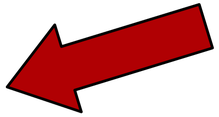
Fast Potential Reduction for LP and its Applications

SIOPT, MAY 31, 2023

Yinyu Ye
Stanford University and CUHKSZ (Sabbatical Leave)

Algorithms for Linear Programming

$$\begin{aligned} \min_{x} \quad & c^\top x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$



one of the most fundamental models in mathematical programming

A recent trend of research activities is to develop algorithms for solving LP that are dominated by first-order methods

- **Simplex method**
- **Interior point method**
- **First-order methods**

	Simplex	Interior point	First-order
Robustness	✓✓✓	✓✓	✓
Theory	✓	✓✓✓	✓✓
Accuracy	✓✓✓	✓✓✓	✓✓
Scalability	✓✓	✓	✓✓✓

Robust and highly accurate, but not that scalable if “dense”

First-order LP Algorithms: Recent Successes

First-order methods for LP

- **(Generally) based on gradient information integrated with many other techniques such as scaling, restart, parameter-tuning…**
- **Free of matrix factorization (each iteration)**
- **Able to achieve medium accuracy**

Solver	Accuracy
ABIP/ABIP+	$10^{-4} \sim 10^{-6}$
SCS	$10^{-4} \sim 10^{-6}$
PDLP	$10^{-6} \sim 10^{-8}$

- **State of the art first-order solvers now solve LPs to medium/high relative accuracy**
- **Enough for certain applications**

Potential Reduction for Linear Programming

The potential function in linear programming:

$$\phi(x) = \rho \log(\mathcal{G}(x)) - \sum_{i=1}^{n} \log x_i$$

Optimality Centrality

- **Proved as a theoretical tool to establish polynomial complexity and implemented as a neighborhood-tuning-free method comparing with the path-following methods**
- **There are different variants of interior point methods reduce different potential functions under linear/affine constraints**
- **Manage an automatic balance between *optimality* G and centrality**

Directly reduce the potential function using the gradients?

Choose the Right Potential Function to Reduce

A desirable first-order friendly potential function should

- **Represent LP as an unconstrained problem**
- **Admit cheap gradient computation**
- ***Convergece with gradient-based optimization***

We choose the potential from the homogeneous self-dual model

$$\mathcal{G}(x, y, s, \tau, \kappa) = \frac{1}{2}\underbrace{\|Ax - b\tau\|^2}_{\text{Primal infeas}} + \frac{1}{2}\underbrace{\|A^\top y + s - c\tau\|^2}_{\text{Dual infeas}} + \frac{1}{2}\underbrace{(b^\top y - c^\top x - \kappa)^2}_{\text{Complementarity}}$$

$$\phi = \rho \log(\mathcal{G}) - \sum_{i=1}^{n} \log x_i s_i - \log \kappa\tau$$

$$\begin{aligned} \min_{x,y,s,\kappa,\tau} \quad & \phi(x, y, s, \tau, \kappa) \\ \text{subject to} \quad & e^\top(x; s; \kappa; \tau) = 1 \end{aligned}$$

Reducing the Potential Function

The LP is represented by an almost unconstrained problem

$$\min_{x} \quad \rho \log(\mathcal{G}(x)) - \sum_{i=1}^{n} \log x_i$$

$$\text{subject to} \qquad e^{\top} x = 1$$

One may apply

- **first-order interior point trust region**
- **dimension-reduced second-order interior point trust region**
- **second-order potential reduction method**

A Briefing of the Algorithms

First-order interior point trust region

$$\begin{aligned} \min_{x} \quad & \nabla\phi(x^k)^\top(x-x^k) \\ \text{subject to} \quad & e^\top(x-x^k)=0 \\ & \|(X^k)^{-1}(x-x^k)\| \le \beta \end{aligned}$$

