Differentially Private Linear Programming: Reduced Sub-Optimality and Guaranteed Constraint Satisfaction

Alexander Benvenuti[1], **Brendan Bialy**[2], **Miriam Dennis**[2], **Matthew Hale**[1]

[1]School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA
[2]Munitions Directorate, Air Force Research Laboratory, Eglin Air Force Base, FL, USA
abenvenuti3@gatech.edu, brendan.bialy@us.af.mil, miriam.dennis.1@us.af.mil, mhale30@gatech.edu

Abstract

Linear programming is a fundamental tool in a wide range of decision systems. However, without privacy protections, sharing the solution to a linear program may reveal information about the underlying data used to formulate it, which may be sensitive. Therefore, in this paper we introduce an approach for protecting sensitive data while formulating and solving a linear program. First, we prove that this method perturbs objectives and constraints in a way that makes them differentially private. Then, we show that (i) privatized problems always have solutions, and (ii) their solutions satisfy the constraints in their corresponding original, non-private problems. The latter result solves an open problem in the literature. Next, we analytically bound the expected sub-optimality of solutions that is induced by privacy. Numerical simulations show that, under a typical privacy setup, the solution produced by our method yields a 65% reduction in sub-optimality compared to the state of the art.

Technical Appendix — https://arxiv.org/abs/2501.19315

1 Introduction

Linear programming is used in a wide range of settings, including resource allocation, power systems, and transportation systems. In many modern systems, user data plays an increasing role in formulating such optimization problems. Sensitive information such as investor data, home power consumption, and travel routes may be use used to formulate these problems (Markowitz 1952; Stott, Marinho, and Alsac 1979), though sharing the solution to an optimization problem may leak this sensitive data (Hsu et al. 2014b). As a result, interest has arisen in solving linear programs while both (i) preserving the privacy of the data used in the problem formulation and (ii) ensuring constraint satisfaction.

In this paper we solve the open problem posed in (Munoz et al. 2021), namely, the privatization of data that is used to generate constraints when solving linear programs, while also maintaining feasibility of solutions with respect to the original, non-private constraints. For a problem with linear constraints $Ax \leq b$ and cost $c^T x$, the work in (Munoz et al. 2021) privatized the data that is used to generate b while also ensuring feasibility with respect to the original constraints. Then (Munoz et al. 2021) named it as an open problem to simultaneously privatize the data that produces A and ensure satisfaction of the original, non-private constraints. We not only solve this open problem, but in fact go one step further by simultaneously privatizing the data that produces all three — A, b, and c — with guaranteed satisfaction of the original constraints.

To produce a private linear program, we use differential privacy. Differential privacy is a statistical notion of privacy originally developed to protect entries in databases (Dwork et al. 2006), and it has seen wide use in the controls (Le Ny and Pappas 2013; Cortés et al. 2016; Hawkins and Hale 2020; Yazdani et al. 2022), planning (Chen et al. 2023; Benvenuti et al. 2024b), and federated learning (Geyer, Klein, and Nabi 2017; Agarwal, Kairouz, and Liu 2021; Chen et al. 2022; Noble, Bellet, and Dieuleveut 2022) communities for the strong guarantees that it provides.

We use differential privacy in this work partly because of its immunity to post-processing (Dwork and Roth 2014), namely that arbitrary computations on private data do not weaken differential privacy. We consider linear programs in which the cost $c^T x$ and constraint terms A and b can all depend on user data, and we use differential privacy to perturb each of these terms in order to protect the data that is used to generate them. The result is a privacy-preserving linear program. We then solve this optimization problem, which is simply a way of post-processing the privacy-preserving problem. Thus, the solution to the private problem preserves the privacy of the data used to formulate the problem, as do any downstream computations that use that solution.

As noted in (Benvenuti et al. 2024a) and (Munoz et al. 2021), common privacy mechanisms such as the Gaussian and Laplace mechanisms (Dwork and Roth 2014) add noise with unbounded support. Such mechanisms can perturb constraints by arbitrarily large amounts, which can therefore cause the solution to a privatized problem to be infeasible with respect to the original constraints. Motivated by this challenge, we develop a new matrix truncated Laplace mechanism to privatize the data that produces A, and we use the truncated Laplace mechanism developed in (Munoz et al. 2021; Geng et al. 2020) to privatize the data that produces b. This approach allows us to privatize the constraints such that they only become tighter, thereby ensuring that the solution to the private problem always satisfies the original, non-

	(Hsu et al. 2014b)	(Cummings et al. 2015)	(Dvorkin et al. 2020)	(Munoz et al. 2021)	(Benvenuti et al. 2024a)	This work
Privatize A	✓	✓			✓	✓
Privatize b	✓	✓	✓	✓		✓
Privatize c	✓	✓				✓
Satisfy Constraints				✓	✓	✓

Table 1: Comparison of differentially private linear programming approaches in the literature.

private constraints. Since the cost does not affect feasibility, we use the unbounded Laplace mechanism in (Dwork and Roth 2014) to provide privacy for the cost. We then bound the accuracy of our method, which provides users with a tool to calibrate privacy based on its tradeoff with performance. To summarize, our contributions are:

- We develop a differential privacy mechanism that simultaneously privatizes all terms in a linear program (Theorem 3.9).
- We prove that a privatized problem produces a solution that is feasible with respect the constraints in the original, non-private problem (Theorem 3.10), which solves an open problem in the literature.
- We bound the accuracy of our method, namely the increase/decrease in optimal cost, in terms of privacy parameters (Theorem 4.1).
- We empirically compare the performance of our method to the state of the art and show that the solution produced by our method yields a 65% reduction in sub-optimality relative to existing work (Section 5).

