However, ensuring a feasible allocation of the shared resources is not always possible unless we have private information from all parties. In addition, apriori partitioning of the shared capacities can easily lead to a suboptimal solution without maximizing the total utility; see, *e.g.*, the discussion given by Birbil et al. (2014) on network capacity fragmentation). Therefore, we need to devise a scheme that divides the shared capacity optimally among the parties without revealing their private data.

For $k = 1, \ldots, K$, we define a new decision vector $\boldsymbol{s}_k \in \mathbb{R}^m$ to be the allocation to party k from the shared resources. Consequently, we have $\boldsymbol{c} = \sum_{k=1}^{K} \boldsymbol{s}_k$. With this new setting, we can write problem (1)-(3) equivalently as

$$
\begin{aligned}
Z_{\mathcal{P}} \;=\; & \text{maximize} && \sum_{k=1}^{K} \boldsymbol{u}_k^{\intercal} \boldsymbol{x}_k, && (4)\\
& \text{subject to} && \boldsymbol{A}_k \boldsymbol{x}_k \leq \boldsymbol{s}_k, \quad k = 1, \ldots, K, && (5)\\
& && \boldsymbol{B}_k \boldsymbol{x}_k \leq \boldsymbol{b}_k, \quad k = 1, \ldots, K, && (6)\\
& && \sum_{k=1}^{K} \boldsymbol{s}_k = \boldsymbol{c}, && (7)\\
& && \boldsymbol{s}_k \geq \boldsymbol{0}, \quad k = 1, \ldots, K. && (8)
\end{aligned}
$$

This model is almost separable except for the set of constraints (7). We can relax this constraint by introducing a Lagrange multiplier vector $\boldsymbol{\lambda} \in \mathbb{R}^m$. Then, the objective function of the relaxed problem becomes

$$
\mathcal{L}(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{\lambda}) := \boldsymbol{c}^{\intercal} \boldsymbol{\lambda} + \sum_{k=1}^{K} \left(\boldsymbol{u}_k^{\intercal} \boldsymbol{x}_k - \boldsymbol{s}_k^{\intercal} \boldsymbol{\lambda} \right),
$$

where $\boldsymbol{x} := (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K)$ and $\boldsymbol{s} := (\boldsymbol{s}_1, \ldots, \boldsymbol{s}_K)$. If for each party k we further define the *subproblem*

$$
\begin{aligned}
g(\boldsymbol{\lambda}; \mathcal{D}_k) \;:=\; & \text{maximize} && \boldsymbol{u}_k^{\intercal} \boldsymbol{x}_k - \boldsymbol{s}_k^{\intercal} \boldsymbol{\lambda}, && (9)\\
& \text{subject to} && \boldsymbol{A}_k \boldsymbol{x}_k \leq \boldsymbol{s}_k,\\
& && \boldsymbol{B}_k \boldsymbol{x}_k \leq \boldsymbol{b}_k,
\end{aligned}
$$

$$\boldsymbol{s}_k \geq \mathbf{0},$$

and assume that its dual has a feasible solution, then we obtain the Lagrangian dual problem

$$Z_{\mathcal{L}} := \min_{\boldsymbol{\lambda}} \max_{\boldsymbol{x},\boldsymbol{s}} \mathcal{L}(\boldsymbol{x},\boldsymbol{s},\boldsymbol{\lambda}) = \min_{\boldsymbol{\lambda}} \left\{ \boldsymbol{c}^{\intercal}\boldsymbol{\lambda} + \sum_{k=1}^{K} g(\boldsymbol{\lambda};\mathcal{D}_k) \right\}. \tag{10}$$

Since we are dealing with linear programs, the strong duality trivially holds, hence $Z_{\mathcal{P}} = Z_{\mathcal{L}}$. This implies that we can solve dual problem (10) instead of the primal problem (1)-(3). We denote the optimal solution to (10) by $\boldsymbol{x}^*$, $\boldsymbol{s}^*$ and $\boldsymbol{\lambda}^*$.

It is important to observe that the dual problem becomes separable over the parties when Lagrange multiplier vector $\boldsymbol{\lambda}$ is fixed. Luckily, the well-known subgradient method for solving the dual problem is also based on updating the Lagrange multiplier vector iteratively (Bertsekas, 2015). That is, the update from $\boldsymbol{\lambda}^{(t)}$ to $\boldsymbol{\lambda}^{(t+1)}$ takes the form

$$\boldsymbol{\lambda}^{(t+1)} = \boldsymbol{\lambda}^{(t)} - \nu^{(t)} \left(\boldsymbol{c} - \sum_{k=1}^{K} \boldsymbol{s}_k^{(t)} \right), \tag{11}$$

where the superscript shows the iteration number and $\nu^{(t)}$ is the step-size. Assuming that norms of subgradients and the distance $\|\boldsymbol{\lambda}^{(0)} - \boldsymbol{\lambda}^*\|$ are bounded, the algorithm is guaranteed to decrease the suboptimality bound at the rate of $\frac{1}{\sqrt{t}}$ for *any types of step-size* choice suggested in Boyd et al. (2003).

The outline of the dual decomposition approach is illustrated for one iteration in Figure 1. The primal solution $\boldsymbol{s}_k^{(t)}$ is obtained by asking party k to solve $g(\boldsymbol{\lambda}^{(t)};\mathcal{D}_k)$. Since each party $k \in \{1,\ldots,K\}$ solves its subproblem, there is no need to share its private dataset $\mathcal{D}_k$, but only the intermediate allocations $\boldsymbol{s}_k^{(t)}$, with the other parties. These intermediate allocations are gathered to obtain $\boldsymbol{\lambda}^{(t+1)}$ as in (11). The overall update at iteration t is given by

$$\begin{aligned} \left(\boldsymbol{x}_k^{(t)}, \boldsymbol{s}_k^{(t)}\right)_{k=1,\ldots,K} &= \arg\max_{\boldsymbol{x},\boldsymbol{s}} \mathcal{L}(\boldsymbol{x},\boldsymbol{s},\boldsymbol{\lambda}^{(t)}), \\ \boldsymbol{\lambda}^{(t+1)} &= \boldsymbol{\lambda}^{(t)} - \nu^{(t)} \left(\boldsymbol{c} - \sum_{k=1}^{K} \boldsymbol{s}_k^{(t)} \right), \end{aligned} \tag{12}$$

The overall algorithm may start with the dual feasible solution $\boldsymbol{\lambda} = \mathbf{0}$. (For ease of reference, we give the dual of (1)-(3) in Appendix C.) In this setting, we assume that each party shares the obtained intermediate allocations with other parties truthfully. This is a well-known assumption when parties want to collaborate for their mutual benefit, and hence, act honestly to obtain the correct results (Atallah et al., 2003; Hyndman et al., 2013). Furthermore, in many cases, the collaboration between firms lasts for more than one occasion. Therefore, a firm must consider the long-term prospects of future collaborations that can be jeopardized by opportunistic behavior.

Although the data-hiding approach prevents sharing data, it does not give a formal guarantee of data pri-

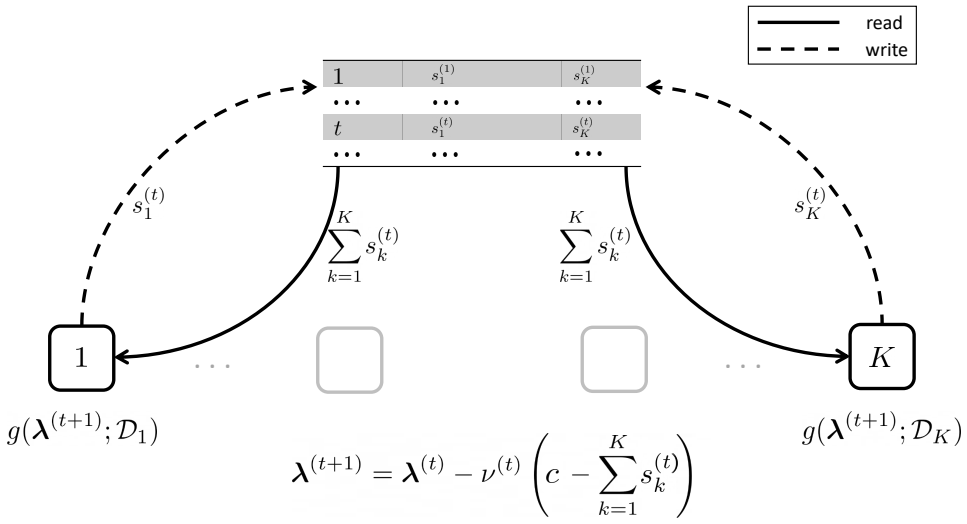

Figure 1: Illustration of the dual decomposition approach to obtain the proposed data-hiding algorithm for multi-party resource sharing.

vacy. The deterministic and iterative nature of the approach could leak information regarding the datasets $\mathcal{D}_1, \ldots, \mathcal{D}_K$. To this end, one can consider applying transformation, encryption, or anonymization techniques for ensuring data privacy. However, these methods either do not guarantee privacy, or they are computationally not feasible for solving even small-scale problems (Sweeney, 1997; Narayanan and Shmatikov, 2006; Toft, 2009). To give a formal data-privacy guarantee, we next adopt a mathematically rigorous notion, called *differential privacy*.