Dimension-reduced second-order trust region

$$\begin{aligned} \min_{x} \quad & \nabla\phi(x^k)^\top(x-x^k)+\frac{1}{2}(x-x^k)^\top\nabla^2\phi(x^k)(x-x^k) \\ \text{subject to} \quad & e^\top(x-x^k)=0 \\ & x-x^k \in \text{span}\{\nabla\phi(x^k), X^k\nabla\phi(x^k), x^k-x^{k-1}\} \\ & \|(X^k)^{-1}(x-x^k)\| \le \beta \end{aligned}$$

Second-order potential reduction

$$\begin{pmatrix} & A & & -b & \\ -A^\top & & -I & c & \\ b^\top & -c & & & -1 \\ & & & \kappa & \tau \\ & S & X & & \end{pmatrix}\begin{pmatrix} \Delta_d \\ \Delta_p \\ \Delta_{\kappa,\tau} \end{pmatrix}=\begin{pmatrix} -r_p \\ -r_d \\ -r_{\kappa,\tau} \end{pmatrix}$$

- Three methods reduce the same potential function
- Optimization can switch between them seamlessly
- First-order method can either do warm-starting or serve as a rescue when second order method fails

Theoretical Results (Gao at al. SHUFE, 2023)

Theorem 1. Let $\Delta^k = \phi(x^{k+1}) - \phi(x^k)$,

$$\Delta^k \leq \begin{cases} -\beta + \dfrac{\rho M}{2\mathcal{G}}\beta^2 + \dfrac{\beta^2}{2(1-\beta)}, & \text{with first-order potential reduction} \\ -\beta + \dfrac{3\rho M}{2\mathcal{G}}\beta^2 + \dfrac{\beta^2}{2(1-\beta)}, & \text{with dimension-reduced trust-region} \\ -\dfrac{\sqrt{3}}{2}\beta + \dfrac{\beta^2}{2(1-\beta)}, & \text{with second-order potential reduction} \end{cases}$$

- **Linear convergence for second-order method**
- **Sub-linear convergence for first-order method**

Numerical Experiments: MIPLIB instances

- **Tested on 971 MIPLIB instances for 1e-04 tolerance**

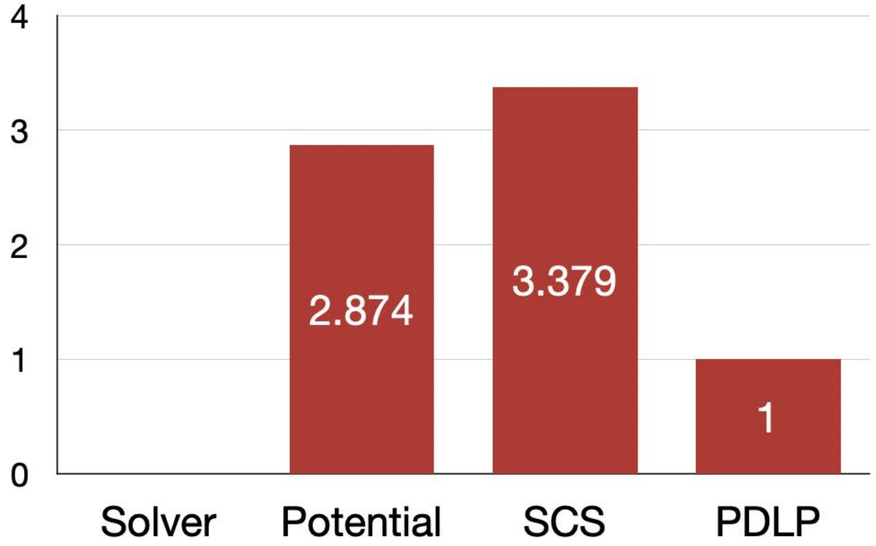
  Timelimit set to 600 seconds

- **Potential reduction switches to second-order method after reaching 1e-03 accuracy**

- **Tested different combinations of first/second order methods (0 step & 5 steps & 15 steps)**

- **First-order method alone achieves low-accuracy**

- **Second-order method follow up to finish solving**



Solver setting	Solved	SGM
Potential First-order	639	19.36
Potential First + Second order (5)	819	6.85
Potential First + Second order (15)	935	2.87
Second order	945	0.54
All	971	–

Why do we need Second-Order?

