New Theory and New Computational Methods for Improving the Effectiveness of First-Order Methods in Optimization

Grant name: New Theory and New Computational Methods for Improving the Effectiveness of First-Order Methods in Optimization

Grant Number: FA9550-22-1-0356

Senior personnel: Robert M. Freund and Cuong Ngoc Nguyen

Funding period: July 1, 2022 – June 30, 2025

Level-Set Geometry and Improving PDHG for Linear Optimization

Zikai Xiong and Robert Freund

(Note: Zikai is on the 2024-2025 academic job market)

Paper on arXiv





Zikai Xiong (MIT OR Center)

Robert Freund (MIT Sloan)

Linear Optimization ("LO" or "LP") is "Everywhere"

















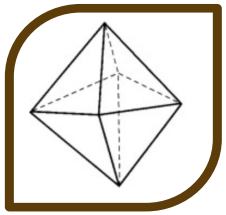
History of Linear Optimization ("LO" or "LP")

1947

Simplex Method

[George Dantzig, 1947]

75+ years ago

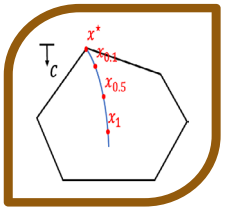


1984

Interior Point Method

[Narendra Karmarkar, 1984]

40 years ago



2024

Huge-scale Method(s)?



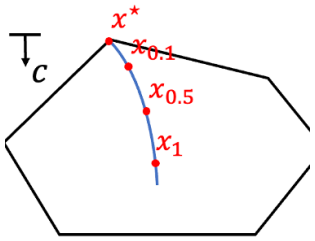


The next method(s) designed to address huge-scale LP

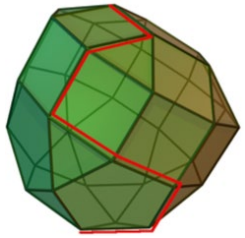
Recent Advances on Huge-Scale LP Solvers in the Industry

Classic methods

Interior-point method



Simplex method



First-order methods

- Solves huge-scale problems
- Benefits from modern computational architecture (such as GPU)

Google

COPT Cardinal Optimizer

State-of-art commercial optimization solver

NVIDIA®

The world's largest GPU company

FICO®

State-of-art commercial optimization solver

2021

Google OR-Tools proposed and implemented a distributed first-order LP method.
(2024 Beale-Orchard-Hays Prize)

Feb 12, 2024

COPT embedded GPU-based LP method

Mar 20, 2024

NVIDIA cuOpt announced a GPU-based LP solver

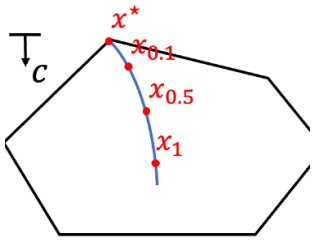
Apr 22, 2024

FICO Xpress embedded a matrix-factorization-free LP method

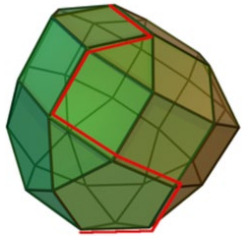
Recent Advances on Huge-Scale LP Solvers in the Industry

Classic methods

Interior-point method



Simplex method



First-order methods

Primal-Dual Hybrid Gradient
(aka PDHG, aka Chambolle-Pock method)

- Solves huge-scale problems
- Benefits from modern computational architecture (such as GPU)
- The method embedded in Google OR-Tools, COPT, and FICO Xpress

Google

COPT Cardinal Optimizer
State-of-art commercial optimization solver

NVIDIA
The world's largest GPU company

FICO
State-of-art commercial optimization solver

2021 **(PDHG)**

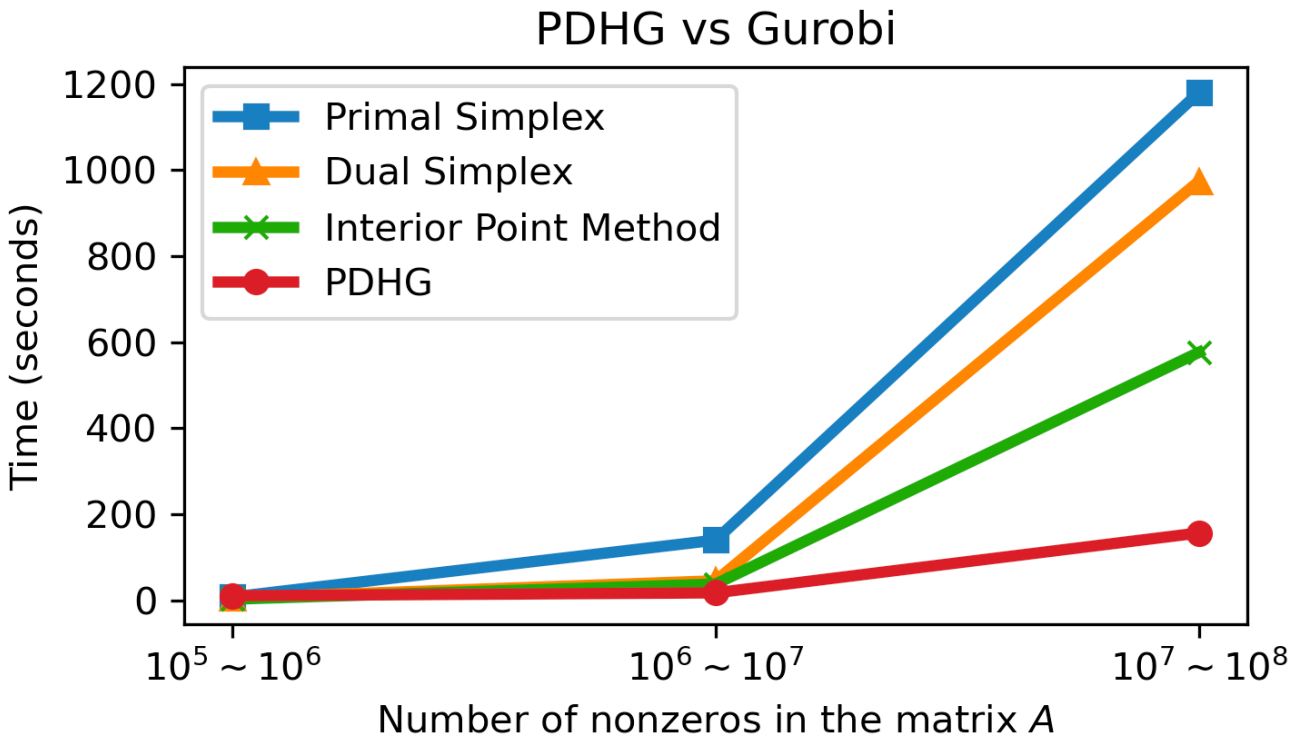
Feb 12, 2024 **(PDHG)**

Mar 20, 2024 **(Likely still PDHG)**

Apr 22, 2024 **(PDHG)**

PDHG is very good at solving Large LP Instances

Geometric average runtime on problems of different scale
from LP relaxations of MIPLIB 2017



Termination tolerance is 10^{-4}

Data from:
Lu, H., & Yang, J. (2023). cuPDLP. jl: A GPU implementation of restarted primal-dual hybrid gradient for linear programming in Julia. arXiv preprint arXiv:2311.12180.

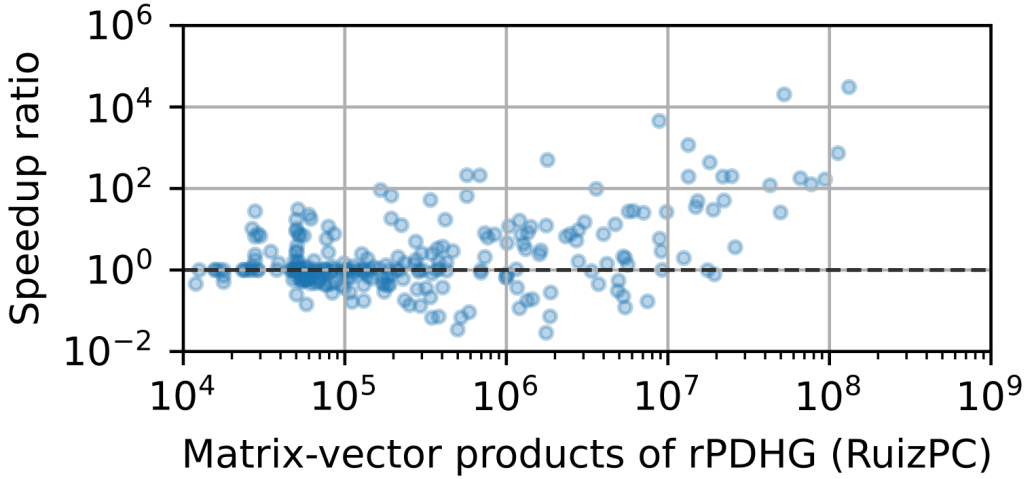
Huge-Scale LP Research

- SCS: Operator splitting/ADMM [O'Donoghue, Chu, Parikh, Boyd, 2016]
- ABIP+: ADMM-based interior-point method
  [Lin, Ma, Ye, Zhang, 2021] & [Deng, et al., 2022]
- Semi-smooth Newton augmented Lagrangian [Li, Sun, Toh, 2020]
- **Primal-Dual Hybrid Gradient (PDHG)** with restarts, applied directly to the primal-dual saddle point problem [Applegate, Hinder, Lu, Lubin, 2023] & [Applegate, et al., 2021] **(2024 Beale-Orchard-Hays Prize)**
- **GPU implementations** of PDHG in Julia and C
  [Lu and Yang, 2023] & [Lu, et al., 2023]
- **Guarantees for PDHG for LP** using "Limiting Error Ratios" and LP Sharpness [Xiong and Freund 2023]
- **Guarantees for PDHG for CLP** – using level-set geometry
  [Xiong and Freund 2024]