1.1 Related Work

There exists substantial previous work on differential privacy in optimization, specifically looking at privacy for objective functions in distributed optimization (Huang, Mitra, and Vaidya 2015; Wang et al. 2016; Han, Topcu, and Pappas 2016; Nozari, Tallapragada, and Cortés 2016; Dobbe et al. 2018; Lv, Yang, and Shi 2020). We differ from these works because we consider the constraints to also be sensitive, not just objectives. Privacy for optimization with linear constraints has been previously investigated in (Hsu et al. 2014b; Cummings et al. 2015; Dvorkin et al. 2020; Munoz et al. 2021; Benvenuti et al. 2024a; Kaplan et al. 2024). Both (Hsu et al. 2014b) and (Cummings et al. 2015) consider differential privacy for both the costs and constraints, but they allow for constraints to be violated, which is unacceptable in many applications, e.g., if constraints encode safety. The authors in (Dvorkin et al. 2020) analyze privacy for the constant vector in equality constraints by reformulating their optimization problem as a stochastic chance-constrained optimization problem. The authors in (Kaplan et al. 2024) consider privacy for all the constraints with a focus on maximizing the number of constraints satisfied. However, both of these works still allow for constraint violation. Both (Munoz et al. 2021) and (Benvenuti et al. 2024a) address the problem of privately solving convex optimization problems with linear constraints with guaranteed constraint satisfaction, but (Munoz et al. 2021) only privatizes b and (Benvenuti et al. 2024a) only privatizes A. We differ because we consider privacy for all components of an LP simultaneously. Moreover, (Benvenuti et al. 2024a) privatizes the entries of A themselves, though here we consider the more general setting of allowing A and b to be functions of user data, and we privatize that user data, not the entries of A and b. Table 1 summarizes our place in the literature.

1.2 Notation

For $N \in \mathbb{N}$, we use $[N] := \{1, 2, \ldots, N\}$. We use $|\cdot|$ to denote the cardinality of a set and $A \ominus B$ to denote the symmetric difference between two sets A and B. We use $\mathbb{E}[X]$ to denote the expectation of a random variable X and $\mathcal{L}(\sigma)$ to be a zero-mean Laplace distribution with scale parameter σ. We use $M_{i,j}$ to denote the $i^{th}j^{th}$ entry of a matrix. Additionally, we use $\|M\|_{1,1} = \sum_{i=1}^{m}\sum_{j=1}^{n}|M_{i,j}|$ to denote the $(1,1)$-norm of a matrix. We use $\mathbb{1}^{m\times n}$ to be an $m \times n$ matrix of all ones and $[-s\mathbb{1}^{m\times n}, s\mathbb{1}^{m\times n}]$ to be an mn-fold Cartesian product of the interval $[-s, s]$. We use $A \circ B$ as the Hadamard product between matrices A and B. We write $\operatorname{diam}(S) = \sup_{s_1,s_2\in S}\|s_1 - s_2\|_2$ for the diameter of a set S.

2 Preliminaries and Problem Formulation

2.1 Linear Programming

We consider linear programs (LPs) formed from a database $D \in \mathcal{D}$ taking the form

$$\begin{aligned} &\underset{x}{\text{maximize}} \quad c(D)^T x \\ &\text{subject to} \;\; A(D)x \le b(D), \quad x \ge 0, \end{aligned} \tag{P}$$

where $\mathcal{D}$ is the set of all possible realizations of the database D, $c(D) \in \mathbb{R}^n$ is the "cost vector", $A(D) \in \mathbb{R}^{m\times n}$ is the "constraint coefficient matrix", and $b(D) \in \mathbb{R}^m$ is the "constraint vector". We also use the sets $\mathcal{A}$ to denote the set of all realizations of $A(D)$ and $\mathcal{B}$ and $b(D)$ for all $D \in \mathcal{D}$. We define the feasible region of the LP for a database realization D as

$$\mathcal{F}(D) = \{x \in \mathbb{R}^n : A(D)x \le b(D)\}. \tag{1}$$

Remark 2.1. We include the constraint $x \ge 0$ without loss of generality since the constraints in a problem may be reformulated to shift the feasible region to the non-negative orthant without changing the problem. We do this because having strictly positive decision variables allows us have insight into how the feasible region changes when perturbing the constraints, which plays a key role in our feasibility analysis in Section 3.

Assumption 2.2. For every $D \in \mathcal{D}$, Problem (P) satisfies Slater's condition.

Remark 2.3. Assumption 2.2 is common in the optimization literature. Slater's condition says that for the constraints $Ax \leq b$ there exists a point $\bar{x}$ such that $A\bar{x} - b < 0$, and thus Assumption 2.2 states that such a point must exist for each realization of the database D. Satisfying Slater's condition implies that the feasible region $\mathcal{F}(D)$ defined by (1) has non-empty interior for all $D \in \mathcal{D}$. If $\mathcal{F}(D)$ has empty interior, then any perturbation to the constraints can produce a private problem whose solution is automatically infeasible with respect to the original, non-private constraints. Thus, such constraints are fundamentally incompatible with privacy. We enforce Assumption 2.2 in order to only consider problems where it is at least possible to attain both privacy and feasibility simultaneously, though we still must determine how to do so.

Assumption 2.4. The set $\mathcal{D}$ is bounded and the bounds are publicly available.

Assumption 2.4 is quite mild since user data may represent physical quantities that do not exceed certain bounds, e.g., with voltages in a power grid, and these can be publicly known without revealing any sensitive user data.

Problem (P) admits an equivalent dual problem of the form

$$\begin{aligned} &\underset{\mu}{\text{minimize}} \quad \mu^T b(D) \\ &\text{subject to} \;\; \mu^T A(D) \leq c(D)^T, \quad \mu \geq 0. \end{aligned}$$

2.2 Differential Privacy

We will provide differential privacy to a database D by perturbing each component of the LP that it produces, namely $A(D)$, $b(D)$, and $c(D)$. The goal of differential privacy is to make "similar" pieces of data appear approximately indistinguishable, and the notion of "similar" is defined by an adjacency relation (Dwork and Roth 2014).

Definition 2.5 (Adjacency)**.** Two databases D and D' are said to be "adjacent" if they differ in at most one entry. If two databases D and D' are adjacent, we say $\text{Adj}(D, D') = 1$; otherwise we write $\text{Adj}(D, D') = 0$.

To make adjacent pieces of data appear approximately indistinguishable, we implement differential privacy, which is done using a randomized map called a "mechanism". In its general form, differential privacy protects a sensitive piece of data y by randomizing some function of it, say $f(y)$. In the case of linear programming, we privatize three functions of the sensitive data D, namely $A(D)$, $b(D)$, and $c(D)$.