2.2 Differential Privacy. Our next step is to modify the data-hiding approach to obtain a differentially private algorithm. In a nutshell, an algorithm is differentially private when its (public) output after a *small change* is indistinguishable from its output without the change. To cloak the output, a random component is introduced to the algorithm. That is, the randomized algorithm takes an input dataset and returns a random output. Most algorithms achieve randomization by adding noise to their output. The amount of this noise is calibrated according to the sensitivity of the algorithm, which is the maximum change in the output when a single entity is changed in the dataset. The formal statements for distance (change) and sensitivity are due to Dwork et al. (2006b). For ease of reference, we give these statements along with the definition of differential privacy in this section together with our notation to be consistent with the rest of this work.

Although the data-hiding approach of Section 2.1 is guaranteed to obtain the optimal solution without sharing datasets, it is not differentially private since each party observes the deterministic allocations of all the other parties. We should also keep in mind that the optimization algorithms are iterative. Consequently, the output (iterates) are shared by each party not only once but multiple times. Hence, even if we add a random component to the proposed scheme, the privacy of the algorithm degrades with every iteration. This

also implies that there is a hard upper bound on the maximum number of iterations. As we are obliged to stop after a permissible number of iterations, we cannot promise that the summation of individual allocations will not exceed the shared capacities. Therefore, we assume that the parties are aware of this situation, and buffer resources exist to compensate for a possible capacity overflow.

Several privacy mechanisms are useful for ensuring differential privacy of iterative algorithms. Among those results, we mainly use two: Gauss mechanism and composition. In the subsequent part, we first propose a differentially private iterative algorithm. Then, we analyze the trade-off between privacy and near-optimality. We first start with defining adjacent dataset collections and the admissible changes to discuss differential privacy formally.

DEFINITION 2.2 *(Adjacency). Let $\mathscr{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_K\}$ and $\mathscr{D}' = \{\mathcal{D}'_1, \ldots, \mathcal{D}'_{K'}\}$ be two dataset collections. These collections are said to be adjacent if they differ by exactly one party, in the sense that one of the following holds.*

- $\diamond$ *$K' = K$; there exists a $j \in \{1, \ldots, K\}$ such that $\mathcal{D}_j \neq \mathcal{D}'_j$ and $\mathcal{D}_k = \mathcal{D}'_k$ for all $k \neq j$;*
- $\diamond$ *$K' = K + 1$ and $\mathscr{D} \subset \mathscr{D}'$;*
- $\diamond$ *$K' = K - 1$, and $\mathscr{D}' \subset \mathscr{D}$.*

The form of adjacency in Definition 2.2 covers several scenarios. For example, an adjacent collection is obtained when a party in the collaboration is replaced by another party, which corresponds to the first condition. An adjacent collection is also obtained when a party joins or leaves the collaboration, which corresponds to the second and third conditions, respectively.

Definition 2.2 for adjacency is valid for different conceptions for the unit of sensitive information. The two previous scenarios in the above paragraph consider one party's dataset as one unit of sensitive information. Differently from those scenarios, it may as well be the case that each party itself is a data-holder, whose dataset is constituted by the sensitive data of several individuals that belong to that party. We remark that Definition 2.2 for adjacency is still valid in such a scenario: A change in one individual's data in a single party k results in a change in $\mathcal{D}_k$, which in turn results in an adjacent collection as described in the first condition of the definition. Finally, we note that the knowledge of the vector $\boldsymbol{c}$ does not cause any privacy problems as all parties have agreed to share these resources. Now, we give the formal definition of differential privacy.

DEFINITION 2.3 *(Differential privacy (Dwork et al., 2006a)). Let $\epsilon, \delta > 0$. A randomized algorithm f is (ε, δ)-differential private if for all adjacent data collections $\mathscr{D}$ and $\mathscr{D}'$, and all $S \subseteq Range(f)$ it satisfies*

$$\mathbb{P}[f(\mathscr{D}) \in S] \leq e^{\varepsilon}\mathbb{P}[f(\mathscr{D}') \in S] + \delta,$$

where $\mathbb{P}(\cdot)$ is a probability space. The definition promises that a randomized algorithm should output similarly

when two adjacent data sets are considered. Additionally, as seen suggested by the definition, the smaller ε and δ are, the more private the algorithm is. The parameter ε stands for the *privacy loss* or the *privacy budget*. The probability of the privacy loss exceeding ε is bounded by δ. When $\delta = 0$, the algorithm achieves ε-differential privacy, or *pure* differential privacy.

In order to obtain a differentially private variant of the dual decomposition approach, we add a random noise to the returned resource allocations at each iteration, and share the noisy allocations among the parties. That is, the update in (12) is modified as

$$\begin{aligned} \left(\boldsymbol{x}_k^{(t)}, \boldsymbol{s}_k^{(t)}\right)_{k=1,\ldots,K} &= \arg\max_{\boldsymbol{x},\boldsymbol{s}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{\lambda}^{(t)}), \\ \tilde{\boldsymbol{s}}_k^{(t)} &= \boldsymbol{s}_k^{(t)} + \boldsymbol{\omega}_k^{(t)}, \\ \boldsymbol{\lambda}^{(t+1)} &= \boldsymbol{\lambda}^{(t)} - \nu^{(t)} \left(\boldsymbol{c} - \sum_{k=1}^{K} \tilde{\boldsymbol{s}}_k^{(t)}\right), \end{aligned} \tag{13}$$

In (13), $\tilde{\boldsymbol{s}}_k^{(t)}$ is the noisy resource allocation shared among the parties where the random noise vector is denoted by $\boldsymbol{\omega}_k^{(t)}$. As before, the solution $(\boldsymbol{x}_k^{(t)}, \boldsymbol{s}_k^{(t)})$ is obtained by party k solving its subproblem $g(\boldsymbol{\lambda}^{(t)}; \mathcal{D}_k)$.

Next, we discuss how to adjust the variance of the random vectors so that the resulting multi-party resource sharing algorithm satisfies a given (ϵ, δ) differential privacy. There are many mechanisms to consider to determine the noise distribution; in this work we consider the *Gaussian mechanism* where the output of a function of the sensitive data is distorted with a noise drawn from the normal distribution. The variance of the normal distribution should be adjusted according to the sensitivity of the function.

DEFINITION 2.4 *We define the sensitivity of a function* $f : \mathbb{N}^{|\mathscr{D}|} \mapsto \mathbb{R}$ *as*

$$\Delta = \max_{\substack{Adjacent \\ \mathscr{D},\mathscr{D}' \in \mathbb{N}^{|\mathscr{D}|}}} |f(\mathscr{D}) - f(\mathscr{D}')|.$$

Since the privacy-preserving part of (13) is step where $\tilde{\boldsymbol{s}}_k^{(t)}$ is generated, the variance of the components of the noise vector $\boldsymbol{\omega}_k$ need to be adjusted according to the mechanism that produces $\boldsymbol{s}_k^{(t)}$'s as a function of the dataset of party k. Without loss of generality, we assume that each party $k \in \{1, \ldots, K\}$ has an additional set of constraints giving the upper bound on its allotment from the shared capacities. We can denote this bound as the capacity of shared resources, $\boldsymbol{c}$. Each party adds the restrictions $\boldsymbol{s}_k \leq \boldsymbol{c}$ to its set of private constraints, $\{\boldsymbol{x}_k \in \mathbb{R}^{n_k} : \boldsymbol{B}_k \boldsymbol{x}_k \leq \boldsymbol{b}_k\}$. We consider sensitivity for each party k and each shared resource. We can therefore deduce that for each component $j \in \{1, \ldots, m\}$, the sensitivity of the mechanism that produces $s_{k,j}^{(t)}$ is c_j. Furthermore, for each k, the Gaussian mechanism is applied during a total of T iterations for all the m components of $\boldsymbol{s}_k^{(t)}$. The sensitivity and the number of times the Gaussian mechanism is used for each party together determine the distribution of $\boldsymbol{\omega}_k^{(t)}$ to provide (ϵ, δ)- differential privacy. The following

proposition specifies that distribution.

PROPOSITION 2.1 *The multi-party resource sharing algorithm using updates* (13) *for* T *iterations provides* (ε, δ)*-differential privacy if the random vectors are drawn as* $\boldsymbol{\omega}_{k,i}^{(t)} \sim \mathcal{N}(0, \frac{2Tmc_i^2 \log(1/\delta)}{\epsilon^2})$ *for* $i = 1, \ldots, m$ *and* $t = 1, \ldots, T$.

The proof of Proposition 2.1 is given in Appendix B. The noise covariance is derived based on a related definition of privacy, namely zero-concentrated differential privacy (Bun and Steinke, 2016). The reason we consider that particular definition is that the inequality that defines it is tight for the Gaussian mechanism. The proof of Proposition 2.1 exploits the zero-concentrated differential privacy of the Gaussian mechanism, the composition property of the zero-concentrated differential privacy, and finally a conversion from zero-concentrated differential privacy to standard (ϵ, δ)-differential privacy. The required results are already presented in Bun and Steinke (2016); we restate them in Appendix A for completeness.