Sneak Preview:

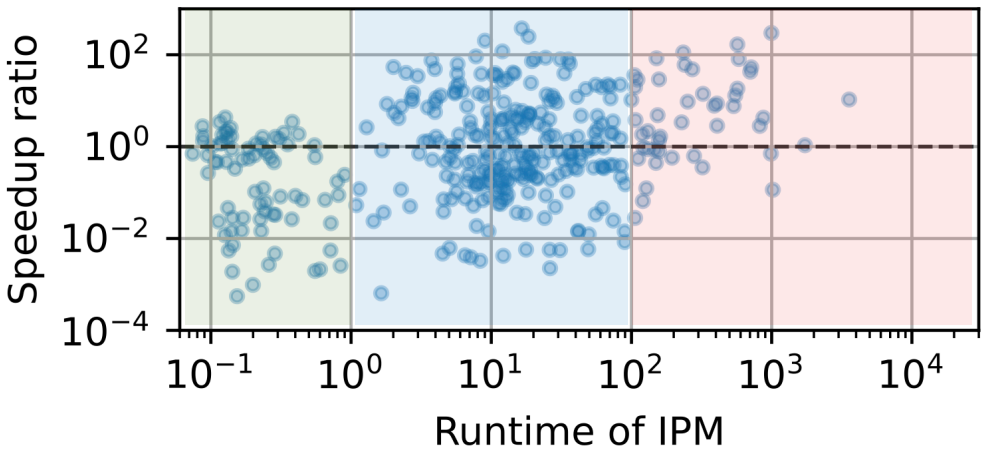
Distribution of Speedups of our method PDHG-AHR

Speedups compared with PDHG with PDLP's rescaling



Speedups compared with a "home-grown" IPM

(Predictor-corrector path-following interior-point method in Nocedal and Wright *Numerical Optimization* (2006))



Linear Optimization ("LO" or "LP")

LP in standard form

(primal)

$$\begin{aligned} \min\ & c^\top x \\ \text{s.t. } & Ax = b \\ & x \geq 0 \end{aligned}$$

(dual)

$$\begin{aligned} \max\ & b^\top y \\ \text{s.t. } & A^\top y \leq c \end{aligned}$$

Decision variables

- $x \in R^n$ (for primal problem)
- $y \in R^m$ (for dual problem)

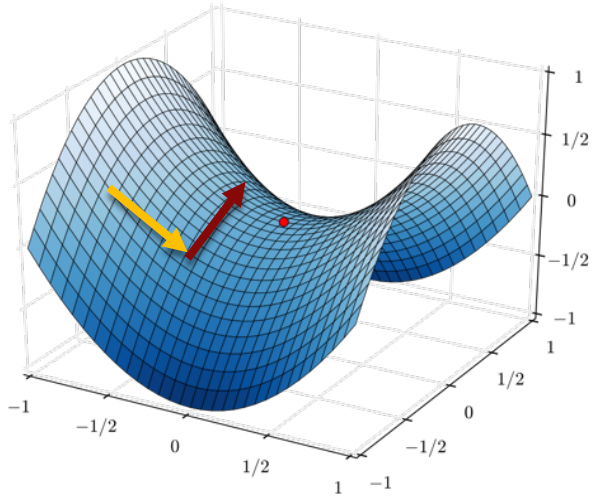
LP Saddlepoint formulation

$$\min_{x \geq 0} \max_{y}\ c^\top x - y^\top A x + b^\top y$$

Primal-Dual Hybrid Gradient Method (PDHG)

LP in Saddlepoint Problem Form

$$\min_{x \geq 0} \max_{y} \; c^\top x - y^\top A x + b^\top y$$



PDHG

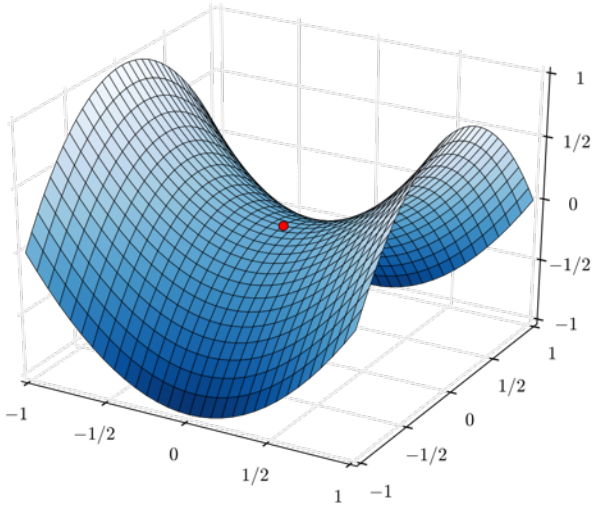
$$x^{k+1} \leftarrow \left(x^k - \tau(c - A^\top y^k)\right)^+$$

$$y^{k+1} \leftarrow y^k + \sigma\left(b - A(2x^{k+1} - x^k)\right)$$

Primal-Dual Hybrid Gradient for LP

PDHG

$$x^{k+1} \leftarrow \left(x^k - \tau(c - A^\top y^k)\right)^+$$
$$y^{k+1} \leftarrow y^k + \sigma\left(b - A(2x^{k+1} - x^k)\right)$$



- **Inexpensive iterations**:
  Only requires matrix-vector multiplications
- **"Fast" convergence rates**:
  Adaptive restarts based on average iterates yield linear convergence
  [Applegate, Hinder, Lu, Lubin, 2023]

We use "PDHG" to denote the "PDHG with adaptive restarts"

Current theory for PDHG for LP

Pros:

- PDHG has global linear convergence for LP instances
- Seemingly good practical performance when implemented wisely

Cons:

- Current theoretical complexity is quite loose and hard to compute/validate

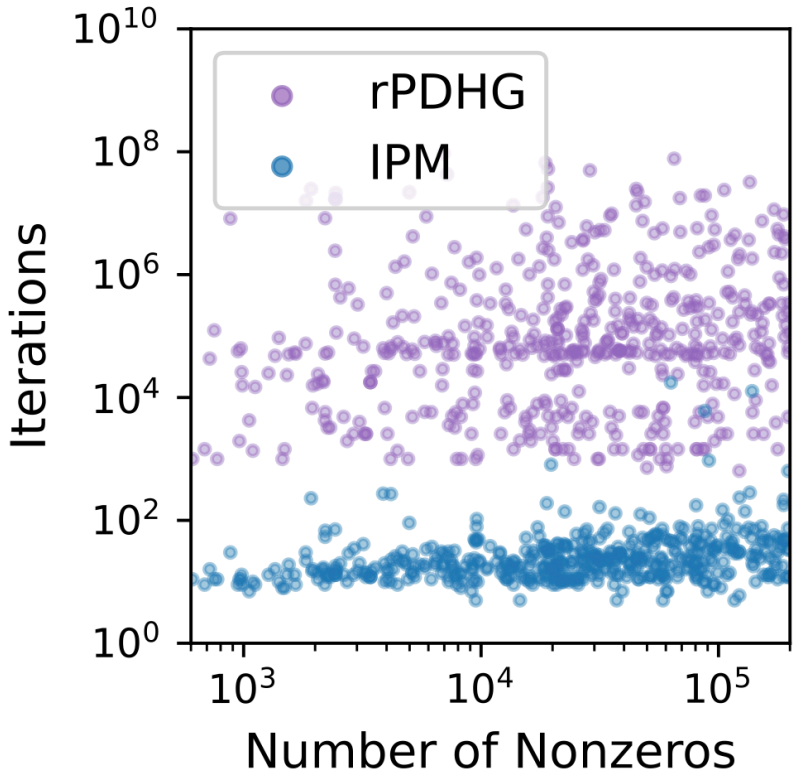
  We seek an iteration bound that is both tighter and easier to analyze

