New Developments of First Order Methods for Linear Programming

Haihao (Sean) Lu

University of Chicago

US-Mexico Workshop on Optimization and its Applications in honor of Steve Wright’s 70th birthday

\footnote{part of the work was done at Google}
Steve and Linear Programming

Steve’s papers on LP

- Implementing proximal point methods for linear programming, 1990
- PCx: An interior-point code for linear programming, 1999
- Modified Cholesky factorizations in interior-point algorithms for linear programming, 1999
- Warm-start strategies in interior-point methods for linear programming, 2002
- Coordinate linear variance reduction for generalized linear programming, 2022
- ...
Collaborators on this Project

David Applegate  Mateo Diaz  Oliver Hinder

Miles Lubin  Brendan O’Donoghue  Warren Schudy  Jinwen Yang
Essentially based on the below paper

- H Lu, J Yang. “On the Geometry and Refined Rate of Primal-Dual Hybrid Gradient for Linear Programming”, *in preparation*

But also mentioned the below earlier papers:

- D Applegate, M Diaz, O Hinder, H Lu, M Lubin, B O’Donoghue, W Schudy. “Practical Large-Scale Linear Programming using Primal-Dual Hybrid Gradient”, *NeurIPS 2021*
- D Applegate, M Diaz, H Lu, M Lubin. “Infeasibility detection with primal-dual hybrid gradient for large-scale linear programming”, revision at *SIOPT*
- H Lu, J Yang. “On the Geometry and Refined Rate of Primal-Dual Hybrid Gradient for Linear Programming”, *in preparation*

Our solver PDLP is publicly available:

- PDLP is open-sourced through Google *ORTools*
- You can also use cvxpy to call PDLP
Linear Programming (LP)

**Linear Programming** (in standard form):

\[
\begin{align*}
\min & \quad c^T x \\
\text{s.t.} & \quad Ax = b \\
& \quad x \geq 0
\end{align*}
\]

- There are wide applications of LP in practice
Is LP a solved problem?

- There are similar claims since 1970s
- Modern commercial LP solvers, such as Gurobi, cplex, Mosek, copt, etc, can usually provide reliable solutions
- Current LP solvers only work for single machine, thus not suitable for big data era

The two previous golden periods for computational LP

- 1947-early 1970s: Simplex method
- late 1980s-early 2000s: barrier method, dual simplex method, dual steepest-edge algorithms, sparse linear algebra

The size of a neural network one can solve is much larger than LP
The Goal of the Project

Our goal:

Develop a general-purpose LP solver that can solve problems 1000 times larger than the state-of-the-art solvers
The Scalability Issues of Current Approaches

The computational bottleneck for both simplex and barrier method is **linear equation solving**

- **Parallel computing issue**: simplex is limited to $\sim 2x$ speed-ups, barrier is limited to multithreaded CPU
- **Memory issue**: Factorization requires at least $10 \sim 100$ times more memory than the constraint matrix

FOM requires at most **matrix-vector multiplication**

- Sparse matrix-vector multiplication (**SpMV**) is well studied and can scale on
  - multi-threaded CPUs
  - GPUs
  - Distributed setting
- No matrix factorization, thus memory is not an issue
How Large an LP Can You Solve?

- Not well defined:
  - Minimal spanning tree with $10^9$ node, Euclidean distance: LP with $10^{18}$ variables, $2^{10^{18}}$ constraints
  - 500m variables/constraints (stochastic programming via special-purpose decomposition and distributed computing)

- 2019 Stackexchange question:
  - 65m variables/constraints, 325m nonzeros (unstructured problem, barrier, 5 days)

- Our try with Gurobi:
  - TSP cutting-plane lower bound: 475m nonzero, 1tb memory
  - LP formulation of pagerank: 800k nonzeros

- Our FOM:
  - TSP cutting-plane lower bound: 91b nonzero, 2000 cpus
Related Literature

There have been research on FOM for LP since 1950s, but they were not numerically successful [bixby 2012]