In Proposition 2.1, we consider capacity-specific sensitivities along the components. Recall that each party k shares its $\tilde{\boldsymbol{s}}_k$, whose nonnegative components $\tilde{s}_{k,1}, \tilde{s}_{k,2}, \ldots, \tilde{s}_{k,m}$ are bounded from above by $c_1, c_2, \ldots, c_m$, respectively. This means that the sensitivity for $\tilde{s}_{k,i}$ is c_i, which leads to the noise variances given in Proposition 2.1. It is also possible to use a common noise variance across the components of $\tilde{\boldsymbol{s}}_k$. In that case, the common variance would be $\frac{2T\|\boldsymbol{c}\|^2 \log(1/\delta)}{\epsilon^2}$ for all $i = 1, \ldots, m$, that is, mc_i^2's would be replaced by $\|\boldsymbol{c}\|_2^2$. In fact, this is not the only alternative choice of noise variances and one can easily optimize this choice, e.g. by minimizing the sum of the variances along the components under the constraint of providing a certain amount of privacy per iteration. The optimization can be done in the same sprit as in Kuru et al. (2020), where a similar noise-distribution problem is addressed. We do not pursue this point further and go with the choice specified in Proposition 2.1.

The privacy guarantee of the modified algorithm comes at the expense of stopping earlier without reaching the optimal solution. Moreover, the new update relation (13) shows that the iterations of the new dual decomposition approach are conducted with noisy subgradients. This stochastic algorithm also affects the convergence behavior of the algorithm. We give an upper bound implying the suboptimality of the differentially private algorithm in the following theorem. For the theorem, we need a typical assumption that is used in the analysis of subgradient methods (Boyd et al., 2003).

ASSUMPTION 2.1 $\|\boldsymbol{\lambda}^{(0)} - \boldsymbol{\lambda}^*\| \leq M$ *for some* $M > 0$.

We define the distance to the optimal objective function value at iteration t as

$$G^{(t)} := \mathcal{L}(\boldsymbol{x}^{(t)}, \boldsymbol{s}^{(t)}, \boldsymbol{\lambda}^{(t)}) - \mathcal{L}(\boldsymbol{x}^*, \boldsymbol{s}^*, \boldsymbol{\lambda}^*), \quad t \geq 0.$$

The following theorem establishes an upper bound for the expectation of $G^{(t)}$.

THEOREM 2.1 *Suppose Assumption 2.1 holds. After executing the (ε, δ)-differentially private multi-party resource sharing algorithm for T iterations using a fixed step-size parameter with the random vectors $\boldsymbol{\omega}_k^{(t)}$, $t = 1, \ldots, T$ drawn from $\mathcal{N}(\mathbf{0}, \frac{Tm\boldsymbol{\Sigma}}{2\rho})$, we obtain the following bound on the optimal objective function value:*

$$\min_{t=0,\ldots,T} \mathbb{E}\left[G^{(t)}\right] \leq M\|\boldsymbol{c}\|\sqrt{\frac{mK}{2\rho} + \frac{(K-1)^2}{T}},$$

where $\boldsymbol{\Sigma} = diag(\boldsymbol{c}\boldsymbol{c}^\intercal)$ and $\rho = \frac{\varepsilon^2}{4\log(1/\delta)}$.

The assumption of the existence of M in the theorem is also a standard assumption that is used in the analysis of subgradient methods (Boyd et al., 2003). The proof of this result is given in Appendix B as well.

To obtain a better bound, one could increase ε or δ, making the algorithm less private. This is a typical trade-off for (ε, δ)-differentially private iterative algorithms, where the variance of the privacy-preserving noise increases with the number of iterations (Ji et al., 2014). Also, independent of the differential privacy parameters, the bound presented in Theorem 2.1 can further be tightened by adding a momentum term to the update steps in (13). Accordingly, new update steps become

$$\boldsymbol{\lambda}^{(t+1)} = \boldsymbol{\lambda}^{(t)} - \nu^{(t)}\left(\boldsymbol{c} - \sum_{k=1}^{K} \tilde{\boldsymbol{s}}_k^{(t)}\right) + \gamma\left(\boldsymbol{\lambda}^{(t)} - \boldsymbol{\lambda}^{(t-1)}\right),$$

where $\tilde{\boldsymbol{s}}_k^{(t)} = \boldsymbol{s}_k^{(t)} + \boldsymbol{\omega}_k^{(t)}$, and the random vectors $\boldsymbol{\omega}_k^{(t)}$ have the distribution $\mathcal{N}(\mathbf{0}, \frac{Tm\boldsymbol{\Sigma}}{2\rho})$. The parameter γ is a non-negative momentum parameter. We note that the privacy of the algorithm is not violated, since the momentum updates use only the information that is already public. Thus, Proposition 2.1 also holds for this scheme. Next, we present a bound on the suboptimality of the differentially private algorithm with momentum updates. The detailed proof is given in Appendix B.

THEOREM 2.2 *Suppose Assumption 2.1 holds. After executing the (ε, δ)-differentially private multi-party resource sharing algorithm for T iterations with momentum updates, we obtain for $0 \leq \nu \leq \frac{\sqrt{G^{(0)2} + (1-\gamma)B^2TM^2} - G^{(0)}}{B^2T}$, the following bound on the optimal objective function value:*

$$\min_{t=0,\ldots,T-1} \mathbb{E}[G^{(t)}] \leq \frac{M^2}{2T\nu} + \frac{\|\boldsymbol{c}\|^2\nu}{2}\left(\frac{TmK}{2\rho} + (K-1)^2\right)$$

2.3 Differential Privacy with Clipping and Adaptive Sensitivity. Clearly, the bounds presented in Theorem 2.1 and Theorem 2.2 become loose when the number of parties participating in the algorithm, K, increases. The main reason to have K in the inequalities is the sensitivity when calibrating the noise for ensuring differential privacy. Recall that any change in the data collections can change the value of $\boldsymbol{s}_k$ from $\mathbf{0}$ to $\boldsymbol{c}$ for each $k \in \{1, \ldots, K\}$, which results in an increased total variance as the number of parties

increases. If we find a way to limit this change, the sensitivity parameter can be lowered. In the following, we propose an alternative update method that decreases the sensitivity, hence reducing the amount of noise in $\tilde{\boldsymbol{s}}_k$'s, without introducing additional privacy loss.

First, we determine a parameter $\alpha \geq 1$ and let $\bar{\boldsymbol{c}} = \alpha \boldsymbol{c}$. For iteration t, we introduce $\bar{\boldsymbol{s}}_k^{(t)}$ with the initialization $\bar{\boldsymbol{s}}_k^{(0)} = \bar{\boldsymbol{c}}/K$. Instead of the update step presented in (13), we have the following update steps.

$$
\begin{aligned}
\left(\boldsymbol{x}_k^{(t)}, \boldsymbol{s}_k^{(t)}\right)_{k=1,\ldots,K} &= \arg\max_{\boldsymbol{x},\boldsymbol{s}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{\lambda}^{(t)}), \\
\tilde{\boldsymbol{s}}_k^{(t)} &= \min\{\bar{\boldsymbol{s}}_k^{(t)}, \boldsymbol{s}_k^{(t)}\} + \mathcal{N}(\boldsymbol{0}, \frac{Tm\boldsymbol{\Sigma}_k^{(t)}}{2\rho}), \quad k = 1, \ldots, K, \\
\boldsymbol{\lambda}^{(t+1)} &= \boldsymbol{\lambda}^{(t)} - \nu^{(t)} \left(\boldsymbol{c} - \sum_{k=1}^{K} \tilde{\boldsymbol{s}}_k^{(t)}\right) + \gamma \left(\boldsymbol{\lambda}^{(t)} - \boldsymbol{\lambda}^{(t-1)}\right), \\
\bar{s}_{k,j}^{(t+1)} &= \bar{c}_j \frac{\max\{\min\{c_j, \tilde{s}_{k,j}^{(t)}\}, \epsilon_{\boldsymbol{c},j}\}}{\sum_{k'=1}^{K} \max\{\min\{c_j, \tilde{s}_{k',j}^{(t)}\}, \epsilon_{\boldsymbol{c},j}\}}, \quad j = 1, \ldots, m; \quad k = 1, \ldots, K
\end{aligned}
\tag{14}
$$

where $\epsilon_{\boldsymbol{c}}$ is a positive vector, and $\boldsymbol{\Sigma}_k^{(t)} = \operatorname{diag}(\bar{\boldsymbol{s}}_k^{(t)}(\bar{\boldsymbol{s}}_k^{(t)})^\intercal)$. In this form of updates, even though the value of $\boldsymbol{s}_k$ can change from $\boldsymbol{0}$ to $\boldsymbol{c}$, we are able to make the sensitivity smaller, since the deterministic part is bounded above by $\bar{\boldsymbol{s}}_k^{(t)}$. One can observe that $\sum_{k=1}^{K} \bar{\boldsymbol{s}}_k^{(t)} = \bar{\boldsymbol{c}}$, which is lower than the previous version of the algorithm for $\alpha < K$. Additionally, we adjust $\bar{\boldsymbol{s}}_k$ values in accordance with other parties such that if a party requires more capacity than the others, then the update steps ensure that the algorithm moves in that direction. For completeness, we show that the modified algorithm that uses clipping to reduce the amount of privacy preserving noise is still (ϵ, δ)-differentially private.