- Some seemingly "easy" problems are hard for rPDHG

  We seek to understand what properties of the "easy" problems make them so hard

Performance of PDHG

Iterations Required for
LP relaxations from MIPLIB 2017

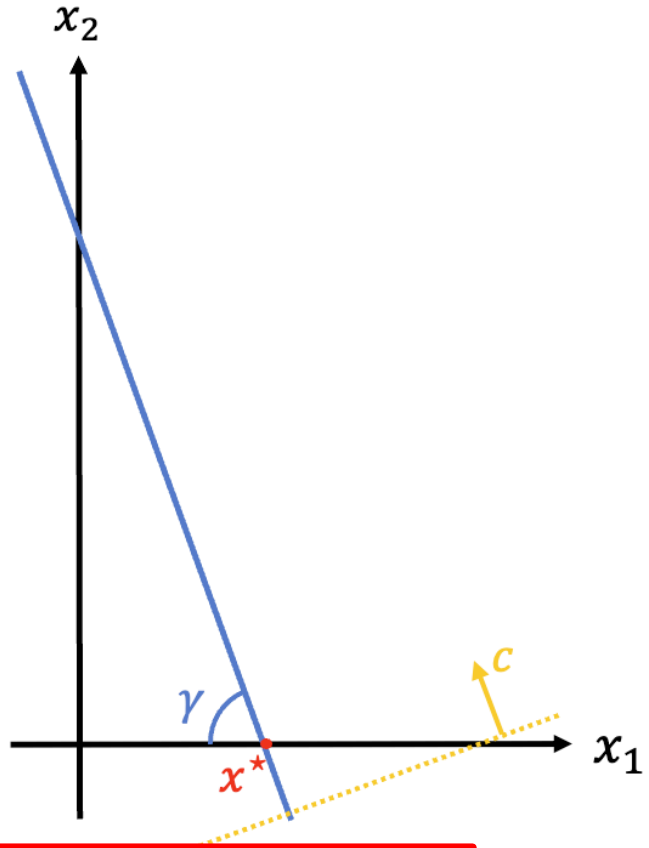


- PDHG uses more iterations than IPM
  makes sense, as a first-order method …
- Some small instances require a very large number of iterations
  a real challenge for PDHG

A seemingly easy LP instance

For $\gamma \in (0, \pi/2)$ define:

$$P(\gamma): \quad \min_{x_1, x_2} \; -\cos(\gamma)x_1 + \sin(\gamma)x_2$$
$$\text{s.t.} \; \sin(\gamma)x_1 + \cos(\gamma)x_2 = 1,$$
$$x_1 \geq 0, \; x_2 \geq 0$$



It is always easy for simplex method and interior-point method to solve $P(\gamma)$

However, when γ is small, PDHG requires at least 100,000 iterations.What **conditions** of $P(\gamma)$ make it so hard for PDHG?

P(γ) is hard to solve using PDHG



What condition numbers drive the performance of PDHG?

Can we improve these condition numbers and so improve computational performance in theory/practice?

When γ is small, PDHG requires at least 100,000 iterations.
What **conditions** of P(γ) make it so hard for PDHG?

Condition numbers describe the state/condition of the problem

Condition numbers related to Level-set Geometry

Primal-Dual Reformulation

Primal

$$\min_{x} c^\top x \quad \text{s.t. } Ax = b,\ x \geq 0$$

Dual

$$\max_{y} b^\top y \quad \text{s.t. } A^\top y \leq c$$

Assumptions:
(not necessary, easy for analysis)

1. Rows of A are linearly independent
2. Strict feasibility
3. $Ac = 0$

$$\max_{y,s} b^\top y \quad \text{s.t. } A^\top y + s = c,\ s \geq 0$$

$$b^\top (AA^\top)^{-1} A(c - s)$$

$$y = (AA^\top)^{-1} A(c - s)$$

Primal-Dual Reformulation

Primal	Dual
$\min_{x} c^\top x$ s.t. $Ax = b$ $x \geq 0$	$\max_{s} b^\top (AA^\top)^{-1} A(c - s)$ s.t. $s \in c + \mathrm{Im}(A^\top)$ $s \geq 0$

Duality gap: $\mathrm{Gap}(x, s) = c^\top x - b^\top (AA^\top)^{-1} A(c - s)$

The "primal-dual slack-space LP is then:

$$\min_{w := (x,s)} \ \mathrm{Gap}(w)$$
$$\text{s.t.} \quad w := (x, s) \ \text{is primal/dual feasible}$$

This problem has a known optimal objective value $= 0$.

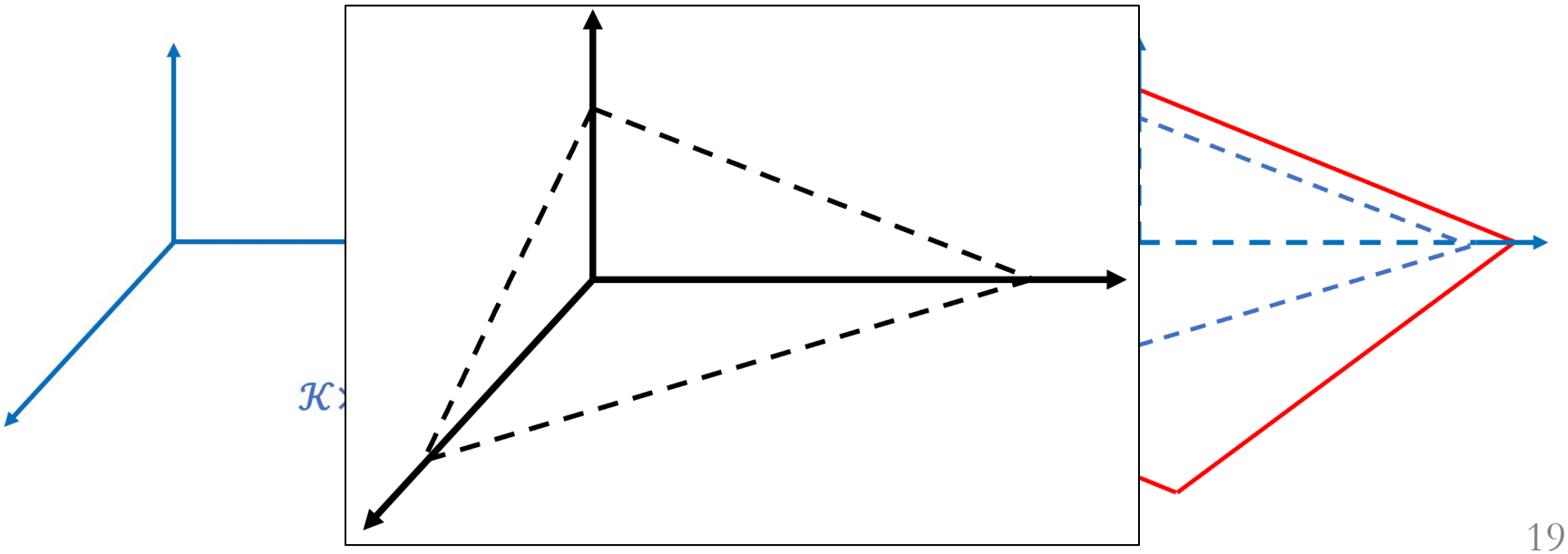
The feasible region in the space of (x, s)

$$\min_{x} \; c^\top x \quad \text{s.t.} \quad Ax = b, \; x \geq 0$$

$$\max_{s} \; b^\top (AA^\top)^{-1} A(c - s) \quad \text{s.t.} \quad s \in c + \mathrm{Im}(A^\top), \; s \geq 0$$

(x, s) in the nonnegative orthant
(a primal-dual cone for conic LP)