Recently, this idea reemerges due to the large-scale applications:

- SCS: Operator splitting/ADMM [O’Donoghue, Chu, Parikh, Boyd, 2016]
- Abip+: ADMM-based interior-point method [Deng et al., 2022]
- Semi-smooth Newton augmented Lagrangian [Li, Sun, Toh, 2019]
- Nesterov’s smoothing and accelerated methods [Basu, Ghoting, Mazumder, Pan, 2020]
Overview of the Talk

- Algorithm and computation
  - The basic FOM we use
  - Many enhancements
  - Computational results on LP benchmark set and larger instances

- Complexity Theory
  - Complexity theory and faster algorithms

- Geometry and Refined Rate
  - Finite time identification for degenerate LP
  - Refined complexity of (restarted) PDHG for LP
Algorithm and Computational Results
Primal-Dual Hybrid Gradient (PDHG)

LP (in standard form):

\[
\begin{align*}
\min & \quad c^T x \\
\text{s.t.} & \quad Ax = b, \quad x \geq 0
\end{align*}
\]

LP (primal-dual form):

\[
\begin{align*}
\min \max_{x \geq 0, y} c^T x + y^T b - y^T Ax
\end{align*}
\]

Primal-Dual Hybrid Gradient (PDHG) [Chambolle and Pock 2011]

\[
\begin{align*}
x^+ &= \text{proj}_{R^+_n}(x + \eta A^T y - \eta c) \\
y^+ &= y - \tau A(2x^+ - x) + \tau b
\end{align*}
\]

- \(\eta\) and \(\tau\) are the primal and dual step-size, respectively.
- Cost per iteration is matrix vector multiplication.
- Primal-dual gap has \(O(1/\epsilon)\) convergence rate for convex-concave problems [Chambolle and Pock 2011].
PDHG is the most promising FOM we tried among:

- Restarted subgradient [Yang and Lin, 2018]
- Radial subgradient [Grimmer 2018]
- Automatic subgradient [Renegar and Grimmer, 2021]
- Accelerated methods for hyperbolic programming [Renegar 2016, 2019]
- Mirror prox [Nemirovski 2004]
- Douglas-Rachford [Douglas and Rachford, 1956]
- Nesterov’s smoothing and accelerated methods [Nesterov 2005]
- Nonsmooth BFGS [Lewis and Overton 2009]
- Stochastic PDHG [Chambolle et al 2018]
- Matrix-free barrier
- ...
PDL(Primal Dual Algorithms for LP)

On a benchmark set with 383 instances and 1 hour time limit

<table>
<thead>
<tr>
<th>Method</th>
<th>Solved to $10^{-4}$ (rela. err.)</th>
<th>Solved to $10^{-8}$ (rela. err.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDHG</td>
<td>40%</td>
<td>19%</td>
</tr>
<tr>
<td>PDLP</td>
<td>97%</td>
<td>85%</td>
</tr>
</tbody>
</table>

Indeed PDHG itself does not work well. We propose many enhancements

- Adaptive step sizes
- Diagonal preconditioning
- Re_starts
- Infeasibility detection
- LP presolving
Adaptive step sizes

- We adaptively choose the step-size

Diagonal preconditioning

\[
\min_{x \geq 0} c^T x \quad \iff \quad \min_{x \geq 0} (D_2 c)^T x
\]

\[
s.t. \quad Ax = b \quad \iff \quad s.t. \quad (D_1 AD_2)x = D_1 b
\]

- \(D_1\) and \(D_2\) are diagonal matrices with positive diagonal entries
- \(D_2x^*\) recovers an optimal solution to the original problem

Infeasibility Detection

- We show that if the primal problem is infeasible, then \(y^{k+1} - y^k\) and \(y^k/k\) converges to an infeasibility certificate; same to the dual infeasible problem.