COROLLARY 2.1 *The multi-party resource sharing algorithm using updates (14) for T iterations is (ε, δ)-differentially private, when the random vectors $\boldsymbol{\omega}_k^{(t)}$, $t = 1, \ldots, T$ are drawn from $\mathcal{N}(\boldsymbol{0}, \frac{Tm\boldsymbol{\Sigma}_k^{(t)}}{2\rho})$ where $\boldsymbol{\Sigma}_k^{(t)} = diag(\bar{\boldsymbol{s}}_k^{(t)}(\bar{\boldsymbol{s}}_k^{(t)})^\intercal)$ and $\rho = \frac{\varepsilon^2}{4\log(1/\delta)}$.*

PROOF. The proof follows from Proposition 2.1. □

As a result of the clipping and truncation operations in (14), the approximate step is no longer an unbiased estimate of the subgradient. Therefore, our convergence and optimality bound results above do not immediately apply. However, we have observed the favorable impact of smaller variance due to clipping and truncation in our numerical experiments.

3. Computational Study. In this section, we present an application with the generic model (1)-(3). We simulate a production planning problem with synthetic data to evaluate the efficiency of the proposed methods and discuss the effects of the privacy parameters on the results. We start by explaining our simulation setup in detail and then continue by presenting our numerical results. Our Python implementation along

with the steps to conduct the simulation study are available at our online repository*.

3.1 Setup. We construct an example with multiple parties and five shared capacities ($m = 5$). The components of the shared capacity vector $\boldsymbol{c} \in \mathbb{R}^m$ are randomly sampled from the interval $[10, 20]$. Each party $k \in \{1, \ldots, K\}$ has r_k private capacities, and n_k products to produce. We also introduce demand constraints for each product, and hence, party k has $m_k = n_k + r_k$ private constraints designated by the right-hand-side vector $\boldsymbol{b}_k \in \mathbb{R}^{m_k}$. The parameters r_k and n_k are sampled from the sets $\{5, 6, \ldots, 10\}$ and $\{10, 11, \ldots, 20\}$, respectively. The private capacity limits are randomly sampled from the interval $[10, 20]$. To guarantee obtaining a feasible problem with respect to the demand constraints, we first solve the problem to optimality without those constraints, and then, determine the intervals for randomly sampling the product demands. The components of the technology matrices $\boldsymbol{A}_k \in \mathbb{R}^{m \times n_k}$ and $\boldsymbol{B}_k \in \mathbb{R}^{m_k \times n_k}$ are obtained randomly from the intervals $[0, 5]$ and $[0, 1]$, respectively. The components of the utility vector $\boldsymbol{u}_k \in \mathbb{R}^{n_k}$ of party k are sampled from the interval $[50, 150]$.

In all our experiments for the differentially private setting, we use the constant step-size scheme. We test the differentially private algorithm on a parameter grid for $\varepsilon = 10$ along with $\delta \in \{0.001, 0.01, 0.05, 0.10, 0.15, 0.20\}$ and $K \in \{5, 8, 10, 20\}$. Overall, we obtain 24 different parameter settings. The computational results are reported over 30 simulation runs.

3.2 Results. We start with discussing the data-hiding algorithm and take a look at its performance over iterations. Recall that most of the data exchange requirements are eliminated in this scheme. Parties need to share only their $\boldsymbol{s}_k$ vectors at each iteration. In Figure 2, we report the percentage distances to the optimal objective function values for the first 1,000 iterations. The solid line represents the mean, whereas the shaded area shows the maximum and minimum values. Note that on the vertical axis of the figure, we show the percentage values with respect to the optimal objective function values.

Figure 2 shows that the objective function value obtained with the data-hiding algorithm can be as close as 1.2% to the optimal objective function value. However, this percentage varies significantly as the problem instance changes. On average, when the number of iterations reaches 475, the algorithm already achieves 5% distance from the optimal objective function value. This indicates that if parties use the algorithm steps (11) only up to a certain number of iterations, they can still achieve an acceptable total utility without sharing their private information but their $\boldsymbol{s}_k$ vectors at each iteration.

We also include the momentum updates on the data-hiding algorithm to see its impact on the objective function value. The comparison for the first 100 iterations can be seen in Figure 3. The figure shows that momentum updates improve the convergence rate significantly. On average, the algorithm achieves 5% distance from the optimal objective function value in 39 iterations, which was 475 for the algorithm without momentum updates. This is very appealing since parties can achieve acceptable results within a few iterations

*https://www.github.com/sibirbil/DPMPRS

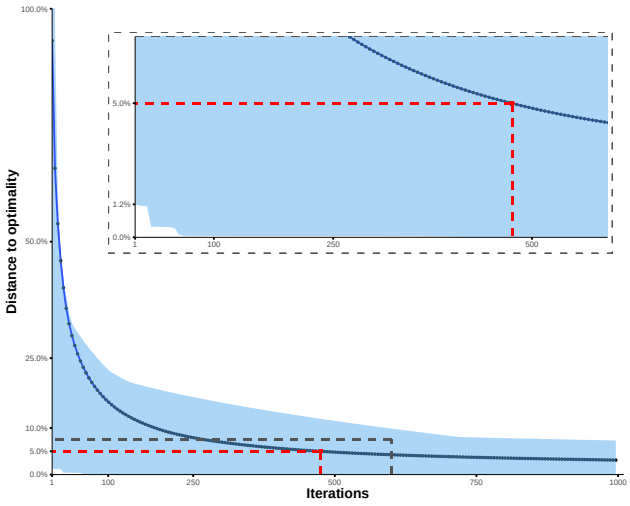

Figure 2: The percentage differences between the objective function values of the overall (non-private) model and the data-hiding model at different iterations. The subfigure shows an enlarged view of the rectangular region marked with the dashed black line on the plot.

without sharing directly any part of their dataset.

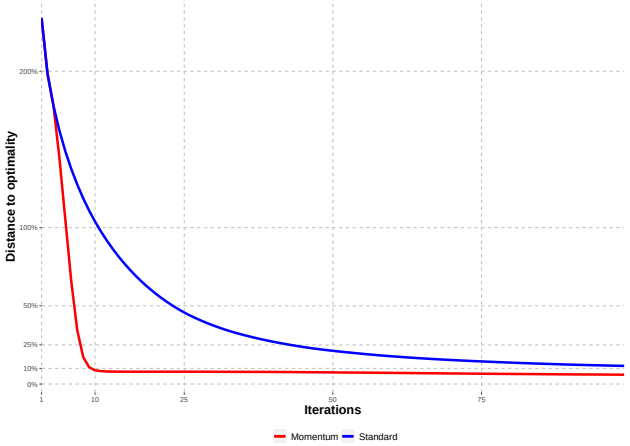

Figure 3: The comparison of data-hiding algorithms (with and without momentum updates) by showing their average percentage differences between the objective function values of the overall (non-private) model and the data-hiding models for the first 100 iterations.

Next, we conduct experiments with the differentially private algorithm and present our results from the point of view of one of the participating parties, say Party 1 ($k = 1$). We analyze the effects of the parameters (K and δ) on the resulting objective function values. We set the maximum number of iterations to $T = 50$ for all scenarios and adjust the other parameters accordingly. The average statistics for objective function values over 30 runs are given in Table 1. Note that we also check the effect of momentum updates when performing differential privacy. As seen in the table, the effect of the momentum updates on the differentially private algorithm is also significant as the number of parties increases. These updates decrease the gap between the objective function value and the differentially private models from 65% to 24%.

Additionally, we see an increase in the gap as the data set becomes larger by including more parties in the data set, *i.e.*, as K increases. This is an expected result since the larger data set results in greater variance in total. Another expected outcome is the effect of the δ parameter. The average percentage gap between the optimal objective function value and the objective function values between differentially private models decreases as the algorithm becomes less private. This is due to the smaller magnitude of variances added on top of $\boldsymbol{s}_k$ values.

Table 1: The average percentage differences between the objective function values of the overall (non-private) models and the differentially private models ($\varepsilon = 10, T = 50$) for varying δ and K values with standard and momentum updates.

	Standard						Momentum					
K \ δ	0.001	0.01	0.05	0.10	0.15	0.20	0.001	0.01	0.05	0.10	0.15	0.20
5	12.06%	13.91%	9.30%	11.25%	9.06%	6.71%	17.45%	15.15%	18.88%	17.48%	15.45%	11.97%
8	16.69%	14.80%	13.27%	11.93%	10.06%	11.30%	15.39%	12.71%	13.36%	12.90%	11.31%	11.51%
10	20.70%	20.07%	20.71%	15.48%	15.47%	14.03%	19.92%	14.55%	15.97%	15.80%	10.59%	11.55%
20	69.07%	65.27%	59.64%	64.78%	49.84%	45.64%	29.55%	43.58%	39.20%	25.37%	23.68%	23.55%

Finally, we present results for the differentially private algorithm with clipping and adaptive sensitivity. The main advantage of this modified algorithm is the lower magnitude of variance added on $\boldsymbol{s}_k$ values. In addition to the altered update steps, we truncate the $\tilde{\boldsymbol{s}}_k$ values. This benefits from the post-processing property of a differentially private mechanism. Before sharing the noisy vector $\tilde{\boldsymbol{s}}_k$, the party k can truncate the noisy vector so that $\tilde{\boldsymbol{s}}_k \in [\epsilon_{\boldsymbol{C}}, \boldsymbol{c}]$. This would not cause any privacy problem since differential privacy is immune to post-processing (Dwork et al., 2014). We again analyze the effects of the privacy parameters (K and δ) on the resulting objective function values. We set the maximum number of iterations to $T = 50$ for all scenarios and adjust the other parameters accordingly. The results are presented in Table 2.

