Jointly Private Convex Programming

"PrivDuDe"

Justin Hsu\textsuperscript{1}, Zhiyi Huang\textsuperscript{2}, Aaron Roth\textsuperscript{1}, Steven Zhiwei Wu\textsuperscript{1}

\textsuperscript{1}University of Pennsylvania
\textsuperscript{2}University of Hong Kong

January 10, 2016
One hot summer...not enough electricity!
Solution: Turn off air-conditioning

Decide when customers get electricity

- Divide day into time slots
- Customers have values for slots
- Customers have hard minimum requirements for slots

Goal: maximize welfare
Scheduling optimization problem

Constants (Inputs to the problem)

- Customer $i$’s value for electricity in time slot $t$: $v_{t}^{(i)} \in [0, 1]$
- Customer $i$’s minimum requirement: $d_{t}^{(i)} \in [0, 1]$
- Total electricity supply in time slot $t$: $s_{t} \in \mathbb{R}$
Scheduling optimization problem

Constants (Inputs to the problem)

- Customer $i$’s value for electricity in time slot $t$: $v_t^{(i)} \in [0, 1]$
- Customer $i$’s minimum requirement: $d_t^{(i)} \in [0, 1]$
- Total electricity supply in time slot $t$: $s_t \in \mathbb{R}$

Variables (Outputs)

- Electricity level for user $i$, time $t$: $x_t^{(i)}$
Scheduling optimization problem

Maximize welfare

$$\max \sum_{i,t} v_t^{(i)} \cdot x_t^{(i)}$$

subject to constraints:

Don't exceed power supply:

$$\sum_i x_t^{(i)} \leq s_t$$

Meet minimum energy requirements:

$$x_t^{(i)} \geq d_t^{(i)}$$
Scheduling optimization problem

Maximize welfare

\[
\max \sum_{i,t} v_t^{(i)} \cdot x_t^{(i)}
\]

...subject to constraints

- Don’t exceed power supply:

\[
\sum_i x_t^{(i)} \leq s_t
\]
Scheduling optimization problem

Maximize welfare

\[
\max \sum_{i, t} v_t^{(i)} \cdot x_t^{(i)}
\]

...subject to constraints

- Don’t exceed power supply:
  \[
  \sum_i x_t^{(i)} \leq s_t
  \]

- Meet minimum energy requirements:
  \[
  x_t^{(i)} \geq d_t^{(i)}
  \]
Privacy concerns

Private data

- Values $\nu_t^{(i)}$ for time slots
- Customer requirements $d_t^{(i)}$
Privacy concerns

Private data

- Values $v_t^{(i)}$ for time slots
- Customer requirements $d_t^{(i)}$

Customers shouldn’t learn private data of others
More generally...

Convex program

- Want to maximize:

$$\sum_i f^{(i)}(x^{(i)})$$

$f^{(i)}$ concave
More generally...

**Convex program**

- **Want to maximize:**
  \[
  \sum_i f^{(i)}(x^{(i)})
  \]
  \( f^{(i)} \) concave

- **Coupling constraints:**
  \[
  \sum_i g_j^{(i)}(x^{(i)}) \leq h_j
  \]
  \( g_j^{(i)} \) convex
More generally...

Convex program

- Want to maximize:
  \[
  \sum_{i} f^{(i)}(x^{(i)}) \quad \text{\(f^{(i)}\) concave}
  \]

- Coupling constraints:
  \[
  \sum_{i} g_{j}^{(i)}(x^{(i)}) \leq h_{j} \quad \text{\(g_{j}^{(i)}\) convex}
  \]

- Personal constraints:
  \[
  x^{(i)} \in S^{(i)} \quad \text{\(S^{(i)}\) convex}
  \]
More generally...

Key feature: separable

- Partition variables: Agent $i$’s “part” of solution is $x^{(i)}$

Agent $i$’s private data affects:

- Objective $f^{(i)}$
- Coupling constraints $g^{(i)}_j$
- Personal constraints $S^{(i)}$

Examples

- Matching LP
- $d$-demand fractional allocation
- Multidimensional fractional knapsack
Our results, in one slide

**Theorem**

Let $\varepsilon > 0$ be a privacy parameter. For a separable convex program with $k$ coupling constraints, there is an efficient algorithm for privately finding a solution with objective at least

$$\text{OPT} - O\left(\frac{k}{\varepsilon}\right),$$

and exceeding constraints by at most $k/\varepsilon$ in total.