Presolve

- Use heuristics to get rid of redundant constraints or variables
Summary of Computational Results

- Medium-size instances: Mittelmann benchmark with 50 instances
- Large instances ($10m - 100m$ nonzeros): three instances
- A huge instance ($100b$ nonzeros)
### Mittelmann Benchmark of Barrier LP (June 2022)

#### Benchmark of Barrier LP solvers (15 Jun 2022)

Choose base solver for comparison:

<table>
<thead>
<tr>
<th>solver</th>
<th>score (as reported)</th>
<th>solved of 50</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>COPT-5.0.0</strong></td>
<td>1.00 (1.00)</td>
<td>100%</td>
</tr>
<tr>
<td><strong>Gurobi-9.5.1</strong></td>
<td>1.33 (1.33)</td>
<td>98%</td>
</tr>
<tr>
<td><strong>MindOpt-0.18.2</strong></td>
<td>2.47 (2.47)</td>
<td>98%</td>
</tr>
<tr>
<td><strong>MOSEK-9.3.20</strong></td>
<td>5.21 (5.21)</td>
<td>94%</td>
</tr>
<tr>
<td><strong>PDLP$</strong></td>
<td>13.91 (13.90)</td>
<td>86%</td>
</tr>
<tr>
<td><strong>KNITRO-13.0.0</strong></td>
<td>15.75 (15.70)</td>
<td>78%</td>
</tr>
<tr>
<td><strong>HiGHS-1.2.2</strong></td>
<td>22.32 (20.30)</td>
<td>84%</td>
</tr>
<tr>
<td><strong>MATLAB-R2020b</strong></td>
<td>49.74 (40.80)</td>
<td>72%</td>
</tr>
<tr>
<td><strong>Tulip-0.9.3</strong></td>
<td>55.44 (55.40)</td>
<td>66%</td>
</tr>
<tr>
<td><strong>CLP-1.17.7</strong></td>
<td>77.79 (77.80)</td>
<td>70%</td>
</tr>
</tbody>
</table>

- **PDLP$** uses $10^{-4}$ relative tolerance
We work with a department store on personalized marketing to maximize profit.

<table>
<thead>
<tr>
<th>Model</th>
<th># nonzeros</th>
<th>Computation Time of Gurobi / PDLP (in seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Primal Simplex</td>
</tr>
<tr>
<td>model A</td>
<td>25m</td>
<td>15488</td>
</tr>
<tr>
<td>model B</td>
<td>37m</td>
<td>31138</td>
</tr>
<tr>
<td>model C</td>
<td>25m</td>
<td>-</td>
</tr>
<tr>
<td>model D</td>
<td>13m</td>
<td>-</td>
</tr>
</tbody>
</table>

- means an error was raised for Gurobi

PDLP solves with $10^{-6}$ relative accuracy
### Large Instance: PageRank

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Edges</th>
<th>PDLP</th>
<th>Gurobi Barrier</th>
<th>Gurobi Simplex</th>
</tr>
</thead>
<tbody>
<tr>
<td>10k</td>
<td>30k</td>
<td>2.7 sec</td>
<td>3.8 sec</td>
<td>43 sec</td>
</tr>
<tr>
<td>100k</td>
<td>300k</td>
<td>19 sec</td>
<td>30 min</td>
<td>19 hours</td>
</tr>
<tr>
<td>1m</td>
<td>3m</td>
<td>7.2 min</td>
<td>~1 month</td>
<td>-</td>
</tr>
<tr>
<td>10m</td>
<td>30m</td>
<td>6.7 hours</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

- Synthetic PageRank instance with Barabasi-Albert random graphs
- PDLP (C++, 20 threads) solves to $10^{-6}$ relative accuracy
- Gurobi barrier with 20 threads and concurrent primal/dual simplex with 2 threads
- For 1M nodes, Gurobi estimates each barrier iteration takes 4.6 days
Robust production inventory problem with linear decision rules