Definition 2.6 (Differential Privacy; (Dwork and Roth 2014))**.** Fix a probability space $(\Omega, \mathcal{F}, \mathbb{P})$. Let $\epsilon > 0$ and $\delta \in [0, \frac{1}{2})$ be given. A mechanism $\mathscr{M} : \mathbb{R}^{m\times n} \times \Omega \to \mathbb{R}^{m\times n}$ is (ϵ, δ)-differentially private if for all $V(D), W(D) \in \mathbb{R}^{m\times n}$ that are adjacent in the sense of Definition 2.5, we have $\mathbb{P}[\mathscr{M}(V) \in T] \leq e^{\epsilon}\mathbb{P}[\mathscr{M}(W) \in T] + \delta$ for all Borel measurable sets $T \subseteq \mathbb{R}^{m\times n}$.

Since all three components of Problem (P) require privacy, next we state a lemma on how composing private mechanisms affects privacy.

Lemma 2.7 (Sequential Composition of Private Mechanisms (Dwork and Roth 2014))**.** For $i \in [N]$, fix $\alpha_i \geq 0$ such that $\sum_{i=1}^{N} \alpha_i = 1$. Let $\mathscr{M}_i : \mathcal{D} \to \mathcal{R}_i$ for $i \in [N]$ be an $(\alpha_i\epsilon, \alpha_i\delta)$-differentially private mechanism. If $\mathscr{M}_{[N]} : \mathcal{D} \to \prod_{i=1}^{N} \mathcal{R}_i$ is defined to be $\mathscr{M}_{[N]}(D) = (\mathscr{M}_1(D), \ldots, \mathscr{M}_N(D))$ then $\mathscr{M}_{[N]}$ is (ϵ, δ)-differentially private.

We refer to the α_i's as the "privacy budget allocation", since they divide ϵ and δ among the privacy mechanisms. Lemma 2.7 implies that an algorithm containing individual privatizations of $A(D)$, $b(D)$, and $c(D)$ is itself differentially private with parameters equal to the sum of the privacy parameters from each individual privatization. This property allows us to form a linear program composed of each privatized quantity and ensure that forming that program is differentially private. Next we state a lemma that solving such an optimization problem also preserves the privacy of the underlying database D.

Lemma 2.8 (Immunity to Post-Processing; (Dwork and Roth 2014))**.** Let $\mathscr{M} : \mathbb{R}^{m\times n} \times \Omega \to \mathbb{R}^{m\times n}$ be an (ϵ, δ)-differentially private mechanism. Let $h : \mathbb{R}^{m\times n} \to \mathbb{R}^{p\times q}$ be an arbitrary mapping. Then the composition $h \circ \mathscr{M} : \mathbb{R}^{m\times n} \to \mathbb{R}^{p\times q}$ is (ϵ, δ)-differentially private.

Since solving an optimization problem is a form of post-processing, Lemma 2.8 implies that the solution to an (ϵ, δ)-differentially private optimization problem is also (ϵ, δ)-differentially private, allowing the solution to a privatized form of Problem (P) to be shared without harming the privacy of D.

2.3 Problem Statements

Consider Problem (P). Computing x^* without any protections depends on the underlying sensitive database D, and thus computing and using x^* can reveal information about D. Therefore, we seek to develop a framework for solving problems in the form of Problem (P) that preserves the privacy of D while still satisfying the constraints in Problem (P). This will be done by solving the following problems.

Problem 1. *Develop a privacy mechanism to privatize D when computing each component of a linear program (namely, $A(D)$, $b(D)$, and $c(D)$).*

Problem 2. *Prove that a solution to the privately generated optimization problem also satisfies the constraints of the original, non-private problem.*

Problem 3. *Bound the sub-optimality in solutions that is induced by the privacy mechanism in terms of the privacy parameters ϵ and δ.*

3 Private Constraints

In this section, we solve Problems 1 and 2. Specifically, we perturb the matrix $A(D)$, the vector $b(D)$, and the vector $c(D)$ in order to privatize D. Since $A(D)$, $b(D)$, and $c(D)$ all have different properties and requirements to preserve feasibility, we use separate privacy mechanisms for each. We begin with privacy for the matrix $A(D)$. The

proofs for all new technical results may be found in the Technical Appendix Section B.

3.1 Privacy for $A(D)$

Definition 3.1. The $L_{1,1}$-sensitivity of a function $f : \mathcal{D} \to \mathbb{R}^{m\times n}$ is

$$\Delta_{1,1} f = \sup_{D,D':\mathrm{Adj}(D,D')=1} \|f(D) - f(D')\|_{1,1}.$$

Next, we extend the definition of the Truncated Laplace Distribution in (Munoz et al. 2021) to matrix-valued draws.

Lemma 3.2 (Matrix-Variate Truncated Laplace Mechanism)**.** Let privacy parameters $\epsilon > 0$ and $\delta \in (0, \frac{1}{2}]$ and sensitivity $\Delta_{1,1}A$ be given. The Matrix-Variate Truncated Laplace Mechanism takes a matrix-valued function of sensitive data $F(y) \in \mathbb{R}^{m\times n}$ as input and outputs the private approximation of $F(y)$, denoted $\tilde{F}(y) = F(y) + Z \in \mathbb{R}^{m\times n}$, where $Z_{i,j} \sim \mathcal{L}_T(\sigma_A, \mathcal{S}_A)$ for all $i \in [m]$ and $j \in [n]$. Here, $\mathcal{L}_T(\sigma_A, \mathcal{S}_A)$ is the scalar truncated Laplace distribution with density $f(Z_{i,j}) = \frac{1}{\zeta}\exp\left(-\frac{1}{\sigma_A}|Z_{i,j}|\right)$, where $\mathcal{S}_A := [-s_A, s_A]$ and the values of s_A and $-s_A$ are bounds on the private outputs such that $Z_{i,j} \in \mathcal{S}_A$. We define $\zeta = \mathbb{P}(Z_{i,j} \le |s_A|)$, and σ_A is the scale parameter of the distribution. The Matrix-Variate Truncated Laplace Mechanism is (ϵ, δ)-differentially private if $\sigma_A \ge \frac{\Delta_{1,1}A}{\epsilon}$ and $s_A = \frac{\Delta_{1,1}A}{\epsilon}\log\left(\frac{mn(\exp(\epsilon)-1)}{\delta} + 1\right)$.