Necessary for Crossover: obtaining an optimal basis from the approximate solution

Numerical Experiments: Crossover

Often we need highly accurate or an optimal basic solution that is hardly achievable by first-order methods:

So we need to do crossover to locate an optimal basis from a "quality" solution

- **We crossover from both interior point solut**
- **We choose 114 Netlib datasets**

Solver	Time
Potential Interior Point	113.2
PDLP	222.5

s (300s)

ding the opt

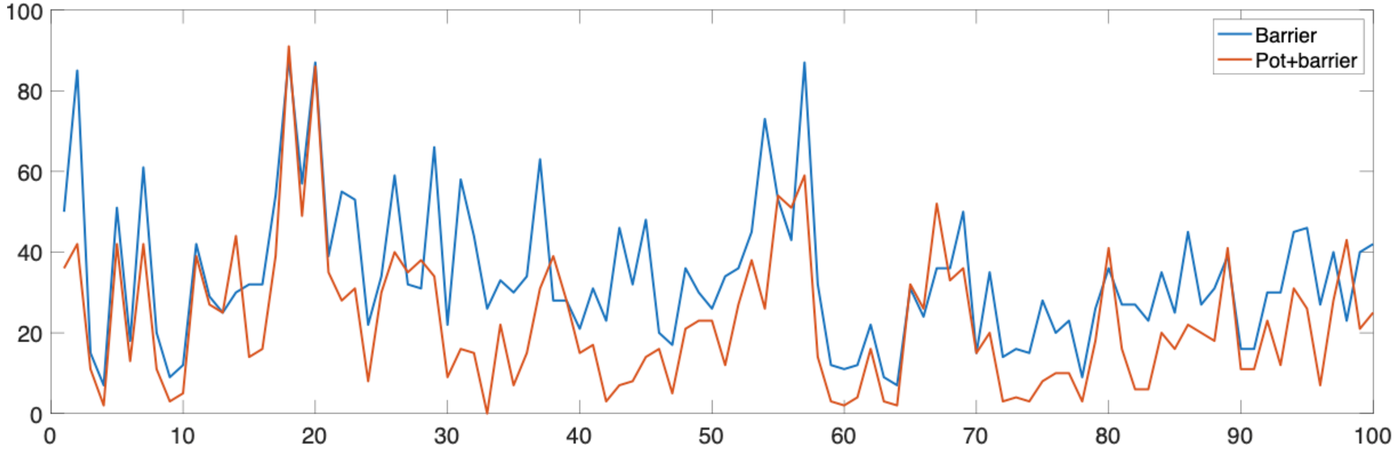
Solver	Potential Interior Point		PDLP	
Instance	10^{-6}	10^{-8}	10^{-6}	10^{-8}
df1001	3678/0.61s	4036/0.72s	21924/6.43s	26929/8.20s
pds-20	8413/0.42s	8647/0.39s	33543/13.19s	8861/0.37s
qap12	705/0.23s	698/0.23s	4801/2.53s	679/0.23s
qap15	3046/2.47s	3052/2.48s	54015/82.14s	2895/2.38

- **Interior point and PDLP solutions work comparatively on 70% of instances**
- **In general, interior point solutions tell more valueble information for crossover:** ***for some instances where b or c has large norm, solutions generated by PDLP fails to provide an efficient start for crossover***

It can Viewed as a Cheaper Warm-Starting Presolver

Accuracy	1e-04	1e-06	1e-08	1e-10
First-order	7.5	798.0	>1200	>1200
Second-order	33.0	56.7	89.3	93.3
First + Second	5.4	12.1	14.1	15.2