\[
\begin{align*}
\text{minimize} & \quad \sum_{\zeta_1 \in U_1, \ldots, \zeta_{T+1} \in U_{T+1}} \left\{ \sum_{t=1}^{T} \sum_{e=1}^{E} c_{te} \left( \sum_{s=1}^{t} y_{t,s} \zeta_s \right) \right\} \\
\text{subject to} & \quad \sum_{t=1}^{T} \left( \sum_{s=1}^{t} y_{t,s} \zeta_s \right) \leq Q_e \quad \forall e \in [E] \\
& \quad 0 \leq \left( \sum_{s=1}^{t} y_{t,s} \zeta_s \right) \leq p_{te} \quad \forall e \in [E], t \in [T] \\
& \quad V_{\min} \leq v_1 + \sum_{\ell=1}^{E} \sum_{e=1}^{E} \left( \sum_{s=1}^{\ell} y_{\ell,s} \zeta_s \right) - \sum_{s=2}^{t+1} \zeta_s \leq V_{\max} \quad \forall t \in [T] \\
& \quad \forall \zeta_1 \in U_1, \ldots, \zeta_{T+1} \in U_{T+1}.
\end{align*}
\]

- 2.1m variables, 5.2m constraints, 19m nonzeros
- Gurobi primal simplex solve to optimality with 531160 iterations in 11800s
- PDLP solves to $10^{-4}$ accuracy with 358464 iterations in 26514s
PDLP (distributed version)

Successfully running on a big TSP LP relaxation instance

- 3.5 B variables, 475M constraints, 92B nonzeros
- 2000 cpu, \( \sim 13 \) seconds/iteration
- 5 days to solve to \( 10^{-4} \) accuracy
Multi-threaded and distributed system speedup

- Figure plots the speedup over 4-threaded shared memory single machine
- On an TSP instance that can fit into memory
- Speedup on running one iteration of PDLP
Complexity Theory and Restarted PDHG for LP

Based on:

The previous convergence results for PDHG on LP are mostly sublinear\(^2\):

- Last iteration: \(O(1/\epsilon^2)\)
- Average iteration: \(O(1/\epsilon)\)
- Local linear convergence after the active basis is fixed

On the other hand, users of LP expect high accuracy solutions.

\(^2\)Other FOMs on LP may achieve linear convergence.
Two Fundamental Questions

- Is there a version of PDHG that can achieve global linear convergence for LP?

- Are there possibly better FOMs?
Two-Loop Restarted PDHG

**Algorithm 1: Restarted PDHG**

1. **Input:** An initial solution \((x^0, y^0, 0, 0)\);
2. repeat
   3. **initialize the inner loop.** inner loop counter \(t \leftarrow 0\);
      4. repeat
         5. \((x^{n,t+1}, y^{n,t+1}) \leftarrow \text{PDHG}((x^{n,t}, y^{n,t}))\);
         6. \((\bar{x}^{n,t+1}, \bar{y}^{n,t+1}) \leftarrow \frac{1}{t+1} \sum_{i=1}^{t+1} (x^{n,i}, y^{n,i})\);
         7. \(t \leftarrow t + 1\);
      8. until a restart conditions holds;
   9. **restart the outer loop.** \((x^{n+1,0}, y^{n+1,0}) \leftarrow (\bar{x}^{n,t}, \bar{y}^{n,t}), n \leftarrow n + 1\);
10. until \((x^n, y^n, 0)\) convergence;
11. **Output:** \(z^{n,0}\).

We have two restart conditions:

- Fixed frequency restart (if the problem parameter is known)
- Adaptive restart
Visualization of Restarted PDHG

\[ \min_x \max_y xy \]
Visualization of Restarted PDHG

\[ \min_x \max_y \ xy \]
Visualization of Restarted PDHG

\[ \min_x \max_y xy \]
The table summaries the complexity of the last iteration, the average iteration of PDHG, the output of restarted PDHG for solving LP

<table>
<thead>
<tr>
<th>Last Iterate</th>
<th>Average Iterate</th>
<th>Restarted</th>
<th>Lower Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Theta \left( \left( \frac{\gamma}{\alpha} \right)^2 \log \left( \frac{1}{\epsilon} \right) \right)$</td>
<td>$\Theta \left( \frac{\gamma}{\alpha \epsilon} \right)$</td>
<td>$\Theta \left( \frac{\gamma}{\alpha} \log \left( \frac{1}{\epsilon} \right) \right)$</td>
<td>$O \left( \frac{\gamma}{\alpha} \log \left( \frac{1}{\epsilon} \right) \right)$</td>
</tr>
</tbody>
</table>