No polynomial dependence on number of variables
The plan today

- Convex program solution ↔ equilibrium of a game
- Compute equilibrium via gradient descent
- Ensure privacy
The convex program game
The players

- **Primal** player: plays candidate solutions $x \in S^{(1)} \times \cdots \times S^{(n)}$
- **Dual** player: plays dual solutions $\lambda$
The convex program two-player, zero-sum game

The players

- **Primal** player: plays candidate solutions \( x \in S(1) \times \cdots \times S(n) \)
- **Dual** player: plays dual solutions \( \lambda \)

The payoff function

- Move constraints depending on multiple players (coupling constraints) into objective as penalty terms

\[
\mathcal{L}(x, \lambda) = \sum_i f^{(i)}(x^{(i)}) + \sum_j \lambda_j \left( \sum_i g^{(i)}_j(x^{(i)}) - h_j \right)
\]

- Primal player maximizes, dual player minimizes
Idea: Solution ↔ equilibrium

Convex duality

- Optimal solution $x^*$ gets payoff $OPT$ versus any $\lambda$
- Optimal dual $\lambda^*$ gets payoff at least $-OPT$ versus any $x$

In game theoretic terms...

- The value of the game is $OPT$
- Optimal primal-dual solution $(x^*, \lambda^*)$ is an equilibrium
Idea: Solution $\leftrightarrow$ equilibrium

Convex duality

- Optimal solution $x^*$ gets payoff $\text{OPT}$ versus any $\lambda$
- Optimal dual $\lambda^*$ gets payoff at least $-\text{OPT}$ versus any $x$

In game theoretic terms...

- The value of the game is $\text{OPT}$
- Optimal primal-dual solution $(x^*, \lambda^*)$ is an equilibrium

Find an equilibrium to find an optimal solution
Idea: Solution ↔ equilibrium

Convex duality

- Optimal solution $x^*$ gets payoff $\text{OPT}$ versus any $\lambda$
- Optimal dual $\lambda^*$ gets payoff at least $-\text{OPT}$ versus any $x$

In game theoretic terms...

- The value of the game is $\text{OPT}$
- Optimal primal-dual solution $(x^*, \lambda^*)$ is an equilibrium

Find an equilibrium to find an optimal solution approximately
Finding the equilibrium
Known: techniques for finding equilibrium [FS96]

Simulated play

- First player chooses the action $x_t$ with best payoff
- Second player uses a no-regret algorithm to select action $\lambda_t$
- Use payoff $L(x_t, \lambda_t)$ to update the second player
- Repeat
Simulated play

- First player chooses the action $x_t$ with best payoff
- Second player uses a no-regret algorithm to select action $\lambda_t$
- Use payoff $\mathcal{L}(x_t, \lambda_t)$ to update the second player
- Repeat

Key features

- Average of $(x_t, \lambda_t)$ converges to approximate equilibrium
- Limited access to payoff data, can be made private
Gradient descent dynamics (linear case)

Idea: repeatedly go “downhill”

▶ Given primal point \( x_t^{(i)} \), gradient of \( \mathcal{L}(x_t, -) \) is

\[
\ell_j = \sum_i g_j^{(i)} \cdot x_t^{(i)} - h_j
\]

▶ Update:

\[
\lambda_{t+1} = \lambda_t - \eta \cdot \ell
\]
Achieving privacy
(Plain) Differential privacy [DMNS06]
More formally

**Definition (DMNS06)**

Let $M$ be a randomized mechanism from databases to range $\mathcal{R}$, and let $D, D'$ be databases differing in one record. $M$ is $(\varepsilon, \delta)$-differentially private if for every $S \subseteq \mathcal{R}$,

$$
\Pr[M(D) \in S] \leq e^\varepsilon \cdot \Pr[M(D') \in S] + \delta.
$$
More formally

Definition (DMNS06)
Let $M$ be a randomized mechanism from databases to range $\mathcal{R}$, and let $D, D'$ be databases differing in one record. $M$ is $(\varepsilon, \delta)$-differentially private if for every $S \subseteq \mathcal{R}$,

$$\Pr[M(D) \in S] \leq e^\varepsilon \cdot \Pr[M(D') \in S] + \delta.$$ 

For us: too strong!
A relaxed notion of privacy [KPRU14]

Idea

- Give separate outputs to agents
- Group of agents can’t violate privacy of other agents
A relaxed notion of privacy [KPRU14]

Idea

- Give separate outputs to agents
- Group of agents can't violate privacy of other agents