We apply Lemma 3.2 to the entire constraint matrix $A(D)$ to produce a differentially private constraint matrix $\bar{A}$. If some entry $A(D)_{i,j}$ is identically zero for all $D \in \mathcal{D}$, then that zero entry may represent that there is no physical relationship between a decision variable and a constraint. For example, in a smart power grid system, one home's power consumption may not influence its neighbor's power consumption. Given their practical relevance, we wish to preserve such properties when privacy is implemented. To ensure that identically zero-valued entries in $A(D)$ (which we refer to as "non-sensitive" entries) remain unchanged by privacy, we set

$$\bar{A} = A(D) + (s_A \mathbb{1}^{m\times n} + Z) \circ \mathbb{I}\{A(D) \neq 0\}, \quad (2)$$

where $\mathbb{I}\{A(D) \neq 0\} \in \mathbb{R}^{m\times n}$ is a matrix of ones and zeros where $\mathbb{I}\{A(D) \neq 0\}_{i,j} = 1$ if there exists some $D \in \mathcal{D}$ such that $A(D)_{i,j} \neq 0$ and $\mathbb{I}\{A(D) \neq 0\}_{i,j} = 0$ if $A(D)_{i,j} = 0$ for all $D \in \mathcal{D}$. We add $s_A\mathbb{1}^{m\times n}$ in (2) to ensure that the coefficients in $\bar{A}$ can only become larger than they were in $A(D)$, thus tightening the constraints to promote feasibility of private solutions with respect to the original, non-private constraints.

3.2 Privacy for $b(D)$

To enforce privacy for $b(D)$, we leverage the approach used in (Munoz et al. 2021), which we restate here for completeness. We begin by defining the sensitivity of a vector-valued function.

Definition 3.3. The ℓ_1-sensitivity of a function $f : \mathcal{D} \to \mathbb{R}^m$ is $\Delta_1 f = \sup_{D,D':\mathrm{Adj}(D,D')=1} \|f(D) - f(D')\|_1$.

We use the multivariate Truncated Laplace Mechanism to enforce privacy for the constraint vector $b(D)$.

Lemma 3.4 (Multivariate Truncated Laplace Mechanism (Munoz et al. 2021; Geng et al. 2020))**.** Let privacy parameters $\epsilon > 0$ and $\delta \in (0, \frac{1}{2}]$ and sensitivity $\Delta_1 b$ be given. The Truncated Laplace Mechanism takes a function of sensitive data $f(y) \in \mathbb{R}^m$ as input and outputs a private approximation of $f(y)$, denoted $\tilde{f}(y) = f(y) + z \in \mathbb{R}^m$, where $z_i \in \mathcal{S}_b$, with $\mathcal{S}_b := [-s_b, s_b]$, and $z_i \sim \mathcal{L}_T(\sigma_b, \mathcal{S}_b)$ for all $i \in [m]$. The multivariate truncated Laplace mechanism is (ϵ, δ)-differentially private if $\sigma_b \ge \frac{\Delta_1 b}{\epsilon}$ and $s_b = \frac{\Delta_1 b}{\epsilon}\log\left(\frac{m(\exp(\epsilon)-1)}{\delta} + 1\right)$.

We apply Lemma 3.4 to $b(D)$ to obtain

$$\bar{b} = b(D) - s_b\mathbb{1}^m + z_b \quad (3)$$

as the privatized constraint vector, where z_b is the noise added to enforce privacy from Lemma 3.4. By subtracting $s_b\mathbb{1}^m$, we ensure that each entry in the constraint vector becomes smaller, thereby tightening each constraint to promote feasibility.

3.3 Privacy for $c(D)$

To enforce privacy for $c(D)$, we use the standard Laplace mechanism, which we define next.

Lemma 3.5 (Laplace Mechanism; (Dwork and Roth 2014))**.** Let $\Delta_1 f > 0$ and $\epsilon > 0$ be given, and fix the adjacency relation from Definition 2.5. The Laplace mechanism takes sensitive data $f(y) \in \mathbb{R}^n$ as input and outputs private data $\tilde{f}(y) = f(y) + z$, where $z \sim \mathcal{L}(\sigma)$. The Laplace mechanism is $(\epsilon, 0)$-differentially private if $\sigma \ge \frac{\Delta_1 f}{\epsilon}$.

Similar to the identically zero entries of $A(D)$, an identically zero, or non-sensitive zero, entry in $c(D)$ encodes the fact that a decision variable does not impact the cost, and thus to preserve that structure we privatize only the non-sensitive zero elements in $c(D)$. Let $c(D)^0$ denote the vector of sensitive entries of $c(D)$. To produce a private cost function, we compute $\tilde{c}^0 = c(D)^0 + z_c$, where $z_c \sim \mathcal{L}(\sigma_c)$ is the noise added using Lemma 3.5 to enforce privacy. We then form $\tilde{c}$ by replacing the sensitive entries in $c(D)$ with the corresponding private entries of $\tilde{c}^0$. Since changing the cost function does not impact feasibility, $\tilde{c}^0$ requires no post-processing and may be used as-is.

3.4 Guaranteeing Feasibility

Along with privacy, we must also enforce feasibility. In order for the privately obtained solution $\tilde{x}^*$ to satisfy the constraints of the non-private problem (namely Problem (P)), it is clear that the two problems must have at least one feasible point in common. Ensuring that this is always true thus leads to the following assumption.

Assumption 3.6 (Perturbed Feasibility)**.** The set $S = \bigcap_{D\subseteq\mathcal{D}}\{x : A(D)x \le b(D)\}$ is not empty.

In words, Assumption 3.6 says that there must exist at least one point that satisfies the constraints produced by every realization of the database D. With Assumption 3.6, we

Algorithm 1: Privately Solving Linear Programs

Inputs: Problem (P), ϵ, δ, $\Delta_{1,1}A$, $\Delta_1 b$, $\Delta_1 c$, α_A, α_b,α_c
Outputs: Privacy-preserving solution $\tilde{x}^*$
Set $\sigma_A = \frac{\Delta_{1,1}A}{\alpha_A \epsilon}$
Set $\sigma_b = \frac{\Delta_1 b}{\alpha_b \epsilon}$
Set $\sigma_c = \frac{\Delta_1 c}{\alpha_c \epsilon}$
Compute the support for the constraint coefficient matrix $s_A = \frac{\Delta_{1,1}A}{\alpha_A \epsilon} \log\left(\frac{2nm(\exp(\alpha_A \epsilon)-1)}{\delta} + 1\right)$
Compute the support for the constraint vector $s_b = \frac{\Delta_1 b}{\alpha_b \epsilon} \log\left(\frac{2m(\exp(\alpha_b \epsilon)-1)}{\delta} + 1\right)$
Generate $\bar{A}$ using (2)
Generate $\bar{b}$ using (3)
Post-process $\bar{A}$ using (4)
Post-process $\bar{b}$ using (5)
Compute $\tilde{c}^0 = c(D)^0 + z_c$
Form $\tilde{c}$ by replacing each non-zero entry in c with its corresponding entry of $\tilde{c}^0$
Solve Problem (DP-P) (via any algorithm) to find $\tilde{x}^*$

post-process $\bar{A}$ from (2) and $\bar{b}$ from (3) according to

$$\tilde{A}_{i,j} = \min\left\{\bar{A}_{i,j}, \sup_{D\in\mathcal{D}} A(D)_{i,j}\right\} \text{ for all } i\in[m], j\in[n] \quad (4)$$

and

$$\tilde{b}_i = \max\left\{\bar{b}_i, \inf_{D\in\mathcal{D}} b(D)_i\right\} \text{ for all } i \in [m]. \quad (5)$$