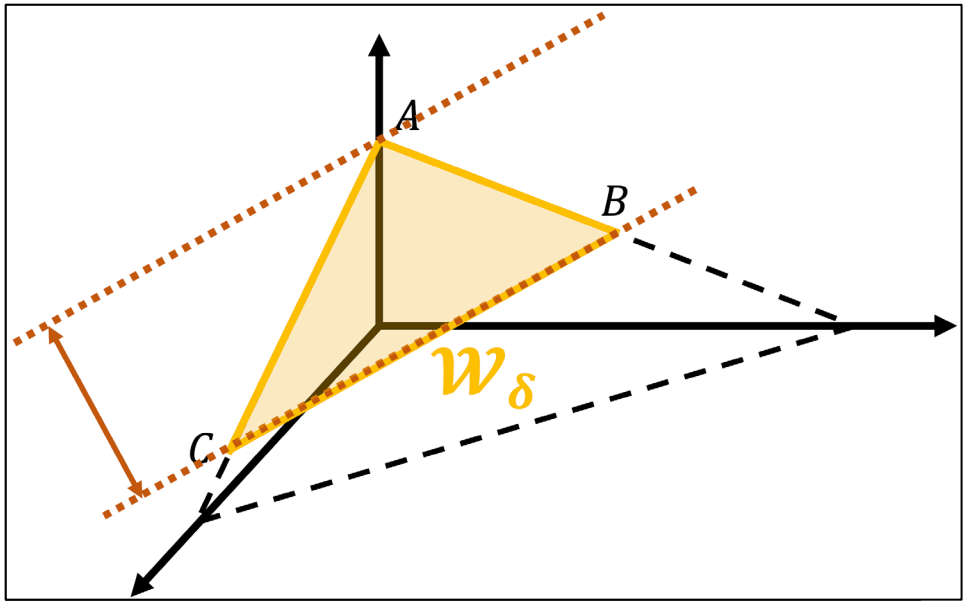
(x, s) lies in an affine subspace



Primal-Dual Sublevel Sets

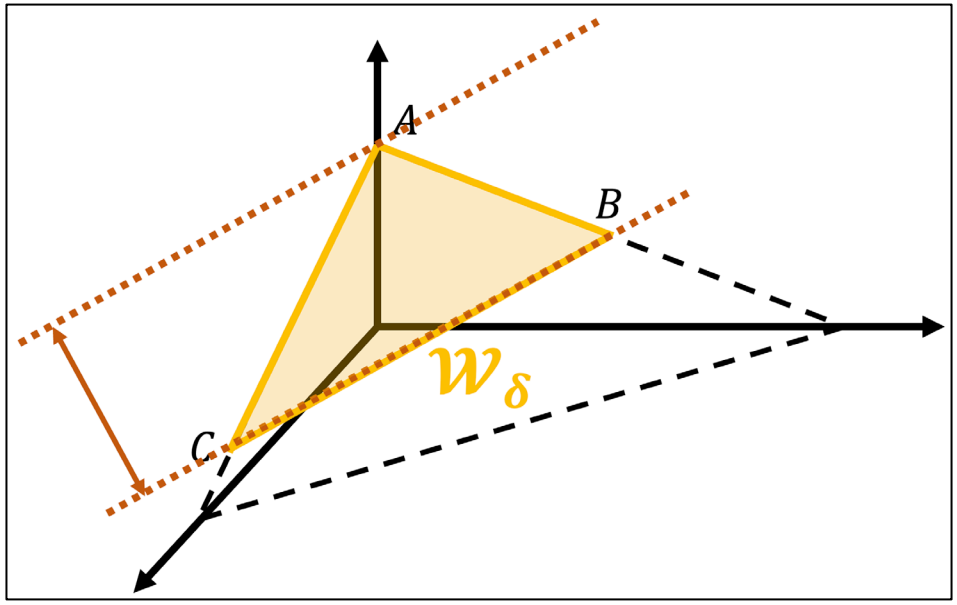
$$\mathcal{W}_\delta := \left\{ w := (x, s) \;\middle|\; \begin{array}{c} w \text{ is primal/dual feasible} \\ \mathbf{Gap}(w) \leq \delta \end{array} \right\}$$

Note: $\mathcal{W}_0 = \mathcal{W}^\star$



Three Geometric Condition Numbers

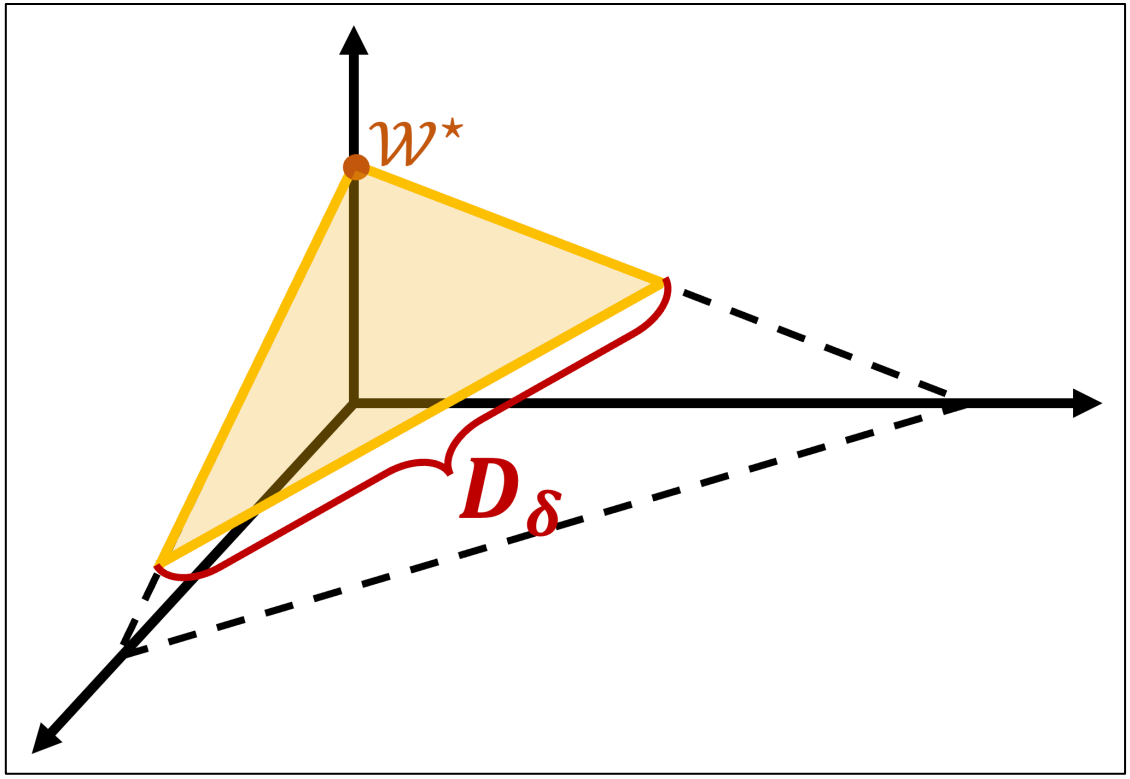
1. D_δ -- the diameter of $\mathcal{W}_\delta$
2. r_δ -- conic radius of $\mathcal{W}_\delta$
3. d_δ^H -- Haussdorf distance between $\mathcal{W}_\delta$ and $\mathcal{W}^\star$



D_δ: Diameter of δ-sublevel set $\mathcal{W}_\delta$

$$D_\delta := \max_{\bar{w}, \hat{w} \in \mathcal{W}_\delta} \|\bar{w} - \hat{w}\|$$

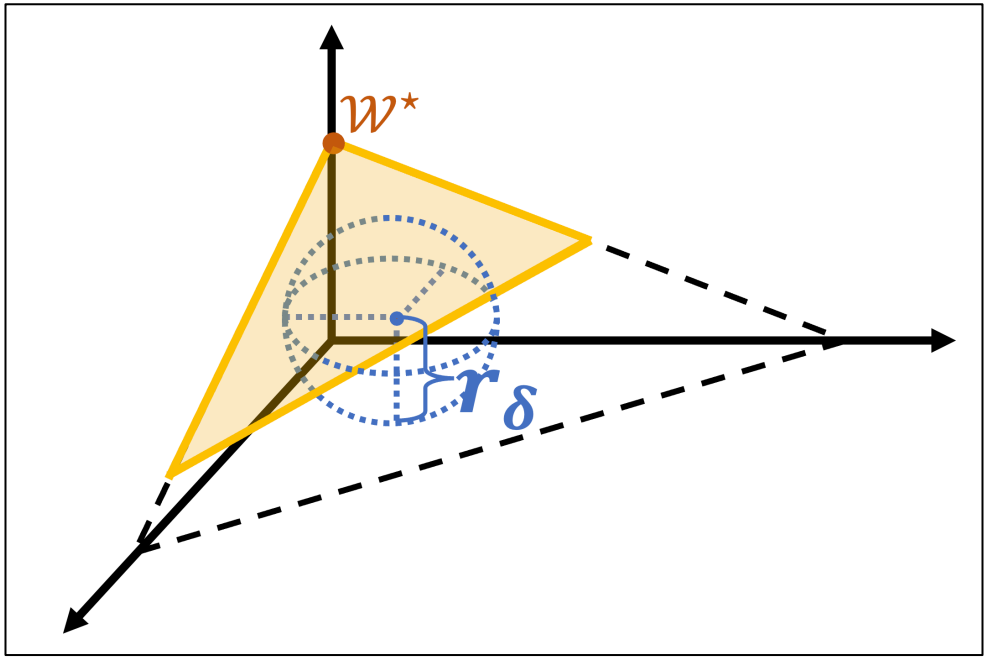
(D_δ is smaller when δ is small)



r_δ: "Conic Radius" of $\mathcal{W}_\delta$

$$r_\delta := \max_{r \geq 0,\, w \in \mathcal{W}_\delta} r \quad \text{s.t.} \quad B_w(r) \subset R_+^{2n}$$