Definition

An algorithm $\mathcal{M} : C^n \rightarrow \Omega^n$ is $(\varepsilon, \delta)$-joint differentially private if for every agent $i$, pair of $i$-neighbors $D, D' \in C^n$, and subset of outputs $S \subseteq \Omega^{n-1}$,

$$\Pr[\mathcal{M}(D)_{-i} \in S] \leq \exp(\varepsilon) \Pr[\mathcal{M}(D')_{-i} \in S] + \delta.$$
Achieving joint differential privacy

“Billboard” mechanisms

- Compute signal $S$ satisfying standard differential privacy
- Agent $i$’s output is a function of $i$’s private data and $S$
Achieving joint differential privacy

“Billboard” mechanisms

- Compute signal $S$ satisfying standard differential privacy
- Agent $i$’s output is a function of $i$’s private data and $S$

Lemma (Billboard lemma [HHRRW14])

Let $S : D \rightarrow S$ be $(\varepsilon, \delta)$-differentially private. Let agent $i$ have private data $D_i \in \mathcal{X}$, and let $F : \mathcal{X} \times S \rightarrow \mathcal{R}$. Then the mechanism

$$M(D)_i = F(D_i, S(D))$$

is $(\varepsilon, \delta)$-joint differentially private.
Privacy for the dual player

- Recall gradient is

\[ \ell_j = \sum_i g_j^{(i)} \cdot x_t^{(i)} - h_j \]

- May depend on private data in a low-sensitivity way
Our signal: noisy dual variables

Privacy for the dual player

- Recall gradient is

\[ \ell_j = \sum_i g_j^{(i)} \cdot x_t^{(i)} - h_j \]

- May depend on private data in a low-sensitivity way
- Use Laplace mechanism to add noise, “noisy gradient”:

\[ \hat{\ell}_j = \sum_i g_j^{(i)} \cdot x_t^{(i)} - h_j + \text{Lap}(\Delta/\varepsilon) \]

- Noisy gradients satisfy standard differential privacy
Private action: best response to dual variables

(Joint) privacy for the primal player

- Best response problem:

\[
\max_{x \in S} \mathcal{L}(x, \lambda_t) = \max_{x \in S} \sum_{i} f^{(i)} \cdot x^{(i)} + \sum_{j} \lambda_{j, t} \left( \sum_{i} g_{j}^{(i)} \cdot x^{(i)} - h_{j} \right)
\]
Private action: best response to dual variables

(Joint) privacy for the primal player

Best response problem:

\[
\max_{x \in S} \mathcal{L}(x, \lambda_t) = \max_{x \in S} \sum_i f^{(i)} \cdot x^{(i)} + \sum_j \lambda_{j,t} \left( \sum_i g^{(i)}_j \cdot x^{(i)} - h_j \right)
\]

Can optimize separately:

\[
\max_{x^{(i)} \in S^{(i)}} f^{(i)} \cdot x^{(i)} + \sum_j \lambda_{j,t} \left( g^{(i)}_j \cdot x^{(i)} \right)
\]
Private action: best response to dual variables

(Joint) privacy for the primal player

- Best response problem:

\[
\max_{x \in S} \mathcal{L}(x, \lambda_t) = \max_{x \in S} \sum_i f^{(i)} \cdot x^{(i)} + \sum \lambda_{j,t} \left( \sum_i g^{(i)}_j \cdot x^{(i)} - h_j \right)
\]

- Can optimize separately:

\[
\max_{x^{(i)} \in S^{(i)}} f^{(i)} \cdot x^{(i)} + \sum \lambda_{j,t} \left( g^{(i)}_j \cdot x^{(i)} \right)
\]

- Key point: optimization for \(x^{(i)}\) depends only on \(\lambda\) and functions of \(i\)’s private data \((S^{(i)}, f^{(i)}, g^{(i)})\)
The algorithm: \texttt{PrivDuDe}

- For iterations $t = 1, \ldots, T$: 

  \begin{itemize}
    \item For iterations $t = 1, \ldots, T$:
      \begin{itemize}
        \item For $i = 1, \ldots, n$, compute best response:
          \begin{align*}
            x^t(i) &= \max_{x \in S(i)} f(i) \cdot x - \sum_j \lambda^t(j) \cdot g(i, j) \cdot x^t(i) \tag{1}
          \end{align*}
        \item For coupling constraints $j = 1, \ldots, k$, compute noisy gradient:
          \begin{align*}
            \hat{\ell}^t(j) &= \sum_i g(i, j) \cdot x^t(i) - h_j + \text{Lap}(\Delta / \varepsilon) \tag{2}
          \end{align*}
        \item Do gradient descent update:
          \begin{align*}
            \lambda^{t+1} &= \lambda^t - \eta \cdot \hat{\ell}^t \tag{3}
          \end{align*}
      \end{itemize}
    \end{itemize}