- $\gamma = \|A\|_2$ is the smoothness parameter of the problem
- $\alpha$ is the “sharpness” of the primal-dual problem (next page for more details)
- $\Theta$ refers the rate is tight for the algorithm
- This shows that restarted PDHG is an optimal FOM for LP
A Bound on the Sharpness for LP

**Proposition: LP is Sharp**

LP $\alpha$-sharp with

$$\alpha \geq \frac{1}{H(K)(1 + 4\text{dist}((x^0, y^0), (X^*, Y^*)))} ,$$

where $H(K)$ is the Hoffman constant of the KKT system of LP.

$H(K)$ is the Hoffman constant of the inequality set $\{K[x; y] \geq h\}$, where

$$K := \begin{pmatrix} I & 0 \\ -A & 0 \\ A & 0 \\ 0 & -A^T \\ -c^T & b^T \end{pmatrix} , \quad h := \begin{pmatrix} 0 \\ -b \\ b \\ -c \\ 0 \end{pmatrix}.$$
Behaviors and Refined Complexity of (Restarted) PDHG for LP

Based on:

- H Lu, J Yang. “On the Geometry and Refined Rate of Primal-Dual Hybrid Gradient for Linear Programming”, *in preparation*
Issues of the Hoffman Constant Argument

Last section shows that

- Restarted PDHG has $\Theta \left( \frac{\gamma}{\alpha} \log \frac{1}{\epsilon} \right)$ rate
- Last iteration of vanilla PDHG has $\Theta \left( \left( \frac{\gamma}{\alpha} \right)^2 \log \frac{1}{\epsilon} \right)$ rate
Issues of the Hoffman Constant Argument

Last section shows that

- Restarted PDHG has $\Theta\left(\frac{\gamma}{\alpha} \log \frac{1}{\epsilon}\right)$ rate
- Last iteration of vanilla PDHG has $\Theta\left(\left(\frac{\gamma}{\alpha}\right)^2 \log \frac{1}{\epsilon}\right)$ rate

However,

- $\alpha$ depends on the Hoffman constant of the KKT system, which is known to be loose and hard to interpret
- A characterization of the Hoffman constant for the linear system $\{Kz \geq h\}$ is

$$\max_{B \subseteq \{1, \ldots, m\}, A_J \text{ full row rank}} \frac{1}{\min_{R_+^J, \|v\|=1} \|K_J^T v\|}$$

- The above bound does not characterize the behaviors of the algorithm (next page for examples)
Two Stages of the Convergence

There are two stages of convergence

- Slow initial convergence
- “Fast” eventual convergence
Typical Behaviors of PDHG for LP

**Question:**

- What controls the eventual linear convergence rate?
- How long does it take to get into the eventual linear stage?
A Simple yet Representative Example

Consider a class of 2-d dual LP with parameter \((\kappa, \delta)\):

\[
\max b^T y, \text{ s.t., } A^T y \leq c
\]

\[
\begin{align*}
\max & \quad y_2 \\
\text{s.t.} & \quad y_1 \geq -1 \\
& \quad y_1 \leq 1 \\
& \quad y_1 + \frac{1}{\kappa} y_2 \leq 1 \\
& \quad -y_1 + \frac{1}{\kappa} y_2 \leq 1 \\
& \quad y_2 \leq \kappa - \delta
\end{align*}
\]

- \(0 < \kappa < 1\) controls the condition number of the matrix \(A\)
- \(0 < \delta < 1 + \kappa\) controls closeness to degeneracy
There are two stages of convergence

- $\delta$ small: show first stage
- $\kappa$ small: slow second stage
What is Going on in the Two Stages?