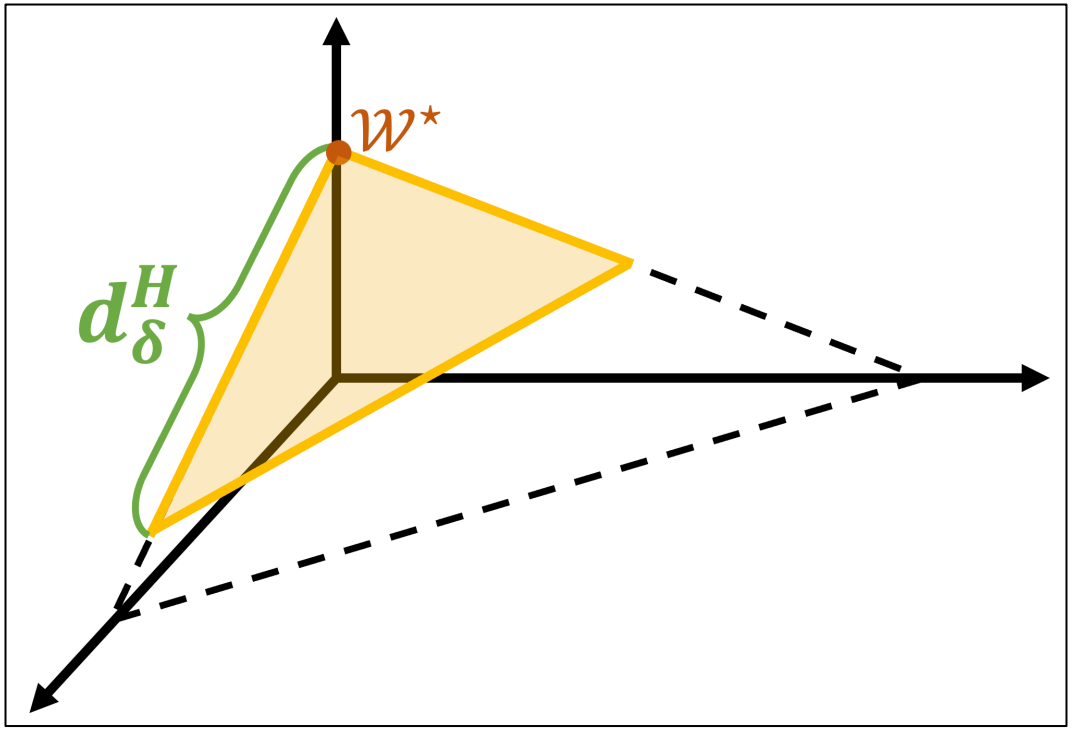
r_δ is the radius of the maximum ball inscribed in R_+^{2n} and centered at a point in $\mathcal{W}_\delta$



d_δ^H: Hausdorff distance from $\mathcal{W}_\delta$ to $\mathcal{W}^\star$

$$d_\delta^H := \max_{w \in \mathcal{W}_\delta} \mathrm{Dist}(w, \mathcal{W}^\star)$$

d_δ^H is small when δ is small



Convergence Guarantee for rPDHG

A "Composite" Computational Guarantee

Theorem: Starting from $(x^{0,0}, y^{0,0}) = (0,0)$, the number of PDHG iterations required to compute an $\boldsymbol{\varepsilon}$-optimal solution is upper bounded by:

$$\tilde{O}\left(\inf_{\delta>0} T_\delta := \boldsymbol{\kappa} \cdot \left[\frac{\boldsymbol{D_\delta}}{\boldsymbol{r_\delta}} \cdot \ln\left(\frac{1}{\boldsymbol{\varepsilon}}\right) + \boldsymbol{d_\delta^H} \cdot \frac{1 + \mathrm{Dist}(0, \mathcal{W}^\star)}{\boldsymbol{\varepsilon}}\right]\right)$$

Corollary: If there exists $\delta > 0$ whose δ-sublevel set satisfies:

$\frac{D_\delta}{r_\delta}$ is small and d_δ^H is small,

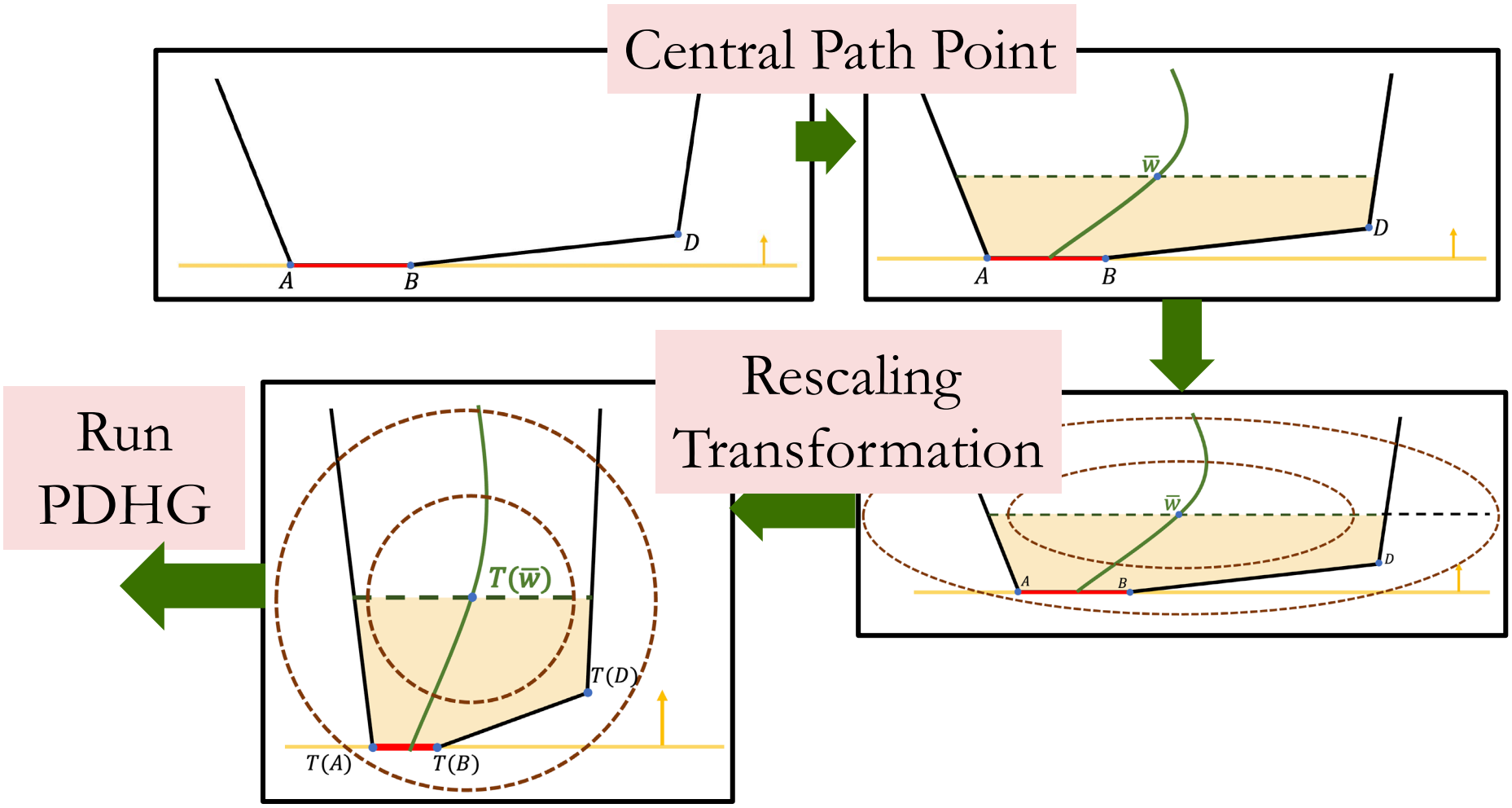


then rPDHG will converge faster. (If not, rPDHG might be slow…)

$\boldsymbol{D_\delta}$: Diameter of …
$\boldsymbol{r_\delta}$: Conic radius of …
$\boldsymbol{d_\delta^H}$: Hausdorff distance …

Using theory to develop practical computational speed-ups of PDHG

The idea


Central Path Point
A
B
D
$\bar{w}$
Rescaling
Transformation
Run
PDHG
$T(\bar{w})$
$T(A)$
$T(B)$
$T(D)$