For $\tilde{A}_{i,j}$, we do so for each (i,j) such that $A_{i,j}$ is non-zero. For $\tilde{b}_i$, we do so for all i. The outputs of these computations are the private constraint coefficient matrix $\tilde{A}$ and private constraint vector $\tilde{b}$.

Remark 3.7. Taking the minimum in (4) ensures that each entry in $\tilde{A}$ appears in some $A \in \mathcal{A}$ and taking the maximum in (5) ensures that each entry in $\tilde{b}$ appears in some $b \in \mathcal{B}$. The supremum and infumum are finite since $\mathcal{D}$ is bounded, and computing them maintains privacy since $\mathcal{D}$ does not depend on sensitive information according to Assumption 2.4.

With this privacy implementation, we will solve the optimization problem

$$\begin{aligned} &\underset{x}{\text{maximize}} \quad \tilde{c}^T x \\ &\text{subject to} \quad \tilde{A}x \leq \tilde{b}, \quad x \geq 0. \end{aligned} \quad \text{(DP-P)}$$

Algorithm 1 provides a unified overview of our approach. Note that Problem (DP-P) may be solved via any algorithm, and thus Algorithm 1 does not introduce any additional computational complexity compared to solving Problem (P).

Remark 3.8. Algorithm 1 presents our approach in the case where every component of the LP depends on the sensitive database; however, one can amend Algorithm 1 if only a subset of these components depends on the sensitive database. For example, if only $A(D)$ and $c(D)$ depend on the database, and the constraint vector b does not, then one can omit steps 4, 7, 9, and 11 from Algorithm 1, and choose $\alpha_A > 0$ and $\alpha_c > 0$ such that $\alpha_A + \alpha_c = 1$. Doing so will yield a more accurate result while still guaranteeing privacy for the database-dependent quantities.

3.5 Characterizing Privacy

Next we prove that Algorithm 1 is differentially private.

Theorem 3.9 (Solution to Problem 1)**.** *Let privacy parameters $\epsilon > 0$ and $\delta \in (0, \frac{1}{2}]$, sensitivities $\Delta_{1,1}A$, $\Delta_1 b$, and $\Delta_1 c$, and privacy budget allocations α_i for $i \in \{A, b, c\}$ be given. Let Assumptions 2.2, 2.4, and 3.6 hold. Then Algorithm 1 keeps the database D (ϵ, δ)-differentially private.*

Theorem 3.9 allows us to privatize each component of the linear program individually to generate an overall (ϵ, δ)-differentially private LP. The solution generated by solving (DP-P) then can be shared without harming privacy.

Theorem 3.10 (Solution to Problem 2)**.** *Let privacy parameters $\epsilon > 0$ and $\delta \in [0, \frac{1}{2})$ be given, and let Assumptions 2.2, 2.4, and 3.6 hold. Then Problem* (DP-P) *is guaranteed to have a solution, and that solution is guaranteed to satisfy the original, non-privatized constraints in Problem* (P)*.*

Theorem 3.10 guarantees that Algorithm 1 produces a feasible LP. Since all of the constraints are tightened by privacy, and the solution to Problem (DP-P) always exists, that solution is guaranteed to satisfy the original, non-private constraints. The conclusion of (Munoz et al. 2021) identifies the privatization of $A(D)$ with guaranteed constraint satisfaction as an open problem, and thus Theorems 3.9 and 3.10 not only solve this open problem but present, to the best of the authors' knowledge, the only private linear programming approach which can simultaneously privatize $A(D)$ and $b(D)$ while guaranteeing satisfaction of the original, non-private constraints.

4 Accuracy

In this section, we solve Problem 3 and bound the expected sub-optimality that is induced by privacy. This bound depends on (i) the largest feasible solution and the largest possible norm of a dual variable, whether it is a solution or not, (ii) the realization of the LP components at the boundary of $\mathcal{D}$, and (iii) the "closeness" of the private and non-private optimization problems in a way that we make precise.

For (i) and (ii), we define the following quantities:

$$\chi = \max_{D\in\mathcal{D}, j\in[N]} \bar{x}(D)^*_j$$

$$\Lambda = \max_{D\in\mathcal{D}} \frac{c(D)^T\eta - c(D)^T\omega}{\min\limits_{j\in[m]} -A(D)_j\eta + b(D)_j}$$