A high-level and non-rigorous answer (a rigorous answer later)

Stage 1: Slow initial convergence
- This is the process to identify active basis set $B$ such that $x_B^* > 0$
- The driving force of this stage is closeness to degeneracy (i.e. $\delta$)

Stage 2: “Fast” eventual convergence
- Once the active set is fixed, the dynamic has faster linear convergence
- The driving force of this stage is the condition number of the matrix $A_B$ (i.e. $\kappa$)

This behavior is tied with the concept of partial smoothness and identification [Lewis, Liang; Lewis, Zhang; Hare, Lewis; etc]
Definition: Non-active set and active set

Let $z^* = (x^*, y^*)$ be the converging optimal solution. We denote

\[
N := \{ i : c_i - A_i^T y^* > 0 \}
\]
\[
B := \{ i : c_i - A_i^T y^* = 0 \}
\]
\[
B_1 := \{ i : c_i - A_i^T y^* = 0, x_i^* > 0 \}
\]
\[
B_2 := \{ i : c_i - A_i^T y^* = 0, x_i^* = 0 \}.
\]
**Definition: Non-active set and active set**

Let $z^* = (x^*, y^*)$ be the converging optimal solution. We denote

$$N := \{ i : c_i - A_i^T y^* > 0 \}$$

$$B := \{ i : c_i - A_i^T y^* = 0 \}$$

$$B_1 := \{ i : c_i - A_i^T y^* = 0, x_i^* > 0 \}$$

$$B_2 := \{ i : c_i - A_i^T y^* = 0, x_i^* = 0 \}.$$

**Definition: Non-degenerate trajectory**

We say the PDHG trajectory is non-degenerate if the converging optimal solution $z^*$ satisfies strict complimentary slackness, i.e., $B_2 = \emptyset$ or equivalently

$$x_i^* > 0 \iff c_i - A_i^T y^* = 0.$$
Some Definitions

Definition: Non-active set and active set

Let $z^* = (x^*, y^*)$ be the converging optimal solution. We denote

$$N := \{i : c_i - A_i^T y^* > 0\}$$

$$B := \{i : c_i - A_i^T y^* = 0\}$$

$$B_1 := \{i : c_i - A_i^T y^* = 0, x_i^* > 0\}$$

$$B_2 := \{i : c_i - A_i^T y^* = 0, x_i^* = 0\}.$$ 

Definition: Non-degenerate trajectory

We say the PDHG trajectory is non-degenerate if the converging optimal solution $z^*$ satisfies strict complimentary slackness, i.e., $B_2 = \emptyset$ or equivalently

$$x_i^* > 0 \iff c_i - A_i^T y^* = 0.$$ 

Definition: Near-Degeneracy

We denote $\delta := \min\{\min_{i \in N}(c - A^T y^*)_i, \min_{i \in B_1} x_i^*\}$ the closeness to degeneracy of the trajectory.
Local Sharpness

**Definition: Local Sharpness**

- **(Primal local sharpness)** Denote $\alpha_p > 0$ as a constant such that
  \[
  \alpha_p \text{dist}(u_B, U_B^*) \leq \left\| \begin{pmatrix} A_B u_B \\ [c_B^T u_B]^+ \end{pmatrix} \right\|, \quad u_B \geq 0,
  \]
  where $U_B^* = \{u_B : A_B u_B = 0, c_B^T u_B \leq 0, u_B \geq 0\}$.

- **(Dual local sharpness)** Denote $\alpha_d > 0$ as a constant such that
  \[
  \alpha_d \text{dist}(v_B, V_B^*) \leq \left\| \begin{pmatrix} [-A_B v]^+ \\ [b^T v]^+ \end{pmatrix} \right\|,
  \]
  where $V_B^* = \{v : A_B^T v \geq 0, b^T v \leq 0\}$.