Table 2: The average percentage differences between the objective function values of the overall (non-private) models and the differentially private models ($\varepsilon = 10, T = 50$) with clipping for varying δ and K values with standard and momentum updates.

	Standard						Momentum					
K \ δ	0.001	0.01	0.05	0.10	0.15	0.20	0.001	0.01	0.05	0.10	0.15	0.20
5	50.01%	62.78%	64.21%	67.27%	72.57%	71.41%	11.61%	12.14%	8.53%	14.75%	13.38%	16.38%
8	6.09%	5.36%	5.55%	6.39%	5.79%	7.64%	6.93%	6.39%	6.89%	6.49%	7.22%	6.96%
10	4.67%	5.38%	6.08%	5.16%	5.66%	5.96%	7.38%	5.87%	7.03%	6.89%	6.53%	6.49%
20	8.79%	8.54%	8.20%	8.80%	6.99%	6.48%	7.92%	8.41%	9.15%	8.84%	8.64%	8.43%

The table shows that the modified algorithm does not significantly worsen as the number of parties increases. This is, in fact, an expected result since the larger the dataset, the less probable that a party can be observable by the algorithm's outputs. The algorithm performs worse on average when $K = 5$, and unlike the previous results, we see the benefit of including momentum updates for the same scenarios.

4. Conclusion. We have formulated a multi-party resource-sharing model, where the primary concern is to ensure the data privacy of the involved parties. We have started with a relaxation of the original model to obtain a decomposition that minimizes the necessity of sharing individual input data. This relaxation

approach has led to a data-hiding scheme. Then, we have modified this scheme to benefit from the mathematically well-defined notion of differential privacy. This modification has allowed us to give a privacy guarantee and obtain our differentially private resource sharing algorithm. As differential privacy guarantee comes inevitably at the loss of optimality, we have also given bounds on this loss for different privacy parameters of the proposed algorithm. We have additionally proposed a modification to the original algorithm to increase the efficiency when the number of parties increases. We have also conducted a computational study on a planning problem with synthetic data to support our analysis. We have reported our results for various parameters of privacy and given a discussion of the effects of privacy level on optimality. These results for the original algorithm have also shown that when the number of parties decreases, the optimality gaps also diminish. Thanks to the modification we have proposed, the optimality gap does not increase significantly with the number of parties.

We believe our current work has the potential to open up a new path for future research. We have reserved this work for a linear programming formulation because linear programs are used frequently in various applications. As we have noted, our approach here can be easily extended to a general convex programming setting. In that case, one needs to give a clear discussion about what constitutes private and non-private datasets. Combined with an application, this could be an interesting research topic. Any iterative differentially private algorithm loses privacy as the number of iterations increases. Therefore, our differentially private algorithm also stops after a fixed number of iterations. This early termination affects not only the optimality but also the feasibility of the algorithm. In other words, the resulting set of allocations from our algorithm may exceed some of the shared capacities. We are not aware of another approach that would lead to a differentially private algorithm that also guarantees feasibility. Investigating this point further is certainly on our agenda for future research.

Appendix A. Further Definitions for Differential Privacy. Here we provide a definition of, and some relevant results about, zero concentrated differential privacy and state its relation to the standard version of differential privacy.

DEFINITION A.1 *(Zero-Concentrated Differential Privacy (zCDP); Bun and Steinke (2016)). A randomized mechanism $\mathcal{M}$ is (ξ, ρ)-zCDP if for all adjacent data collections $\mathscr{D}$ and $\mathscr{D}'$ and all $\alpha \in (1, \infty)$,*

$$D_\alpha(\mathcal{M}(\mathscr{D})||\mathcal{M}(\mathscr{D}')) \leq \xi + \rho\alpha,$$

where $D_\alpha(\mathcal{M}(\mathscr{D})||\mathcal{M}(\mathscr{D}'))$ is the α-Rényi divergence between the distribution of $\mathcal{M}(\mathscr{D})$ and the distribution of $\mathcal{M}(\mathscr{D}')$.

The next result gives the relationship between differential privacy and zero concentrated differential privacy that is due to Bun and Steinke (2016).

THEOREM A.1 *(Approximate DP and zCDP; Bun and Steinke (2016)). Let a randomized mechanism $\mathcal{M} : \mathcal{D} \to \mathbb{R}$ satisfies (ξ, ρ)-zCDP. Then, $\mathcal{M}$ satisfies (ε, δ) differential privacy for all $\delta > 0$ and $\varepsilon = \xi + \rho + \sqrt{4\rho \log(1/\delta)}$.*

Gauss mechanism is one of the most used tools in the literature for ensuring differential privacy. The mechanism adds noise drawn from the Normal distribution with variance σ^2 to the output of the function.

THEOREM A.2 *(Gauss Mechanism, (Bun and Steinke, 2016, Proposition 6)). For a randomized mechanism $\mathcal{M}$ that adds noises to the output of a function f, sampled from Normal(0, σ^2) enjoys $(\Delta^2/2\sigma^2)$-zCDP where* $\Delta = \max_{\substack{Adjacent \\ \mathscr{D}, \mathscr{D}' \in \mathbb{N}^{|\mathscr{D}|}}} \|f(\mathscr{D}) - f(\mathscr{D}')\|_2$.

The following composition theorem quantifies the total privacy loss when multiple calculations of the sensitive data are released.

THEOREM A.3 *(Composition; Lemma 7, Bun and Steinke (2016)). Let $\mathcal{M}_i : \mathbb{N}^{|\mathscr{D}|} \to \mathbb{R}_i$, $i = 1, \ldots, m$ be a collection of (ρ_i)- zero concentrated differentially private mechanisms, respectively. Then, the mechanism $\mathcal{M}(\mathscr{D}) := (\mathcal{M}_1(\mathscr{D}), \ldots, \mathcal{M}_m(\mathscr{D}))$ is $\sum_{i=1}^m \rho_i$-zCDP. Moreover, the same privacy guarantee holds even the mechanism $\mathcal{M}_j$ uses the outputs of its predecessors, $\mathcal{M}_1, \ldots, \mathcal{M}_{j-1}$ for all $j = 1, \ldots, m$.*

Appendix B. Omitted Proofs. We reserve this section for the omitted proofs in the main text.

PROOF OF PROPOSITION 2.1. To obtain (ε, δ) differentially private algorithm, we use Theorems A.1 and A.3. Basically, we need to have a $\frac{\rho}{Tm}$-zCDP mechanism, since we are using the modified version of the algorithm, which lets party k share its noisy vector $\tilde{\boldsymbol{s}}_k$ one component at a time. Thus, we achieve

(ε, δ) differential privacy where $\rho = \frac{(\varepsilon-\xi)^2}{4\log(1/\delta)}$. We can guarantee $\frac{\rho}{Tm}$-zCDP by adding a noise drawn from $N(0, \frac{Tm\boldsymbol{\Sigma}}{2\rho})$.

By Theorem A.2, a mechanism that outputs $N(\boldsymbol{s}_k, \frac{Tm\boldsymbol{\Sigma}}{2\rho})$ achieves $\frac{\rho}{Tm}$-zero concentrated differential privacy. Using the composition feature of zCDP mentioned in Theorem A.3, we conclude that over T iterations and m shared resources, the algorithm is ρ-zCDP. Lastly, we make use of Theorem A.1, which completes the proof. □

Next, we prove the theorems on the upper bounds of the optimality gap for the proposed methods. Before proceeding, we let $G^{(t)} = \mathcal{L}(\boldsymbol{x}^{(t)}, \boldsymbol{s}^{(t)}, \boldsymbol{\lambda}^{(t)}) - \mathcal{L}(\boldsymbol{x}^*, \boldsymbol{s}^*, \boldsymbol{\lambda}^*)$ for $t \geq 0$ and $G^{(t_1,t_2)} = G^{(t_1)} - G^{(t_2)} = \mathcal{L}(\boldsymbol{x}^{(t_1)}, \boldsymbol{s}^{(t_1)}, \boldsymbol{\lambda}^{(t_1)}) - \mathcal{L}(\boldsymbol{x}^{(t_2)}, \boldsymbol{s}^{(t_2)}, \boldsymbol{\lambda}^{(t_2)})$ for $t_2 \geq t_1 \geq 0$.