- Output: time averages $\frac{1}{T} \sum_t x^t(i)$ to agent $i$.
The algorithm: $\text{PrivDuDe}$

- For iterations $t = 1, \ldots, T$:
- For $i = 1, \ldots, n$, compute best response:

$$x_t^{(i)} = \max_{x \in S^{(i)}} f^{(i)} \cdot x - \sum_j \lambda_{j,t} (g_j^{(i)} \cdot x)$$
The algorithm: **PrivDuDe**

- For iterations $t = 1, \ldots, T$:
  - For $i = 1, \ldots, n$, compute best response:
    
    $$x_t^{(i)} = \max_{x \in S^{(i)}} f^{(i)} \cdot x - \sum_j \lambda_{j,t}(g_j^{(i)} \cdot x)$$

- For coupling constraints $j = 1, \ldots, k$, compute noisy gradient:
  
  $$\hat{\ell}_{j,t} = \sum_i g_j^{(i)} \cdot x_t^{(i)} - h_j + \text{Lap}(\Delta/\varepsilon)$$
The algorithm: **PrivDuDe**

- For iterations $t = 1, \ldots, T$:
  - For $i = 1, \ldots, n$, compute best response:
    \[
    x_t^{(i)} = \max_{x \in S(i)} f^{(i)} \cdot x - \sum_j \lambda_{j,t}(g^{(i)}_j \cdot x)
    \]
  - For coupling constraints $j = 1, \ldots, k$, compute noisy gradient:
    \[
    \hat{\ell}_{j,t} = \sum_i g^{(i)}_j \cdot x_t^{(i)} - h_j + \text{Lap}(\Delta/\varepsilon)
    \]
  - Do gradient descent update:
    \[
    \lambda_{t+1} = \lambda_t - \eta \cdot \hat{\ell}_t
    \]
The algorithm: **PrivDuDe**

- For iterations $t = 1, \ldots, T$:
  - For $i = 1, \ldots, n$, compute best response:
    
    $$x_t^{(i)} = \max_{x \in S(i)} f^{(i)} \cdot x - \sum_j \lambda_{j,t} (g^{(i)}_j \cdot x)$$

- For coupling constraints $j = 1, \ldots, k$, compute noisy gradient:
  
  $$\hat{\ell}_{j,t} = \sum_i g^{(i)}_j \cdot x_t^{(i)} - h_j + \text{Lap}(\Delta/\varepsilon)$$

- Do gradient descent update:
  
  $$\lambda_{t+1} = \lambda_t - \eta \cdot \hat{\ell}_t$$

- Output: time averages $\frac{1}{T} \sum_t x_t^{(i)}$ to agent $i$
Privacy guarantee

**Theorem**

**PrivDuDe** satisfies $(\varepsilon, \delta)$-joint differential privacy. The mechanism that releases just the dual variables $\lambda_t$ satisfies $(\varepsilon, \delta)$-standard differential privacy.
Accuracy guarantee

Theorem

PrivDuDe produces a solution $x$ such that:

- it achieves objective at least $\text{OPT} - \alpha$;
- it satisfies all personal constraints; and
- the total infeasibility over all coupling constraints is at most $\alpha$;

where $\alpha = \tilde{O}(\sigma k \log(1/\delta)/\epsilon)$, and $\sigma$ measures the sensitivity of the convex program.
Wrapping up
See paper for...

Approximate truthfulness

Exact feasibility
Conclusion

Main ideas

- Equilibrium $\leftrightarrow$ solution to convex program
- Joint differential privacy for separable convex programs

PrivDuDe

- Approximately solve separable convex programs
- Satisfies (joint) differential privacy
- Error/infeasibility linear in number of coupling constraints
Open problems and future directions

Expanding the class of convex programs

- Can we handle something beyond separable convex programs?
- Terms depending on at most two agents?

Improving the accuracy

- Is linear dependence on number of constraints $k$ necessary?
- What is the best dependence possible?
Jointly Private Convex Programming

“PrivDuDe”

Justin Hsu\(^1\), Zhiyi Huang\(^2\), Aaron Roth\(^1\), Steven Zhiwei Wu\(^1\)

\(^1\)University of Pennsylvania
\(^2\)University of Hong Kong

January 10, 2016