Example: Speedup on QAP instances



Iteration count on Netlib instances

- First-order method solves to 1e-02 accuracy and then switch to second-order
- An average reduction of 30% iterations compared to trivial start
- There are instances where first + second is faster than first/second

There are More: Predicting Power for Mixed Integer Linear Programming

- **MILP are hard to solve in general**
- **Special heuristics are needed for acceleration**

$$\begin{aligned} \min_{x,y} \quad & c_1^\top x + c_2^\top y \\ \text{subject to} \quad & Ax + By \leq d \\ & y_i \in \{0, 1\} \end{aligned}$$

Interior point solution of the LP relaxation is a natural prediction of the likelihood each variable takes 1 in the optimal solution since it contains unbiased information of the optimal set

How to use this information safely and efficiently?

Pooling the Risk via Variance Reduction

- **Given an MILP, the interior point solution of the LP relaxation tells us**

$$\begin{pmatrix} \hat{y}_1(\xi) \\ \hat{y}_2(\xi) \\ \vdots \\ \hat{y}_n(\xi) \end{pmatrix} = \begin{pmatrix} 0.99 \\ 0.12 \\ \vdots \\ 0.38 \end{pmatrix}$$

- **Each $\hat{y}$ is the likelihood a variable takes 1 or 0 in the optimal solution**

- **Each variable introduces some risk/variance of such rounding**

  so that dealing them separately results in extremely risk outcomes

Q: What should we do seeing a set of risky guesses? **A: Put them in a pool!**

Risk Pooling through Variance Reduction

- **Pooling the binary variables by adding “confidence” cardinality cuts**

$$\sum_{i \in \mathcal{U} = \{j : \hat{y}_j(\xi) \geq 0.9\}} y_i^*(\xi) \geq \alpha \cdot |\mathcal{U}| \qquad \sum_{i \in \{j : \hat{y}_j(\xi) \leq 0.1\}} y_i^*(\xi) \leq \beta \cdot |\mathcal{L}|$$

- **Intuitively we know that the above two inequalities are expectedly to hold for $\alpha \to 0.9$ and $\beta \to 0.1$**
- **These two inequalities are exactly *cutting planes* for MILP**
- **The last issue is how to choose α, β to increase the confidence level:**

  Interpret y_j^* as Bernoulli random variables with expectation $\hat{y}_j$,

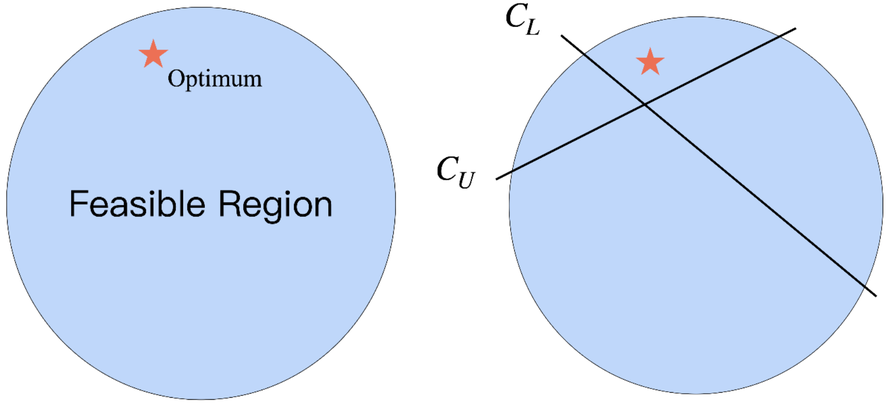
  then justify by concentration inequalities

Statistical Confidence Cut Generation (Gao at al. SHUFE, 2023)