$$\hat{A} = \left[\sup_{D\in\mathcal{D}} A(D)_{i,j}\right]_{i\in[m], j\in[n]}$$

$$\hat{b} = \left[\sup_{D\in\mathcal{D}} b(D)_i\right]_{i\in[m]},$$

$$\rho = \begin{cases} \left(2m\left(\frac{\Delta_1 b}{\alpha_b \epsilon}\right)^2 + ms_b^2 + 2s_b\chi\sum_{i=1}^m \left(n_i^0 s_A\right) + \Lambda^2\left(2\left(\frac{\Delta_1 b}{\alpha_b\epsilon}\right)^2 + s_b^2\right) + \right. & \\ \left. \chi^2\sum_{i=1}^m\left(2n_i^0\left(\frac{\Delta_{1,1}A}{\alpha_A\epsilon}\right)^2 + (n_i^0 s_A)^2\right) + 2n^{0,c}\left(\frac{\Delta_1 c}{\alpha_c\epsilon}\right)^2 + \right. & \text{if } \tilde{A}_{i,j} = A(D)_{i,j} + (s_A + Z_{i,j})\mathbb{I}\{A \neq 0\}_{i,j} \text{ and } \tilde{b}_i = \\ \left. m\Lambda^2\sum_{j=1}^n\left(2m_j^0\left(\frac{\Delta_{1,1}A}{\alpha_A\epsilon}\right)^2 + (m_j^0 s_A)^2\right) + 2\chi^2 n^{0,c}\left(\frac{\Delta_1 c}{\alpha_c\epsilon}\right)^2\right)^{\frac{1}{2}} & b(D)_i - s_b + z_{b_i} \text{ for all } i,j \\ \left\| \begin{bmatrix} \sqrt{n}\|(A(D) - \hat{A})\|_F\chi \\ \sqrt{m}\|(A(D) - \hat{A})^T\|_F\Lambda \\ 2\sqrt{n}\frac{\Delta_1 c}{\alpha_c\epsilon}\chi \end{bmatrix} \right\|_2 + \left\| \begin{bmatrix} \|(b(D) - \hat{b})\|_2 \\ 2\frac{\Delta_1 c}{\alpha_c\epsilon} \\ \sqrt{m}\|b(D) - \hat{b}\|_2\Lambda \end{bmatrix} \right\|_2 & \text{otherwise} \end{cases} \tag{6}$$

where η is a solution to Problem (P) and ω is a Slater point for Problem (P). For (iii), we use Corollary 3.1 from (Robinson 1973), which we state formally in the Technical Appendix, Section A as Lemma A.1. Lemma A.1 then forms the basis for our accuracy result, which we state next.

Theorem 4.1 (Solution to Problem 3)**.** *Fix privacy parameters $\epsilon > 0$ and $\delta \in [0, \frac{1}{2})$, and let the sensitivities $\Delta_{1,1}A$, $\Delta_1 b$, and $\Delta_1 c$, and privacy budget allocations α_i for $i \in \{A, b, c\}$ be given. Let Assumptions 2.2, 2.4, and 3.6 hold. Let x^* be the solution to Problem* (P) *and let $\tilde{x}^*$ be the solution to Problem* (DP-P)*. Let $H(G, C)$ be the Hoffman constant, as defined in (Robinson 1973), associated with Problem* (P)*. Then $\mathbb{E}\left[c(D)^T x^* - c(D)^T\tilde{x}^*\right] \leq \|c(D)\|_2 H(G, C)\rho$, where ρ is defined in* (6)*.*

Remark 4.2. Given an LP in the form of Problem (P), the Hoffman constant $H(G, C)$ is well-defined and always exists. Computing exact Hoffman constants is known to be NP-Hard (Pena, Vera, and Zuluaga 2018), though a variety of upper bounds and efficient approximation algorithms for them exist, and any one of them can be used in conjunction with Theorem 4.1. A full exposition is beyond the scope of this article, and we refer the reader to (Pena, Vera, and Zuluaga 2018, 2021; Hoffman 1952) and references therein for an extended discussion.

The accuracy guarantee of Theorem 4.1 enables users to trade off the worst-case average sub-optimality induced by privacy in order to design the parameters ϵ and δ. Additionally, we find that the order of ρ is $\mathcal{O}(\log(\frac{1}{\delta})\epsilon^{-\frac{1}{2}})$ in terms of the privacy parameters and $\mathcal{O}(nm)$ in terms of the problem parameters, which implies linear growth of the error in terms of the privacy parameters in the worst case. However, in practice, we find virtually no increase in error with increasing problem size, which is shown empirically in Section 5. Next, we bound the magnitude of the fluctuations around the mean.

Theorem 4.3. *Let the conditions of Theorem 4.1 hold. Let $R = \|x^* - \tilde{x}^*\|_2$. Then,*

$$\mathbb{P}\left(R - \mathbb{E}\left[R\right] \geq \operatorname{diam}(\mathcal{F}(D))\sqrt{\log(1/t)/2}\right) \geq 1 - t.$$

Theorem 4.3 indicates that the fluctuations of the error about the mean error depends on the size of the original, non-private feasible region.

5 Numerical Simulation

In this section, we present simulations on the internet advertising setting described in (Munoz et al. 2021), which we restate here for completeness. See Technical Appendix C.2 for additional simulations. In this setting, the pages of a website are partitioned into N groups, and group $i \in [N]$ receives n_i unique visitors. The database D is the confidential business information from advertisers, such as market research on the products being advertised. For a group of M advertisers, for each $j \in [M]$, advertiser j lists a price $p_{ij}(D)$ that they are willing to pay per unique visit, and a budget $b(D)_j$ that that they are willing to spend on advertising. This setting yields the optimization problem

$$\begin{aligned} \underset{x \geq 0}{\text{maximize}} \quad & \sum_{i\in[N]}\sum_{j\in[M]} p_{ij}(D)x_{ij} \\ \text{subject to} \quad & \sum_{j\in[M]} x_{ij} \leq n_i \quad \text{for } i \in [N] \\ & \sum_{i\in[N]} p_{ij}(D)x_{ij} \leq b(D)_j \quad \text{for } j \in [M]. \end{aligned}$$

We consider two scenarios: (i) where only $p_{ij}(D)$ requires privacy for all i, j and (ii) where both $p_{ij}(D)$ and $b(D)$ require privacy. For both scenarios, $\alpha_i = 1/3$ for all i. See Technical Appendix C.1 for results with varying α_i.

For case (i), we utilize Algorithm 1 modified in the manner detailed in Remark 3.8 to provide privacy for just $p_{ij}(D)$, and we compare the quality of the solution (i.e., the loss in revenue) and the fraction of the constraints violated (i.e., how often advertisers go over budget) to that of (Hsu et al. 2014b). We note that (Hsu et al. 2014b) has known issues with privacy leakages, as pointed out in (Munoz et al. 2021). Specifically, (Hsu et al. 2014b) recommends scaling the problem by the ℓ_1 norm of the optimal solution, though doing so will alter the sensitivity of the problem, and no analysis is provided on how to compute the scaled problem's sensitivity. Additionally, the multiplicative weights algorithm used by (Hsu et al. 2014a) in problems with private constraints, such as in their Algorithm 5, may fail to converge in practice when noise draws with strong privacy are too large. Knowing these issues, (Hsu et al. 2014b) still presents the closest work to ours.