Central-Path solutions are good interior-points

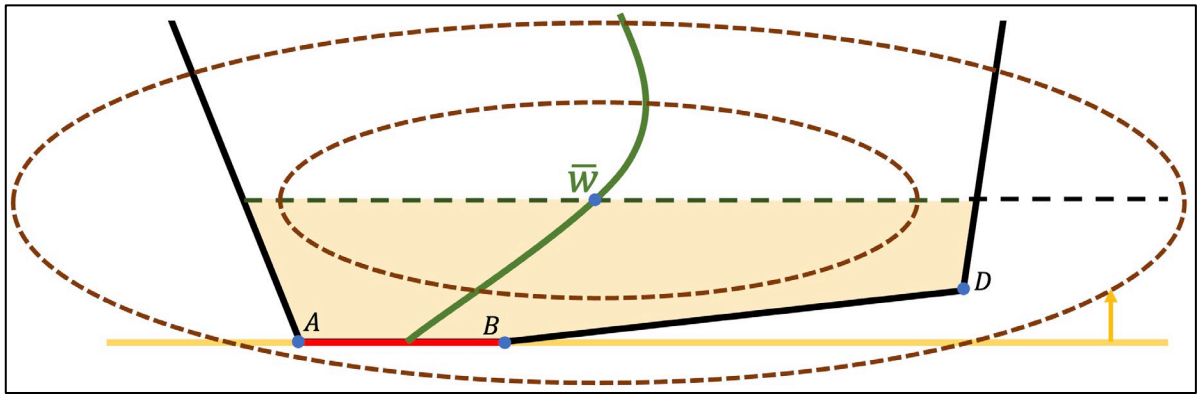
Barrier function: $F(x) \coloneqq -\sum_i \ln(x_i)$

Central path $\{w_\eta \coloneqq (x_\eta, s_\eta)\}$ are solutions of barrier problems:

$$x_\eta = \arg\min_x c^\top x + \eta \cdot F(x) \quad \text{s.t.} \quad Ax = b, x \geq 0$$

$$s_\eta = \arg\max_{y,s} b^\top y + \eta \cdot F(s) \quad \text{s.t.} \quad A^\top y + s = c, s \geq 0$$

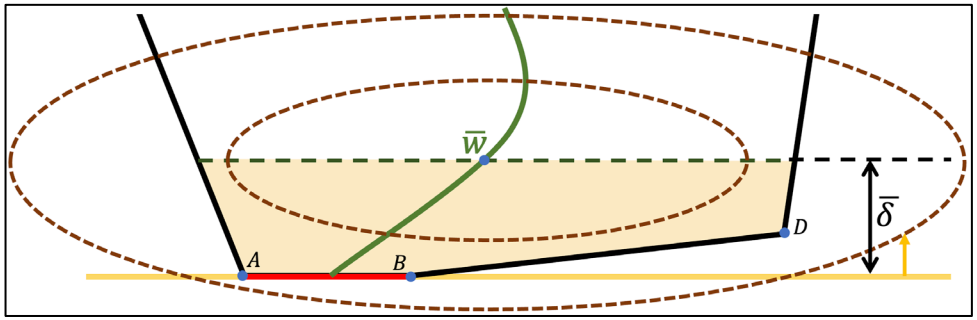
When $\bar{w} = w_\eta = (x_\eta , s_\eta)$, the ball of the "local norm" (matrix norm of the barrier function Hessian) nicely approximates the shape of the sublevel set:



Transformation based on a central-path solution

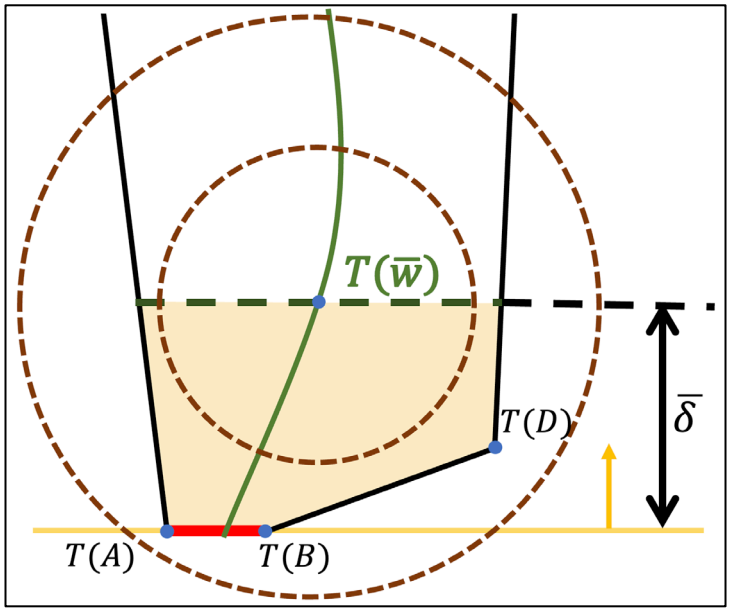
Let $H := \mu \cdot \nabla^2 F(x_\mu)$, then

- $\bar{\delta} := n\mu = \text{Gap}(\bar{w})$
- $D_{\bar{\delta}} \le 2n \cdot \sigma^+_{\min}(H)^{-1/2}$
- $r_{\bar{\delta}} \ge \sigma^+_{\max}(H)^{-1/2}$
- $d^H_{\bar{\delta}} \le 2n \cdot \sigma^+_{\min}(H)^{-1/2}$



After a rescaling transformation (turns the local-norm ball into a Euclidean norm ball) we have:

- $D_{\bar{\delta}} \le 2\sqrt{n} \cdot \sqrt{\bar{\delta}}$
- $r_{\bar{\delta}} \ge \sqrt{\frac{1}{n}} \cdot \sqrt{\bar{\delta}}$
- $d^H_{\bar{\delta}} \le 2\sqrt{n} \cdot \sqrt{\bar{\delta}}$
- $\frac{D_{\bar{\delta}}}{r_{\bar{\delta}}} \le 2n$
- $d^H_{\bar{\delta}}$ is small if $\bar{\delta}$ is small
- "Very nice theory"



Complexity under good linear transformation

Suppose we do the following:

1. rescaling transformation using a central-path solution **with duality gap** $\bar{\delta}$
2. row transformation to try to decrease closer to $\kappa = 1$

Then the number of PDHG iterations required to compute an ε-optimal solution of the original problem is upper bounded by:

$$\tilde{O}\left(n \cdot \left(\ln\left(\frac{1}{\varepsilon}\right) + \frac{D_{\bar{\delta}} + \bar{\delta}}{\varepsilon}\right)\right)$$

Suppose we do the following:

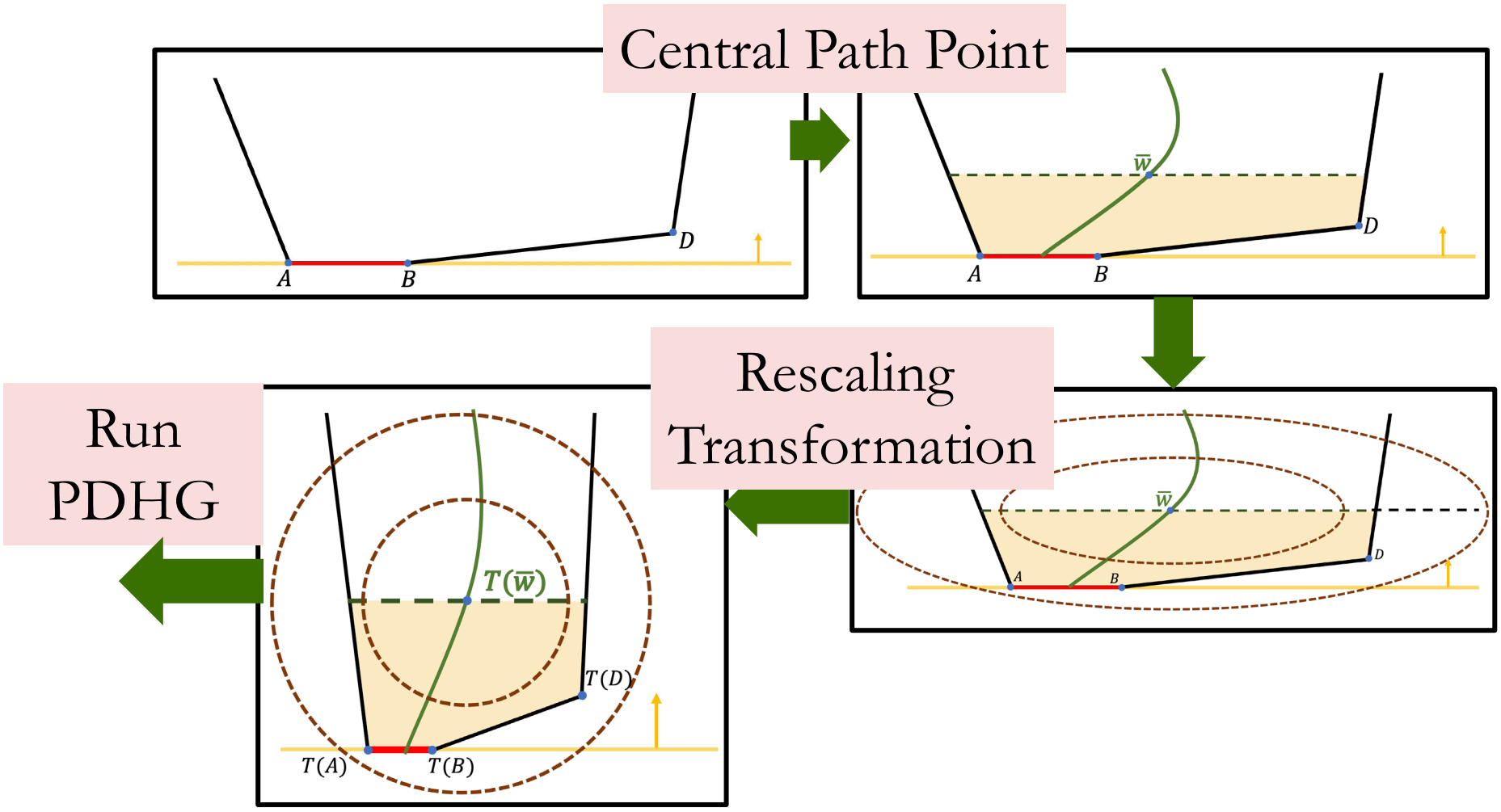
1. rescaling transformation using a central-path solution **with duality gap** $\bar{\delta}$
2. row transformation to try to decrease closer to $\kappa = 1$