Proof of Theorem 2.1. Recall the update steps (13) and $\tilde{\boldsymbol{s}}_k^{(t)} = \boldsymbol{s}_k^{(t)} + \boldsymbol{\omega}_k^{(t)}$, where $\boldsymbol{\omega}_k^{(t)} \sim N(\boldsymbol{0}, \frac{Tm\boldsymbol{\Sigma}}{2\rho})$ with $\boldsymbol{\Sigma} = \operatorname{diag}(\boldsymbol{c}\boldsymbol{c}^\intercal)$. Let $\tilde{\boldsymbol{g}}^{(t)} = \boldsymbol{c} - \sum_{k=1}^K \tilde{\boldsymbol{s}}_k^{(t)}$ denote the approximate subgradient. Then, we have

$$\mathbb{E}[\|\tilde{\boldsymbol{g}}^{(t)}\|^2] \leq \frac{TmK\|\boldsymbol{c}\|^2}{2\rho} + \|(K-1)\boldsymbol{c}\|^2 = \left(\frac{TmK}{2\rho} + (K-1)^2\right)\|\boldsymbol{c}\|^2. \tag{15}$$

We also obtain

$$\begin{aligned}\mathbb{E}[\|\boldsymbol{\lambda}^{(t+1)} - \boldsymbol{\lambda}^*\|^2|\boldsymbol{\lambda}^{(t)}] &= \mathbb{E}[\|\boldsymbol{\lambda}^{(t)} - \boldsymbol{\lambda}^* - \nu^{(t)}\tilde{\boldsymbol{g}}^{(t)}\|^2|\boldsymbol{\lambda}^{(t)}] \\ &= \|\boldsymbol{\lambda}^{(t)} - \boldsymbol{\lambda}^*\|^2 - 2\nu^{(t)}\mathbb{E}[\tilde{\boldsymbol{g}}^{(t)}|\boldsymbol{\lambda}^{(t)}]^\intercal(\boldsymbol{\lambda}^{(t)} - \boldsymbol{\lambda}^*) + (\nu^{(t)})^2\mathbb{E}[\|\tilde{\boldsymbol{g}}^{(t)}\|^2|\boldsymbol{\lambda}^{(t)}].\end{aligned}$$

Using the definition of subgradient, we have

$$\mathcal{L}(\boldsymbol{x}^*, \boldsymbol{s}^*, \boldsymbol{\lambda}^*) \geq \mathbb{E}[\tilde{\boldsymbol{g}}^{(t)}|\boldsymbol{\lambda}^{(t)}]^\intercal(\boldsymbol{\lambda}^* - \boldsymbol{\lambda}^{(t)}) + \mathcal{L}(\boldsymbol{x}^{(t)}, \boldsymbol{s}^{(t)}, \boldsymbol{\lambda}^{(t)}),$$

which yields

$$\mathbb{E}[\|\boldsymbol{\lambda}^{(t+1)} - \boldsymbol{\lambda}^*\|^2|\boldsymbol{\lambda}^{(t)}] \leq \|\boldsymbol{\lambda}^{(t)} - \boldsymbol{\lambda}^*\|^2 - 2\nu^{(t)}G^{(t)} + (\nu^{(t)})^2\mathbb{E}[\|\tilde{\boldsymbol{g}}^{(t)}\|^2|\boldsymbol{\lambda}^{(t)}]. \tag{16}$$

Then, taking the expectation of both sides of (16) leads to

$$2\nu^{(t)}\mathbb{E}[G^{(t)}] \leq \mathbb{E}\|\boldsymbol{\lambda}^{(t)} - \boldsymbol{\lambda}^*\|^2 - \mathbb{E}\|\boldsymbol{\lambda}^{(t+1)} - \boldsymbol{\lambda}^*\|^2 + (\nu^{(t)})^2\mathbb{E}[\|\tilde{\boldsymbol{g}}^{(t)}\|^2].$$

Let $B^2 = \left(\frac{TmK}{2\rho} + (K-1)^2\right)\|\boldsymbol{c}\|^2$, which clearly satisfies $\mathbb{E}[\|\tilde{\boldsymbol{g}}^{(t)}\|^2] \leq B^2$ by (15). Using B^2 for an upper

bound in (B), and taking the summation of both sides, we obtain

$$2\sum_{t=0}^{T-1}\nu^{(t)}\mathbb{E}[G^{(t)}] \leq \mathbb{E}\|\boldsymbol{\lambda}^{(0)}-\boldsymbol{\lambda}^*\|^2-\mathbb{E}\|\boldsymbol{\lambda}^{(T)}-\boldsymbol{\lambda}^*\|^2+B^2\sum_{t=0}^{T-1}(\nu^{(t)})^2 \leq \mathbb{E}\|\boldsymbol{\lambda}^{(0)}-\boldsymbol{\lambda}^*\|^2+B^2\sum_{t=0}^{T-1}(\nu^{(t)})^2.$$

This inequality also implies

$$\min_{t=0,\ldots,T-1}\mathbb{E}[G^{(t)}] \leq \frac{\mathbb{E}\|\boldsymbol{\lambda}^{(0)}-\boldsymbol{\lambda}^*\|^2+B^2\sum_{t=0}^{T-1}(\nu^{(t)})^2}{2\sum_{t=0}^{T-1}\nu^{(t)}}. \tag{17}$$

Recall that our assumption satisfies $\|\boldsymbol{\lambda}^{(0)}-\boldsymbol{\lambda}^*\|^2 \leq M^2$. With the constant step-size $\nu^{(t)}=\frac{M}{B\sqrt{T}}$, inequality (17) becomes

$$\min_{t=0,\ldots,T-1}\mathbb{E}[G^{(t)}] \leq \frac{M^2+B^2\sum_{t=0}^{T-1}(\nu^{(t)})^2}{2\sum_{t=0}^{T-1}\nu^{(t)}} \leq \frac{MB}{\sqrt{T}} = M\|\boldsymbol{c}\|\sqrt{\frac{mK}{2\rho}+\frac{(K-1)^2}{T}}.$$

This shows the desired result. □

Proof of Theorem 2.2. Recall the update steps

$$\boldsymbol{\lambda}^{(t+1)} = \boldsymbol{\lambda}^{(t)} - \nu^{(t)}\left(\boldsymbol{c}-\sum_{k=1}^{K}\tilde{\boldsymbol{s}}_k^{(t)}\right)+\gamma\left(\boldsymbol{\lambda}^{(t)}-\boldsymbol{\lambda}^{(t-1)}\right),$$

where $\tilde{\boldsymbol{s}}_k^{(t)} = \boldsymbol{s}_k^{(t)}+\boldsymbol{\omega}_k^{(t)}$, and the components of $\boldsymbol{\omega}_k^{(t)}$ have the distribution where the components of $\boldsymbol{\omega}_k^{(t)}$ have the distribution $N(\mathbf{0},\frac{Tm\boldsymbol{\Sigma}}{2\rho})$ for each party k with $\boldsymbol{\Sigma}=\mathrm{diag}(\boldsymbol{c}\boldsymbol{c}^\intercal)$. Let $\tilde{\boldsymbol{g}}^{(t)}=\boldsymbol{c}-\sum_{k=1}^{K}\tilde{\boldsymbol{s}}_k^{(t)}$ denote the approximate subgradient. Then, we have

$$\mathbb{E}[\|\tilde{\boldsymbol{g}}^{(t)}\|^2] \leq \frac{TmK\|\boldsymbol{c}\|^2}{2\rho}+\|(K-1)\boldsymbol{c}\|^2 = \left(\frac{TmK}{2\rho}+(K-1)^2\right)\|\boldsymbol{c}\|^2. \tag{18}$$

We use an equivalent representation of the update step (Ghadimi et al., 2015)

$$\boldsymbol{\lambda}^{(t+1)}+\boldsymbol{p}^{(t+1)} = \boldsymbol{\lambda}^{(t)}+\boldsymbol{p}^{(t)}-\frac{\nu^{(t)}}{1-\gamma}\tilde{\boldsymbol{g}}^{(t)},$$

where

$$\boldsymbol{p}^{(t)} = \begin{cases}\frac{\gamma}{1-\gamma}\left(\boldsymbol{\lambda}^{(t)}-\boldsymbol{\lambda}^{(t-1)}\right), & t\geq 1;\\ 0, & t=0.\end{cases}$$

Then, we proceed as

$$\begin{aligned}
\mathbb{E}\left[\|\boldsymbol{\lambda}^{(t+1)}+\boldsymbol{p}^{(t+1)}-\boldsymbol{\lambda}^*\|^2|\boldsymbol{\lambda}^{(t)}\right] &= \mathbb{E}\left[\|\boldsymbol{\lambda}^{(t)}+\boldsymbol{p}^{(t)}-\boldsymbol{\lambda}^*-\frac{\nu^{(t)}}{1-\gamma}\tilde{\boldsymbol{g}}^{(t)}\|^2|\boldsymbol{\lambda}^{(t)}\right] \\
&= \|\boldsymbol{\lambda}^{(t)}+\boldsymbol{p}^{(t)}-\boldsymbol{\lambda}^*\|^2-\mathbb{E}\left[\frac{2\nu^{(t)}}{1-\gamma}(\boldsymbol{\lambda}^{(t)}+\boldsymbol{p}^{(t)}-\boldsymbol{\lambda}^*)^\intercal\tilde{\boldsymbol{g}}^{(t)}\right] \\
&\quad +\mathbb{E}\left[\left(\frac{\nu^{(t)}}{1-\gamma}\right)^2\|\tilde{\boldsymbol{g}}^{(t)}\|^2|\boldsymbol{\lambda}^{(t)}\right] \\
&= \|\boldsymbol{\lambda}^{(t)}+\boldsymbol{p}^{(t)}-\boldsymbol{\lambda}^*\|^2-\mathbb{E}\left[\frac{2\nu^{(t)}}{1-\gamma}(\boldsymbol{\lambda}^{(t)}-\boldsymbol{\lambda}^*)^\intercal\tilde{\boldsymbol{g}}^{(t)}\right] \\
&\quad -\mathbb{E}\left[\frac{2\nu^{(t)}}{1-\gamma}\left(\frac{\gamma}{1-\gamma}(\boldsymbol{\lambda}^{(t)}-\boldsymbol{\lambda}^{(t-1)})\right)^\intercal\tilde{\boldsymbol{g}}^{(t)}\right] \\
&\quad +\mathbb{E}\left[\left(\frac{\nu^{(t)}}{1-\gamma}\right)^2\|\tilde{\boldsymbol{g}}^{(t)}\|^2|\boldsymbol{\lambda}^{(t)}\right] \\
&= \|\boldsymbol{\lambda}^{(t)}+\boldsymbol{p}^{(t)}-\boldsymbol{\lambda}^*\|^2+\mathbb{E}\Big[-\frac{2\nu^{(t)}}{1-\gamma}(\boldsymbol{\lambda}^{(t)}-\boldsymbol{\lambda}^*)^\intercal\tilde{\boldsymbol{g}}^{(t)} \\
&\quad -\frac{2\nu^{(t)}\gamma}{(1-\gamma)^2}\left(\boldsymbol{\lambda}^{(t)}-\boldsymbol{\lambda}^{(t-1)}\right)^\intercal\tilde{\boldsymbol{g}}^{(t)}+\left(\frac{\nu^{(t)}}{1-\gamma}\right)^2\|\tilde{\boldsymbol{g}}^{(t)}\|^2|\boldsymbol{\lambda}^{(t)}\Big].
\end{aligned}$$