In case (i), we analyze sub-optimality, $\mathbb{E}\left[(c(D)^T x^* - c(D)^T\tilde{x}^*)/(c(D)^T x^*)\right]$ under varying levels of privacy. We set $N = 10$ and $M = 5$. Each $p_{ij}(D)$

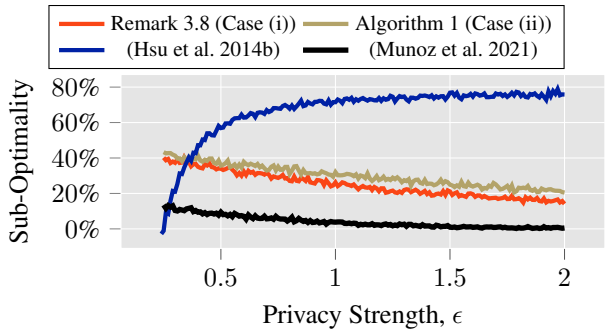

Figure 1: Performance loss with varying privacy strength. Combining Algorithm 5 with a privatized objective from (Hsu et al. 2014b) leads to constraint violation for all $\epsilon \in [0.25, 2]$. High constraint violation allows the solution to give the appearance of superior performance; however, such a solution leads to significant violation of some advertisers' budgets, which is unacceptable. Even when allowing this constraint violation, the solution produced by (Hsu et al. 2014b) still yields worse performance than that of Remark 3.8 and Algorithm 1. We also compare to (Munoz et al. 2021), and we emphasize that Munoz incurs lower sub-optimality because it privatizes only $b(D)$ in the constraints, while Algorithm 1 is used to privatize both $A(D)$ and $b(D)$. The approach in (Munoz et al. 2021) only incurs 0.5% sub-optimality at $\epsilon = 2$, while the approach in Algorithm 1 incurs roughly 20% sub-optimality, which indicates that privacy for $A(D)$ induces 19.5% additional sub-optimality.

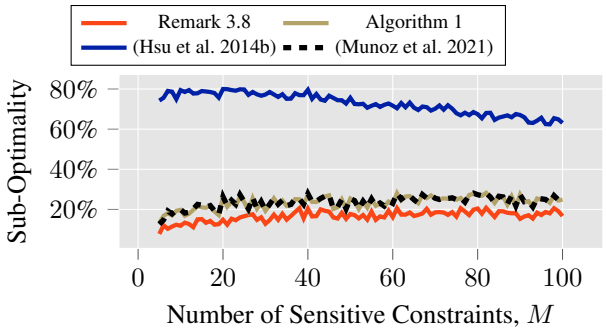

Figure 2: Performance loss with varying M. As the number of variables increases, Algorithm 5 in (Hsu et al. 2014b) allows their solver to run for more iterations, leading to improvement in accuracy, though this leads to a dramatic increase in computation time. In Algorithm 1, we see only an 11% decrease in optimal revenue with a $10\times$ increase in problem size, going from 13.3% sub-optimality with $M = 10$ to 24% sub-optimality at $M = 100$. Performance remains roughly constant with increasing number of constraints for Algorithm 1 and (Munoz et al. 2021), while Algorithm 5 in (Hsu et al. 2014b) steadily improves in performance but still has much higher sub-optimality.

is 0 with probability 0.2 and is drawn uniformly from $[0, 1]$ with probability 0.8. We also set $b_i = 10^7$ for all $i \in [N]$ and $n_j = 10^7$ for all $j \in [M]$. The performance loss for case (i) is shown in Figure 1 for $\epsilon \in [0.25, 2]$ and $\delta = 0.1$. Our work maintains zero constraint violation, as guaranteed by Theorem 3.10, while (Hsu et al. 2014b) violates constraints at every value of ϵ, up to 51% of constraints at $\epsilon = 0.25$.

For case (ii), we use Algorithm 1 without any modifications using the same problem parameters as case (i). There is no other work to the authors' knowledge that can keep both $p_{ij}(D)$ and $b(D)$ private simultaneously. However, we still present comparisons to (Munoz et al. 2021). The work in (Munoz et al. 2021) can only keep $b(D)$ private in the constraints, which means that it leaks private information about $p_{ij}(D)$ by not keeping $A(D)$ private, but we include this comparison to quantify how performance is affected by providing privacy to $A(D)$ in addition to $c(D)$ and $b(D)$.

The performance loss for case (ii) is shown in Figure 1. In case (ii), when more quantities are kept private, i.e., both $p_{ij}(D)$ and $b(D)$, Algorithm 1 still out-performs (Hsu et al. 2014b), which highlights our method's improvement over the state of the art. In addition, we see at most a 6% difference in the performance of our method between cases (i) and (ii) (which occurs when $\epsilon = 2$), indicating only minor performance loss with additional private quantities in an LP. Similarly, we find a 20% increase in sub-optimality between Algorithm 1 and (Munoz et al. 2021) at $\epsilon = 2$, indicating modest increases in sup-optimality when privatizing both $p_{ij}(D)$ and $b(D)$ in the constraints as opposed to only $b(D)$.

Next, we analyze how performance varies with increasing problem size, namely increasing M shown in Figure 2. We note that increasing M increases both the number of variables and the number of sensitive constraints. Since we have shown that our mechanism never violates the constraints while the work in (Hsu et al. 2014b) does, we instead focus our comparisons on performance. We fix $\epsilon = 1$, $\delta = 0.1$, $N = 20$, and a randomly drawn $p_{ij}(D)$, where $p_{ij}(D)$ is 0 with probability 0.2 and is drawn uniformly from $[0, 1]$ with probability 0.8. We simulate 100 samples for each $M \in \{5, 6, \ldots, 100\}$. Since Algorithm 5 in (Hsu et al. 2014b) runs for more iterations on problems with more decision variables, they see an increase in solution quality with increasing M, though in exchange for a significant increase in computation time. We see only a 10% increase sub-optimality for a 10 fold increase in M without any change in computational complexity due to privacy, highlighting the scalability of our work. Additionally when privatizing $b(D)$ with increasing M, the performance of Algorithm 1 is virtually identical to that of the method from (Munoz et al. 2021), which highlights both methods' ability to perform at scale.

6 Conclusion

We presented a method for simultaneously keeping the constraints and costs private in linear programming. We showed that this method is differentially private and that it always produces a feasible solution with respect to the original, non-private constraints. Future work will focus on guaranteed constraint satisfaction while privatizing nonlinear and stochastic constraints.