Then the number of PDHG iterations required to compute an ε-optimal solution of the original problem is upper bounded by:

$$\tilde{O}\left(n \cdot \left(\ln\left(\frac{1}{\varepsilon}\right) + \frac{D_{\bar{\delta}} + \bar{\delta}}{\varepsilon}\right)\right)$$

- We have replaced $\frac{D_\delta}{r_\delta}$ by n
- The smaller $\bar{\delta}$ is, the faster the convergence
- Similar results hold for conic LPs
  (for example, n becomes 2 for second-order cone programs)
- Of course we will need to "pay" to compute the point on the central path…

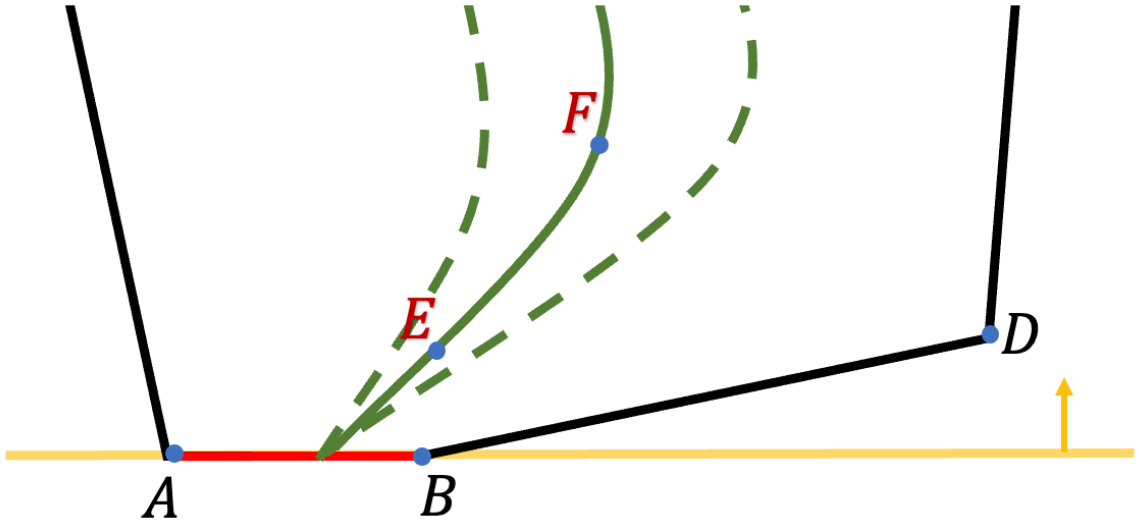
The idea, again



rPDHG-AHR ("Adaptive Hessian Rescaling") and Computational Experiments

Main Strategy

Main strategy: use a "CG-IPM" to compute a low-accuracy central-path solution to obtain a good rescaling, and then use rPDHG

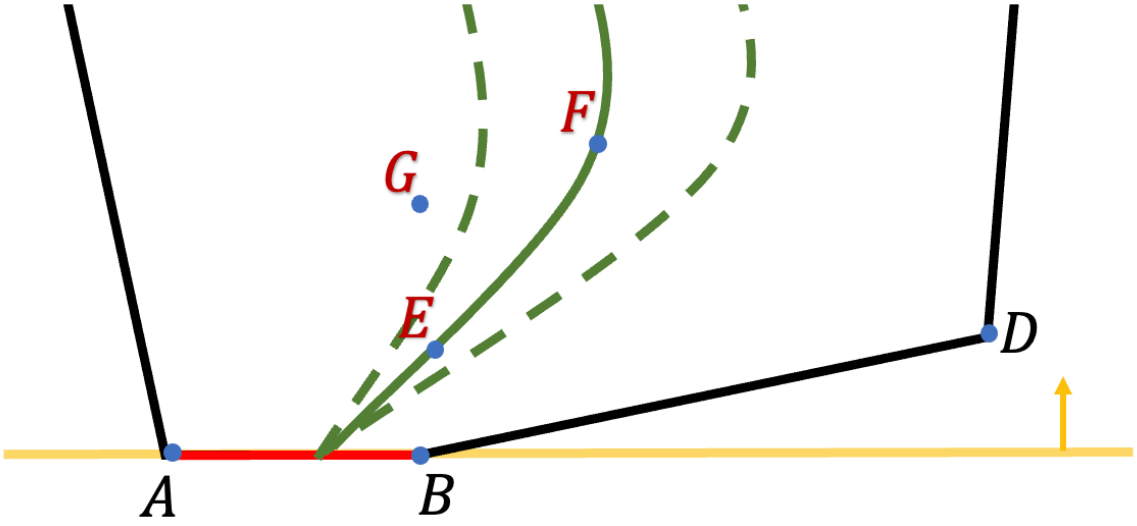


Main Strategy, continued

Main strategy: use a "CG-IPM" to compute a low-accuracy central-path solution to obtain a good rescaling, and then use rPDHG

- **We use a first-order method to compute a "central path" solution**
  - we use a conjugate-gradient-method-based IPM (**CG-IPM**)
  - Implementation follows Nocedal and Wright *Numerical Optimization* (2006)
  - but we solve the normal equations using conjugate gradient method
- **We employ only diagonal row rescaling to try to improve κ**
  - Column rescaling uses the central-path Hessian of w_{int} followed by PDLP's rescaling (Ruiz rescaling and Pock-Chambolle rescaling), which we call the "w_{int}-rescaled problem"

Using Adaptive Rescaling



Motivating concepts of Adaptive Rescaling:

- Adaptively balance the cost of computing the rescaling (CG-IPM) with the savings from running rPDHG on the rescaled instance
- Try to identify a "good-enough" rescaling as early as possible and use the good-enough rescaling
- Seek to avoid unnecessary extra computation to get an "even better" rescaling when the improvement is not cost-effective

Adaptive rescaling heuristic

rPDHG with Adaptive central-path Hessian Rescaling : **rPDHG−AHR**

Algorithm 4: Scheme for rPDHG with Adaptive Hessian Rescaling (rPDHG-AHR)

Input: Initial iterate $z^0 := (x^{0,0}, y^{0,0})$, initial point w^0 for CP-CGM, time parameter t, time multiplier ω, target relative error ε, and adaptivity error parameters $\hat{\varepsilon}$ and $\bar{\varepsilon}$. Define $k := 0$, $\varepsilon_0 := +\infty$;

repeat

  Run (or continue running) CP-CGM from w^k for t seconds. Output w^{k+1} ;

  Construct new rescaled problem: define $\eta^{k+1} := (\mathsf{s}^{k+1})^\top \mathsf{x}^{k+1}$, $\tilde{D}_1 := \sqrt{\eta^{k+1}} H^{-1/2}_{\mathsf{x}^{k+1}}$ and $\tilde{D}_2 = I$. Optionally do further rescaling by introducing $\bar{D}_1$ and $\bar{D}_2$, and setting $D_1 := \bar{D}_1\tilde{D}_1$ and $D_2 := \bar{D}_2\tilde{D}_2$. Then construct the new rescaled problem $(\tilde{\mathrm{P}})_{k+1}$ using D_1 and D_2 ;

  Run rPDHG on rescaled problem $(\tilde{\mathrm{P}})_{k+1}$ for ωt seconds. Output the transformed solution z^{k+1} and the relative error $\varepsilon_{k+1} := \mathrm{MErr}_\varepsilon(x^{k+1}, y^{k+1})$;

  $k \leftarrow k+1$ and $t \leftarrow 2t$;

until *either (a)* $\varepsilon_k \le \bar{\varepsilon}$, *or (b)* $\varepsilon_k > \varepsilon_{k-1}$ *and* $\varepsilon_{k-1} \le \hat{\varepsilon}$;