Using the definition of subgradient, we obtain

$$\mathcal{L}(\boldsymbol{x}^*,\boldsymbol{s}^*,\boldsymbol{\lambda}^*)\geq\mathbb{E}[\tilde{\boldsymbol{g}}^{(t)}|\boldsymbol{\lambda}^{(t)}]^\intercal(\boldsymbol{\lambda}^*-\boldsymbol{\lambda}^{(t)})+\mathcal{L}(\boldsymbol{x}^{(t)},\boldsymbol{s}^{(t)},\boldsymbol{\lambda}^{(t)}),$$

which yields

$$\begin{aligned}
\mathbb{E}\left[\|\boldsymbol{\lambda}^{(t+1)}+\boldsymbol{p}^{(t+1)}-\boldsymbol{\lambda}^*\|^2|\boldsymbol{\lambda}^{(t)}\right] &\leq \|\boldsymbol{\lambda}^{(t)}+\boldsymbol{p}^{(t)}-\boldsymbol{\lambda}^*\|^2-\frac{2\nu^{(t)}}{1-\gamma}G^{(t)} \\
&\quad +\mathbb{E}\Big[-\frac{2\nu^{(t)}\gamma}{(1-\gamma)^2}\left(\boldsymbol{\lambda}^{(t)}-\boldsymbol{\lambda}^{(t-1)}\right)^\intercal\tilde{\boldsymbol{g}}^{(t)}+\left(\frac{\nu^{(t)}}{1-\gamma}\right)^2\|\tilde{\boldsymbol{g}}^{(t)}\|^2|\boldsymbol{\lambda}^{(t)}\Big].
\end{aligned}$$

Further, from the definition of subgradient, we have

$$\mathcal{L}(\boldsymbol{x}^{(t-1)},\boldsymbol{s}^{(t-1)},\boldsymbol{\lambda}^{(t-1)})\geq\mathbb{E}[\tilde{\boldsymbol{g}}^{(t)}|\boldsymbol{\lambda}^{(t)}]^\intercal(\boldsymbol{\lambda}^{(t-1)}-\boldsymbol{\lambda}^{(t)})+\mathcal{L}(\boldsymbol{x}^{(t)},\boldsymbol{s}^{(t)},\boldsymbol{\lambda}^{(t)}).$$

We end up with

$$\begin{aligned}
\mathbb{E}\left[\|\boldsymbol{\lambda}^{(t+1)}+\boldsymbol{p}^{(t+1)}-\boldsymbol{\lambda}^*\|^2|\boldsymbol{\lambda}^{(t)}\right] &\leq \|\boldsymbol{\lambda}^{(t)}+\boldsymbol{p}^{(t)}-\boldsymbol{\lambda}^*\|^2-\frac{2\nu^{(t)}}{1-\gamma}G^{(t)}+\frac{2\nu^{(t)}\gamma}{(1-\gamma)^2}G^{(t-1,t)} \\
&\quad +\left(\frac{\nu^{(t)}}{1-\gamma}\right)^2\mathbb{E}\left[\|\tilde{\boldsymbol{g}}^{(t)}\|^2|\boldsymbol{\lambda}^{(t)}\right].
\end{aligned}$$

This implies

$$\mathbb{E}\left[\|\boldsymbol{\lambda}^{(t+1)}+\boldsymbol{p}^{(t+1)}-\boldsymbol{\lambda}^*\|^2|\boldsymbol{\lambda}^{(t)}\right] \leq \|\boldsymbol{\lambda}^{(t)}+\boldsymbol{p}^{(t)}-\boldsymbol{\lambda}^*\|^2-\frac{2\nu^{(t)}}{1-\gamma}G^{(t)}+\frac{2\nu^{(t)}\gamma}{(1-\gamma)^2}G^{(t-1,t)} + \left(\frac{\nu^{(t)}}{1-\gamma}\right)^2\mathbb{E}\left[\|\tilde{\boldsymbol{g}}^{(t)}\|^2|\boldsymbol{\lambda}^{(t)}\right].$$

We then take the expectation of both sides

$$\frac{2\nu^{(t)}}{1-\gamma}\mathbb{E}\left[G^{(t)}\right]-\frac{2\nu^{(t)}\gamma}{(1-\gamma)^2}\mathbb{E}\left[G^{(t-1,t)}\right] \leq \mathbb{E}\left[\|\boldsymbol{\lambda}^{(t)}+\boldsymbol{p}^{(t)}-\boldsymbol{\lambda}^*\|^2\right]-\mathbb{E}\left[\|\boldsymbol{\lambda}^{(t+1)}+\boldsymbol{p}^{(t+1)}-\boldsymbol{\lambda}^*\|^2\right] + \left(\frac{\nu^{(t)}}{1-\gamma}\right)^2\mathbb{E}\left[\|\tilde{\boldsymbol{g}}^{(t)}\|^2\right].$$

Let $B^2=\left(\frac{TmK}{2\rho}+(K-1)^2\right)\|\boldsymbol{c}\|^2$. If we take the step-size parameter fixed, $\nu^{(t)}=\nu$, and sum over iterations $t=0,\ldots,T-1$, we obtain

$$\begin{aligned}\frac{2\nu}{1-\gamma}\sum_{t=0}^{T-1}\mathbb{E}\left[G^{(t)}\right]-\frac{2\nu\gamma}{(1-\gamma)^2}\mathbb{E}\left[G^{(0,T-1)}\right] &\leq \mathbb{E}\left[\|\boldsymbol{\lambda}^{(0)}-\boldsymbol{\lambda}^*\|^2\right]-\mathbb{E}\left[\|\boldsymbol{\lambda}^{(T)}+\boldsymbol{p}^{(T)}-\boldsymbol{\lambda}^*\|^2\right]+\left(\frac{B}{1-\gamma}\right)^2T\nu^2\\ &\leq \mathbb{E}\left[\|\boldsymbol{\lambda}^{(0)}-\boldsymbol{\lambda}^*\|^2\right]+T\left(\frac{B\nu}{1-\gamma}\right)^2,\end{aligned}$$

since $\mathbb{E}[\|\tilde{\boldsymbol{g}}^{(t)}\|^2]\leq B^2$ by (18). This is equivalent to the following

$$2\nu\sum_{t=0}^{T-1}\mathbb{E}\left[G^{(t)}\right]\leq(1-\gamma)\mathbb{E}\left[\|\boldsymbol{\lambda}^{(0)}-\boldsymbol{\lambda}^*\|^2\right]+\frac{2\nu\gamma}{(1-\gamma)}\mathbb{E}\left[G^{(0,T-1)}\right]+\frac{T(\nu B)^2}{1-\gamma}.$$

This inequality also implies

$$\min_{t=0,\ldots,T-1}\mathbb{E}[G^{(t)}]\leq\frac{(1-\gamma)\mathbb{E}\|\boldsymbol{\lambda}^{(0)}-\boldsymbol{\lambda}^*\|^2+\frac{2\nu\gamma}{(1-\gamma)}\mathbb{E}\left[G^{(0,T-1)}\right]+\frac{T(\nu B)^2}{1-\gamma}}{2T\nu}.$$

Recall the assumption $\|\boldsymbol{\lambda}^{(0)}-\boldsymbol{\lambda}^*\|^2\leq M^2$, which implies that $G^{(0)}<\inf$. This further implies that $\mathbb{E}[G^{(0,t)}]\leq G^{(0)}$. Therefore,

$$\min_{t=0,\ldots,T-1}\mathbb{E}[G^{(t)}]\leq\frac{(1-\gamma)M^2+\frac{2\nu\gamma}{1-\gamma}G^{(0)}+\frac{T(\nu B)^2}{1-\gamma}}{2T\nu}.$$

To obtain a tighter bound, we need to satisfy the following inequality

$$\frac{(1-\gamma)M^2+\frac{2\nu\gamma}{1-\gamma}G^{(0)}+\frac{T(\nu B)^2}{1-\gamma}}{2T\nu}\leq\frac{M^2+B^2T\nu^2}{2T\nu}.$$

Thus, if we select ν as

$$0 \leq \nu \leq \frac{\sqrt{G^{(0)2} + (1-\gamma)B^2TM^2} - G^{(0)}}{B^2T},$$

then we guarantee to have a tighter bound. □

Appendix C. Let $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_K$ and $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_K$ be the dual vectors corresponding to the constraints (5) and (6), respectively. Further, we define $\boldsymbol{\lambda}$ as the dual vector for Constraint (7). Then, the dual formulation of (4)-(8) becomes

$$\begin{aligned}
\text{minimize} \quad & \boldsymbol{c}^\intercal \boldsymbol{\lambda} + \sum_{k=1}^{K} \boldsymbol{b}_k^\intercal \boldsymbol{\beta}_k \\
\text{subject to} \quad & \boldsymbol{A}_k^\intercal \boldsymbol{\alpha}_k + \boldsymbol{B}_k^\intercal \boldsymbol{\beta}_k \geq \boldsymbol{u}_k, && k = 1, \ldots, K, \\
& \boldsymbol{\lambda} - \boldsymbol{\alpha}_k \geq \boldsymbol{0}, && k = 1, \ldots, K, \\
& \boldsymbol{\alpha}_k, \boldsymbol{\beta}_k \geq \boldsymbol{0}, && k = 1, \ldots, K, \\
& \boldsymbol{\lambda} \text{ unrestricted.}
\end{aligned}$$

References

Agarwal, R. and Ergun, Ö. (2010). Network design and allocation mechanisms for carrier alliances in liner shipping. *Operations Research*, 58(6):1726–1742.