Acknowledgements

This work was partially supported by AFRL under grant FA8651-23-F-A008, NSF under CAREER grant 2422260 and Graduate Research Fellowship under grant DGE-2039655, ONR under grant N00014-24-1-2432, and AFOSR under grant FA9550-19-1-0169. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of sponsoring agencies.

References

Agarwal, N.; Kairouz, P.; and Liu, Z. 2021. The skellam mechanism for differentially private federated learning. *Advances in Neural Information Processing Systems*, 34: 5052–5064.

Benvenuti, A.; Bialy, B.; Dennis, M.; and Hale, M. 2024a. Guaranteed Feasibility in Differentially Private Linearly Constrained Convex Optimization. *IEEE Control Systems Letters*, 8: 2745–2750.

Benvenuti, A.; Hawkins, C.; Fallin, B.; Chen, B.; Bialy, B.; Dennis, M.; and Hale, M. 2024b. Differentially Private Reward Functions for Markov Decision Processes. In *2024 IEEE Conference on Control Technology and Applications (CCTA)*, 631–636. IEEE.

Chen, B.; Hawkins, C.; Karabag, M. O.; Neary, C.; Hale, M.; and Topcu, U. 2023. Differential privacy in cooperative multiagent planning. In *Uncertainty in Artificial Intelligence*, 347–357. PMLR.

Chen, W.-N.; Choo, C. A. C.; Kairouz, P.; and Suresh, A. T. 2022. The fundamental price of secure aggregation in differentially private federated learning. In *International Conference on Machine Learning*, 3056–3089. PMLR.

Cortés, J.; Dullerud, G. E.; Han, S.; Le Ny, J.; Mitra, S.; and Pappas, G. J. 2016. Differential privacy in control and network systems. In *55th IEEE Conference on Decision and Control (CDC)*, 4252–4272.

Cummings, R.; Kearns, M.; Roth, A.; and Wu, Z. S. 2015. Privacy and truthful equilibrium selection for aggregative games. In *Web and Internet Economics: 11th International Conference*, 286–299.

Dobbe, R.; Pu, Y.; Zhu, J.; Ramchandran, K.; and Tomlin, C. 2018. Customized local differential privacy for multi-agent distributed optimization. *arXiv preprint arXiv:1806.06035*.

Dvorkin, V.; Fioretto, F.; Van Hentenryck, P.; Kazempour, J.; and Pinson, P. 2020. Differentially private convex optimization with feasibility guarantees. *arXiv preprint arXiv:2006.12338*.

Dwork, C.; McSherry, F.; Nissim, K.; and Smith, A. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, 265–284. Springer.

Dwork, C.; and Roth, A. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4): 211–407.

Geng, Q.; Ding, W.; Guo, R.; and Kumar, S. 2020. Tight analysis of privacy and utility tradeoff in approximate differential privacy. In *International Conference on Artificial Intelligence and Statistics*, 89–99. PMLR.

Geyer, R. C.; Klein, T.; and Nabi, M. 2017. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*.

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

Hawkins, C.; and Hale, M. 2020. Differentially private formation control. In *2020 59th IEEE Conference on Decision and Control (CDC)*, 6260–6265. IEEE.

Hoffman, A. J. 1952. On Approximate Solutions of Systems of Linear Inequalities. *Journal of Research of the National Bureau of Standards*, 49(4).

Hsu, J.; Gaboardi, M.; Haeberlen, A.; Khanna, S.; Narayan, A.; Pierce, B. C.; and Roth, A. 2014a. Differential privacy: An economic method for choosing epsilon. In *2014 IEEE 27th Computer Security Foundations Symposium*, 398–410. IEEE.

Hsu, J.; Roth, A.; Roughgarden, T.; and Ullman, J. 2014b. Privately solving linear programs. In *Automata, Languages, and Programming: 41st International Colloquium*, 612–624. Springer.

Huang, Z.; Mitra, S.; and Vaidya, N. 2015. Differentially private distributed optimization. In *Proceedings of the 16th International Conference on Distributed Computing and Networking*, 1–10.

Kaplan, H.; Mansour, Y.; Moran, S.; Stemmer, U.; and Tur, N. 2024. On Differentially Private Linear Algebra. *arXiv preprint arXiv:2411.03087*.

Le Ny, J.; and Pappas, G. J. 2013. Differentially private filtering. *IEEE Transactions on Automatic Control*, 59(2): 341–354.

Lv, Y.-W.; Yang, G.-H.; and Shi, C.-X. 2020. Differentially private distributed optimization for multi-agent systems via the augmented lagrangian algorithm. *Information Sciences*, 538: 39–53.

Markowitz, H. 1952. Portfolio Selection. *The Journal of Finance*, 7(1): 77–91.

Munoz, A.; Syed, U.; Vassilvtiskii, S.; and Vitercik, E. 2021. Private optimization without constraint violations. In *International Conference on Artificial Intelligence and Statistics*, 2557–2565. PMLR.

Noble, M.; Bellet, A.; and Dieuleveut, A. 2022. Differentially private federated learning on heterogeneous data. In *International Conference on Artificial Intelligence and Statistics*, 10110–10145. PMLR.

Nozari, E.; Tallapragada, P.; and Cortés, J. 2016. Differentially private distributed convex optimization via objective perturbation. In *2016 American control conference (ACC)*, 2061–2066. IEEE.

Pena, J.; Vera, J.; and Zuluaga, L. 2018. An algorithm to compute the Hoffman constant of a system of linear constraints. *arXiv preprint arXiv:1804.08418*.

Pena, J.; Vera, J. C.; and Zuluaga, L. F. 2021. New characterizations of Hoffman constants for systems of linear constraints. *Mathematical Programming*, 187: 79–109.

Robinson, S. M. 1973. Bounds for error in the solution set of a perturbed linear program. *Linear Algebra and its applications*, 6: 69–81.

Stott, B.; Marinho, J.; and Alsac, O. 1979. Review of linear programming applied to power system rescheduling. In *Proceedings of the Power Industry Computer Applications Conference*, 142–154.

Wang, Y.; Hale, M.; Egerstedt, M.; and Dullerud, G. E. 2016. Differentially private objective functions in distributed cloud-based optimization. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, 3688–3694. IEEE.

Yazdani, K.; Jones, A.; Leahy, K.; and Hale, M. 2022. Differentially private LQ control. *IEEE Transactions on Automatic Control*.