If (a) holds, then fix the new rescaling: run rPDHG on $(\tilde{\mathrm{P}})_k$ until a solution $z = (x, y)$ is computed for which $\mathrm{MErr}_\varepsilon(x, y) \le \varepsilon$;

If (b) holds, then revert to and fix the previous rescaling: run rPDHG on $(\tilde{\mathrm{P}})_{k-1}$ until a solution $z = (x, y)$ is computed for which $\mathrm{MErr}_\varepsilon(x, y) \le \varepsilon$;

Computational Experiments with rPDHG-AHR

Algorithm 4: Scheme for rPDHG with Adaptive Hessian Rescaling (rPDHG-AHR)

Input: Initial iterate $z^0 := (x^{0,0}, y^{0,0})$, initial point w^0 for CP-CGM, time parameter t, time multiplier ω, target relative error ε, and adaptivity error parameters $\hat{\varepsilon}$ and $\bar{\varepsilon}$. Define $k := 0$, $\varepsilon_0 := +\infty$;

repeat

  Run (or continue running) CP-CGM from w^k for t seconds. Output w^{k+1} ;

  Construct new rescaled problem: define $\eta^{k+1} := (\mathsf{s}^{k+1})^\top \mathsf{x}^{k+1}$, $\tilde{D}_1 := \sqrt{\eta^{k+1}} H_{\mathsf{x}^{k+1}}^{-1/2}$ and $\tilde{D}_2 = I$. Optionally do further rescaling by introducing $\bar{D}_1$ and $\bar{D}_2$, and setting $D_1 := \bar{D}_1 \tilde{D}_1$ and $D_2 := \bar{D}_2 \tilde{D}_2$. Then construct the new rescaled problem $(\tilde{\mathrm{P}})_{k+1}$ using D_1 and D_2 ;

  Run rPDHG on rescaled problem $(\tilde{\mathrm{P}})_{k+1}$ for ωt seconds. Output the transformed solution z^{k+1} and the relative error $\varepsilon_{k+1} := \mathrm{MErr}_\varepsilon(x^{k+1}, y^{k+1})$;

  $k \leftarrow k+1$ and $t \leftarrow 2t$;

until *either (a)* $\varepsilon_k \le \bar{\varepsilon}$, *or (b)* $\varepsilon_k > \varepsilon_{k-1}$ *and* $\varepsilon_{k-1} \le \hat{\varepsilon}$;

If (a) holds, then fix the new rescaling: run rPDHG on $(\tilde{\mathrm{P}})_k$ until a solution $z = (x, y)$ is computed for which $\mathrm{MErr}_\varepsilon(x, y) \le \varepsilon$;

If (b) holds, then revert to and fix the previous rescaling: run rPDHG on $(\tilde{\mathrm{P}})_{k-1}$ until a solution $z = (x, y)$ is computed for which $\mathrm{MErr}_\varepsilon(x, y) \le \varepsilon$;

We compare:

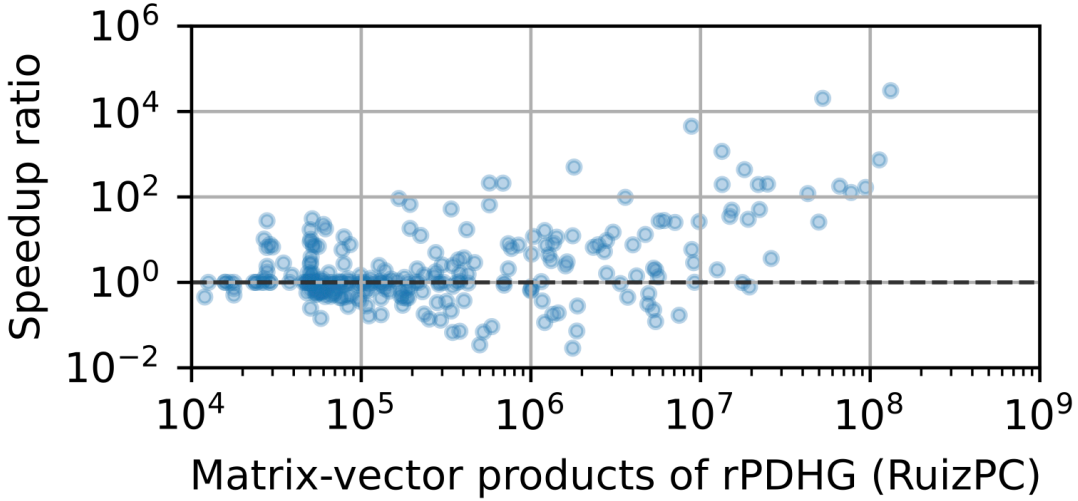
1. **rPDHG-AHR** : rPDHG with Adaptive central-path Hessian Rescaling (using CG-IPM for the central-path computations)
2. **rPDHG-RuizPC** : use heuristic Ruiz rescaling on ***A***, followed by Pock-Chambolle rescaling. (This is the same as the rescaling used in PDLP.)
3. **IPM** : a home-grown standard primal-dual predictor-corrector interior point method, straight from Nocedal and Wright *Numerical Optimization* (2006).

Test set: 417 LP relaxations from MIPLIB 2017 dataset

Computational Comparison: rPDHG-AHR and rPDHG(RuizPC)

Speedup Ratio:

$$\frac{\text{Matrix−vector products } \textbf{rPDHG(RuizPC)}}{\text{Matrix−vector products } \textbf{rPDHG−AHR}}$$



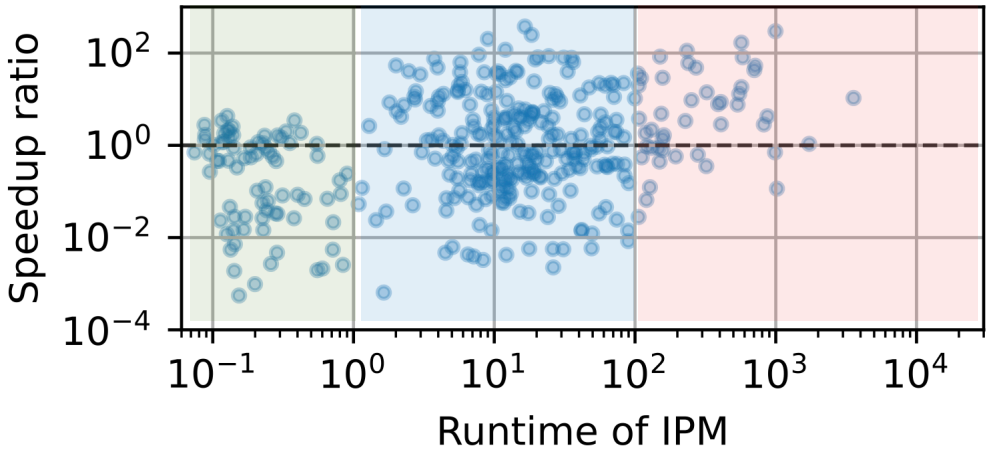
		Fraction solved by **rPDHG-AHR**	
		S	**N**
Fraction solved by **rPDHG (RuizPC)**	**S**	**87.7%**	**1.7%**
	N	**7.5%**	**3.1%**

- In general, the harder the problem is for rPDHG(RuizPC), the larger the speedups from using rPDHG-AHR
- rPDHG-AHR solves about 95.2% of problems and is more reliable than rPDHG(RuizPC) (solves about 89.4% of problems)

Computational Comparison: rPDHG-AHR and IPM

Speedup Ratio:

$$\frac{\text{Runtime } \textbf{IPM}}{\text{Runtime } \textbf{rPDHG-AHR}}$$



		Fraction solved by **rPDHG−AHR**	
		S	**N**
Fraction solved by **IPM**	**S**	**93.0%**	**1.5%**
	N	**2.2%**	**3.4%**

- Generally speaking, the harder the problem is for IPM, the larger the speedups from using rPDHG-AHR.
- rPDHG-AHR is slightly more reliable than IPM

Recap, Takeaways, and Remarks

Recap and takeaways:

- The convergence rate of PDHG is related to the geometry of primal-dual sublevel sets measured with D_δ, r_δ, d_δ^H
- Rescaling using a central-path solution can improve the geometry of the primal-dual sublevel sets
- Our strategy: use a "CG-IPM" to compute a low-accuracy central-path solution to obtain a good rescaling, and then use rPDHG
- FOMs can competitively compute solutions of the same high accuracy as IPMs

Remarks:

- Our results extend to general (convex) **conic linear programs**, in which the nonnegative cone becomes $\mathcal{K} \times \mathcal{K}^*$.
- Our results relied only on PDHG's average iterate convergence and non-expansiveness properties. Similar results might also hold for other FOMs, in particular ADMM, EGM, …

Thank you!