- In general, $\alpha_p, \alpha_d \ll \alpha$
- $\alpha_p$ and $\alpha_d$ does not depend on $A_N$
- $\alpha_p$ does not depend on $b$ and $\alpha_d$ does not depend on $c$
- Both $\alpha_p$ and $\alpha_d$ have geometric meanings (i.e., angle between certain spaces)
Refined Complexity

**Theorem: Two-stage convergence of PDHG for non-degenerate LP**

Consider PDHG for LP, then we have

- **(Finite time identification)** We have $x^k_N = 0, x^k_B > 0$, for any $k \geq K := O \left( \frac{\gamma^2}{\alpha^2} \log \left( \frac{1}{\delta} \right) + \frac{\gamma}{\delta} \right)$.

- **(Linear convergence after identification)** After identification, the PDHG find a solution $z$ such that $\text{dist}(z, Z^*) \leq \epsilon$ within the following number of iterations
  
  $$O \left( \frac{||A_B||^2}{(\min(\alpha_P, \alpha_D))^2} \log \left( \frac{1}{\epsilon} \right) \right).$$

- **(Total complexity)** The total number of iterations to find an $\epsilon$ close solution is
  
  $$O \left( \frac{\gamma^2}{\alpha^2} \log \left( \frac{1}{\delta} \right) + \frac{\gamma}{\delta} + \frac{||A_B||^2}{(\min(\alpha_P, \alpha_D))^2} \log \left( \frac{\delta}{\epsilon} \right) \right).$$

Last iteration of vanilla PDHG has complexity

$$\Theta \left( \left( \frac{\gamma}{\alpha} \right)^2 \log \frac{1}{\epsilon} \right)$$
Degenerate Case, Restarted PDHG

Theorem: Two-stage convergence of Restarted PDHG for LP

Consider Restarted PDHG for LP, then we have

- **(Finite time identification)** We have \( x_N^k = 0, x_{B_1}^k > 0 \), for any

  \[
  k \geq K := O \left( \frac{\gamma}{\alpha} \log \left( \frac{1}{\delta} \right) + \frac{\gamma}{\delta} \right).
  \]

- **(Linear convergence after identification)** After identification, the PDHG find a solution \( z \) such that \( \text{dist}(z, Z^*) \leq \epsilon \) within the following number of iterations

  \[
  O \left( \frac{\|A_B\|}{\min(\alpha_P, \alpha_D)} \log \left( \frac{1}{\epsilon} \right) \right).
  \]

- **(Total complexity)** The total number of iterations to find an \( \epsilon \) close solution is

  \[
  O \left( \frac{\gamma}{\alpha} \log \left( \frac{1}{\delta} \right) + \frac{\gamma}{\delta} + \frac{\|A_B\|}{\min(\alpha_P, \alpha_D)} \log \left( \frac{\delta}{\epsilon} \right) \right).
  \]
Two Stages of the Convergence

\[ R^2 = 0.904 \]

\[ \text{metric} = \| (x^0, y^0) - (x^*, y^*) \|_2 \| A \|_2 / \delta \]

- Each point is one dataset from MIPLIB, and we choose those vanilla PDHG solves in a reasonable time.
Degeneracy itself does not slow down the convergence, but near-degeneracy does.
Open Questions and Future Directions

This talk is just the beginning of FOM for LP. Many open questions in this area

- What is the condition number theory?
- More reliable step-size choices?
- Extension to QP, SDP, etc?
- Extension to mixed-integer programming (crossover)?
- More applications?

Thank you!
Additional Slides
Improvements of the Enhancements

MIP Relaxations Improvements

Normalized KKT passes SGM10

- tolerance 1E-04
- tolerance 1E-08

PDHG +restarts +scaling +primal weight +step size +presolve (= PDLP)

7 × 10^{-2}
7 × 10^{-1}
10^{-1}
10^0
5 × 10^{-1}
3 × 10^{-1}
2 × 10^{-1}
10^{-2}
10^{-3}
10^{-4}

PDHG +restarts +scaling +primal weight +step size +presolve (= PDLP)
PDLP vs Glop (Best of Both)

- PDLP faster on 33% of instances, and 10× faster on 12%.
PDLP (4 threads) vs Glop (Best of Both)