Albrecht, M. and Stadtler, H. (2015). Coordinating decentralized linear programs by exchange of primal information. *European Journal of Operational Research*, 247(3):788–796.

Amaruchkul, K., Cooper, W. L., and Gupta, D. (2011). A note on air-cargo capacity contracts. *Production and Operations Management*, 20(1):152–162.

Atallah, M., Elmongui, H., Deshpande, V., and Schwarz, L. (2003). Secure supply-chain protocols. In *EEE International Conference on E-Commerce, 2003*, pages 293–302.

Bednarz, A., Bean, N., and Roughan, M. (2009). Hiccups on the road to privacy-preserving linear programming. In *Proceedings of the 8th ACM Workshop on Privacy in the Electronic Society*, pages 117–120.

Bertsekas, D. P. (2015). *Convex Optimization Algorithms*. Athena Scientific, Belmont.

Birbil, Ş. İ., Frenk, J., Gromicho, J. A., and Zhang, S. (2014). A network airline revenue management framework based on decomposition by origins and destinations. *Transportation Science*, 48(3):313–333.

Boyd, S., Xiao, L., and Mutapcic, A. (2003). Subgradient methods. *Lecture Notes of EE392o, Stanford University, Autumn Quarter*. http://web.mit.edu/6.976/www/notes/subgrad_method.pdf (accessed on 13.02.2021).

Bun, M. and Steinke, T. (2016). Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography Conference*, pages 635–658. Springer.

Chun, S. Y., Kleywegt, A. J., and Shapiro, A. (2016). When friends become competitors: the design of resource exchange alliances. *Management Science*, 63(7):2127–2145.

Ding, S. and Kaminsky, P. M. (2020). Centralized and decentralized warehouse logistics collaboration. *Manufacturing & Service Operations Management*, 22(4):812–831.

Dreier, J. and Kerschbaum, F. (2011). Practical privacy-preserving multiparty linear programming based on problem transformation. In *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*, pages 916–924.

Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., and Naor, M. (2006a). Our data, ourselves: Privacy via distributed noise generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 486–503. Springer.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. (2006b). Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference*, pages 265–284. Springer.

Dwork, C., Roth, A., et al. (2014). *The Algorithmic Foundations of Differential Privacy*. Number 3–4. Now Publishers, Inc., Boston.

Geyer, R. C., Klein, T., and Nabi, M. (2017). Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*. https://arxiv.org/abs/1712.07557 (accessed on 01.04.2021).

Ghadimi, E., Feyzmahdavian, H. R., and Johansson, M. (2015). Global convergence of the heavy-ball method for convex optimization. In *2015 European Control Conference (ECC)*, pages 310–315. IEEE.

Guo, L. and Wu, X. (2018). Capacity sharing between competitors. *Management Science*, 64(8):3554–3573.

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

Han, S., Topcu, U., and Pappas, G. J. (2016). Differentially private distributed constrained optimization. *IEEE Transactions on Automatic Control*, 62(1):50–64.

Hong, Y. and Vaidya, J. (2014). An inference–proof approach to privacy-preserving horizontally partitioned linear programs. *Optimization Letters*, 8(1):267–277.

Hsu, J., Huang, Z., Roth, A., and Wu, Z. S. (2016). Jointly private convex programming. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 580–599. SIAM.

Hsu, J., Roth, A., Roughgarden, T., and Ullman, J. (2014). Privately solving linear programs. In *International Colloquium on Automata, Languages, and Programming*, pages 612–624. Springer.

Huang, Z., Mitra, S., and Vaidya, N. (2015). Differentially private distributed optimization. In *Proceedings of the 2015 International Conference on Distributed Computing and Networking*, page 4. ACM.

Hyndman, K., Kraiselburd, S., and Watson, N. (2013). Aligning capacity decisions in supply chains when demand forecasts are private information: Theory and experiment. *Manufacturing & Service Operations Management*, 15(1):102–117.

Ibaraki, T. and Katoh, N. (1988). *Resource Allocation Problems: Algorithmic Approaches*. MIT Press, Cambridge, MA, USA.

Ji, Z., Lipton, Z. C., and Elkan, C. (2014). Differential privacy and machine learning: a survey and review. *arXiv preprint*. https://arxiv.org/abs/1412.7584 (accessed on 13.12.2020).

Karaca, U., Birbil, Ş. İ., Aydın, N., and Sağol, G. (2022). Masking primal and dual models for data privacy in network revenue management. *arXiv preprint*. https://arxiv.org/abs/2102.07178 (accessed on 22.02.2022).

Konečnỳ, J., McMahan, H. B., Ramage, D., and Richtárik, P. (2016). Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint*. https://arxiv.org/abs/1610.02527 (accessed on 16.05.2021).

Kovács, A., Egri, P., Kis, T., and Váncza, J. (2013). Inventory control in supply chains: Alternative approaches to a two-stage lot-sizing problem. *International Journal of Production Economics*, 143(2):385–394.

Kuru, N., Birbil, Ş. İ., Gurbuzbalaban, M., and Yildirim, S. (2020). Differentially private accelerated optimization algorithms. *arXiv preprint arXiv:2008.01989*.

Lai, M., Xue, W., and Hu, Q. (2019). An ascending auction for freight forwarder collaboration in capacity sharing. *Transportation Science*, 53(4):1175–1195.

Li, T., Sahu, A. K., Talwalkar, A., and Smith, V. (2020). Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60.

Mangasarian, O. L. (2011). Privacy-preserving linear programming. *Optimization Letters*, 5(1):165–172.

Mangasarian, O. L. (2012). Privacy-preserving horizontally partitioned linear programs. *Optimization Letters*, 6(3):431–436.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR.

Mothukuri, V., Parizi, R. M., Pouriyeh, S., Huang, Y., Dehghantanha, A., and Srivastava, G. (2021). A survey on security and privacy of federated learning. *Future Generation Computer Systems*, 115:619–640.

Narayanan, A. and Shmatikov, V. (2006). How to break anonymity of the Netflix prize dataset. *arXiv preprint*. https://arxiv.org/pdf/cs/0610105.pdf (accessed on 22.03.2021).

Poundarikapuram, S. and Veeramani, D. (2004). Distributed decision-making in supply chains and private e-marketplaces. *Production and Operations Management*, 13(1):111–121.

Shokri, R. and Shmatikov, V. (2015). Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1310–1321.

Speranza, M. G. (2018). Trends in transportation and logistics. *European Journal of Operational Research*, 264(3):830–836.

Sweeney, L. (1997). Weaving technology and policy together to maintain confidentiality. *The Journal of Law, Medicine & Ethics*, 25(2-3):98–110.

Toft, T. (2009). Solving linear programs using multiparty computation. In Dingledine, R. and Golle, P., editors, *Financial Cryptography and Data Security*, pages 90–107, Berlin, Heidelberg. Springer.

Topaloglu, H. (2012). A duality based approach for network revenue management in airline alliances. *Journal of Revenue and Pricing Management*, 11(5):500–517.

Toyota (2006). Fuji Heavy Industries' U.S. plant to build Toyota Camry. https://global.toyota/en/newsroom/corporate/25607796.html (accessed on 04.07.2021).

Truex, S., Baracaldo, N., Anwar, A., Steinke, T., Ludwig, H., Zhang, R., and Zhou, Y. (2019). A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, pages 1–11.

Vaidya, J. (2009). Privacy-preserving linear programming. In *Proceedings of the 2009 ACM Symposium on Applied Computing*, pages 2002–2007. ACM.

Verdonck, L., Caris, A., Ramaekers, K., and Janssens, G. K. (2013). Collaborative logistics from the perspective of road transportation companies. *Transport Reviews*, 33(6):700–719.

Wright, C. P., Groenevelt, H., and Shumsky, R. A. (2010). Dynamic revenue management in airline alliances. *Transportation Science*, 44(1):15–37.

Yang, Q., Liu, Y., Chen, T., and Tong, Y. (2019). Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19.

Zhang, T. and Zhu, Q. (2017). Dynamic differential privacy for admm-based distributed classification learning. *IEEE Transactions on Information Forensics and Security*, 12(1):172–187.