- PDLP faster on **39%** of instances, and **10× faster on 15%**.
Applications: Personalized Marketing

We work with a department store on personalized marketing to maximize profit

- 3 states in U.S., 2.5 million targeting customers
- 5 marketing treatments

**Goal**: target different marketing actions to different customers to maximize the expected profit

**Constraints** of the marketing actions:

- *Volume constraints*: limit the total number of promotion
- *Similarity constraints*: make sure the promotion plan is fair

**Data**: demographic info (income, size, age, ...), location (zipcode, average income, ...), past activity, etc

This is an ongoing work with Duncan Simester and Yuting Zhu at MIT
LP Formulation for Personalized Marketing

We have different LP formulations. One of it is

\[
\max_{x_i^j} \sum_{i=1}^{N} \sum_{j=1}^{5} p_i^j x_i^j \\
\text{s.t.} \quad a_k^j \leq \sum_{i \in S_k} x_i^j \leq b_k^j, \quad \text{for } j = 1, \ldots, 5, \quad k = 1, \ldots, K \\
\frac{1}{n_{k_1}} \sum_{i \in S_{k_1}} x_i^j \leq \lambda \frac{1}{n_{k_2}} \sum_{i \in S_{k_2}} x_i^j, \quad \text{for } k_1 = 1, \ldots, K, \quad k_2 = 1, \ldots, K \\
\sum_{i=1}^{N} \sum_{j=1}^{5} c_i^j x_i^j \leq C \\
\sum_{j=1}^{5} x_i^j \leq 1, \quad \text{for } i = 1, \ldots, N \\
x_i^j \geq 0 ,
\]

- This LP has 14m variables, 3m constraints, 37m nonzeros
- Either beyond the capacity of Gurobi, or we are about 100 times faster than Gurobi
Large Instance: Production Inventory Problem (Ben-Tal et al 2004)

Robust production inventory problem with linear decision rules

\[
\begin{align*}
\text{minimize} & \quad \sum_{t \in [T]} \sum_{e \in [E]} \sum_{s \in \mathcal{U}_e} c_{te} \left( \sum_{s=1}^{t} y_{t,s,e} \zeta_s \right) \\
\text{subject to} & \quad \sum_{s=1}^{T} \left( \sum_{t=1}^{T} y_{t,s,e} \zeta_s \right) \leq Q_e \quad \forall e \in [E] \\
& \quad 0 \leq \left( \sum_{s=1}^{t} y_{t,s,e} \zeta_s \right) \leq p_{te} \quad \forall e \in [E], t \in [T] \\
& \quad V_{\min} \leq v_1 + \sum_{\ell=1}^{E} \sum_{e=1}^{E} \left( \sum_{s=1}^{\ell} y_{\ell,s,e} \zeta_s \right) - \sum_{s=2}^{t+1} \zeta_s \leq V_{\max} \quad \forall t \in [T] \\
& \quad \forall \zeta_1 \in \mathcal{U}_1, \ldots, \zeta_{T+1} \in \mathcal{U}_{T+1}.
\end{align*}
\]

- 2.1m variables, 5.2m constraints, 19m nonzeros
- Gurobi primal simplex solve to optimality with 531160 iterations in 11800s
- PDLP solves to $10^{-4}$ accuracy with 358464 iterations in 26514s
Multi-threaded and distributed system speedup

- Figure plots the speedup over 4-threaded shared memory single machine
- On an TSP instance that can fit into memory
- Speedup on running one iteration of PDLP
Large Instance: Production Inventory Problem (Ben-Tal et al 2004)

- 50 factories, 100 time periods
- 2.1m variables, 5.2m constraints, 19m nonzeros
- Gurobi primal simplex solve to optimality with 531160 iterations in 11800s
- PDLP solves to $10^{-4}$ accuracy with 358464 iterations in 26514s
Visualization of PDHG, 1d, unconstrained

\[ \min_x \max_y L(x, y) = xy \]
[Lu, 2021] provides an ODE description on such behaviors.