Theorem 2. *Given independent random variables* $\{y_1^*(\xi),\ldots,y_n^*(\xi)\}$ *such that* $\mathbb{E}[y_i^*(\xi)|\xi]=\hat{y}_i(\xi)$, *letting* $\mathcal{U}:=\{i\colon \hat{y}_i(\xi)\geq\tau\}$ *and* $\mathcal{L}:=\{i\colon \hat{y}_i(\xi)<1-\tau\}$ *for* $0.5\leq\tau\leq 1$. *Then w.p.* $1-\delta$, *each of the inequalities below holds.*

$$C_{\mathcal{U}}\colon \sum_{i\in\mathcal{U}} y_i^*(\xi)\geq\sum_{i\in\mathcal{U}}\hat{y}_i(\xi)-\sqrt{\frac{|\mathcal{U}|}{2}\log(1/\delta)}$$

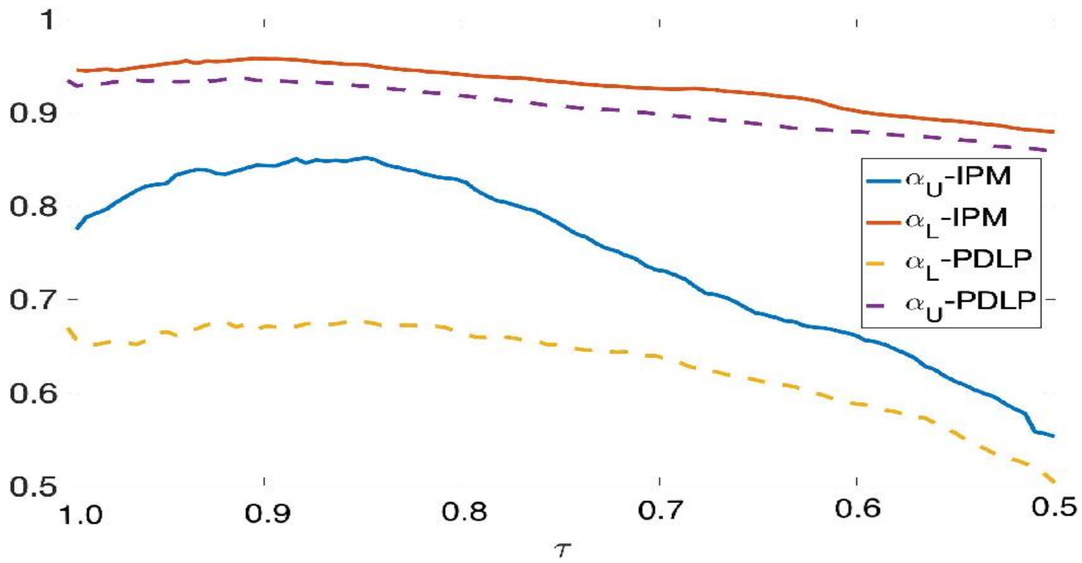
$$C_{\mathcal{L}}\colon \sum_{i\in\mathcal{L}} y_i^*(\xi)\leq\sum_{i\in\mathcal{L}}\hat{y}_i(\xi)+\sqrt{\frac{|\mathcal{L}|}{2}\log(1/\delta)}$$



- **Overall, the two cuts (and their complement) split the whole feasible region into four regions**
- **Solving the most likelihood region of two cuts often gives a satisfying solution with confidence**
- **Branching over all four regions independently will not miss the optimal solution**

Numerical Experiments: Online Cut-Generation

- Tested on IEEEE unit commitment problems using COPT
- Using pre-solved instances to compare speed



Accuracy of prediction by the IPM and PDLP

- Accuracy of interior point prediction can reach 80%
- No loss of optimality
- Remarkable acceleration using proper choosing cut generation parameters

Instance	Original	Stat. cut	Instance	Original	Stat. cut
1	882	187	6	1151	405
2	242	170	7	3600	1510
3	268	168	8	1820	1029
4	495	167	9	3600	758
5	241	182	10	3600	579

Improvement of COPT on IEEE instances

Offline-Training: Using past instances to improve prediction quality

Data-Driven Approaches to Mixed Integer Optimization

Learn from the past and predict the future such as the unit commitment problem in Electrical Power Generation

- **Many real-life MIO applications are solved on a regular basis**

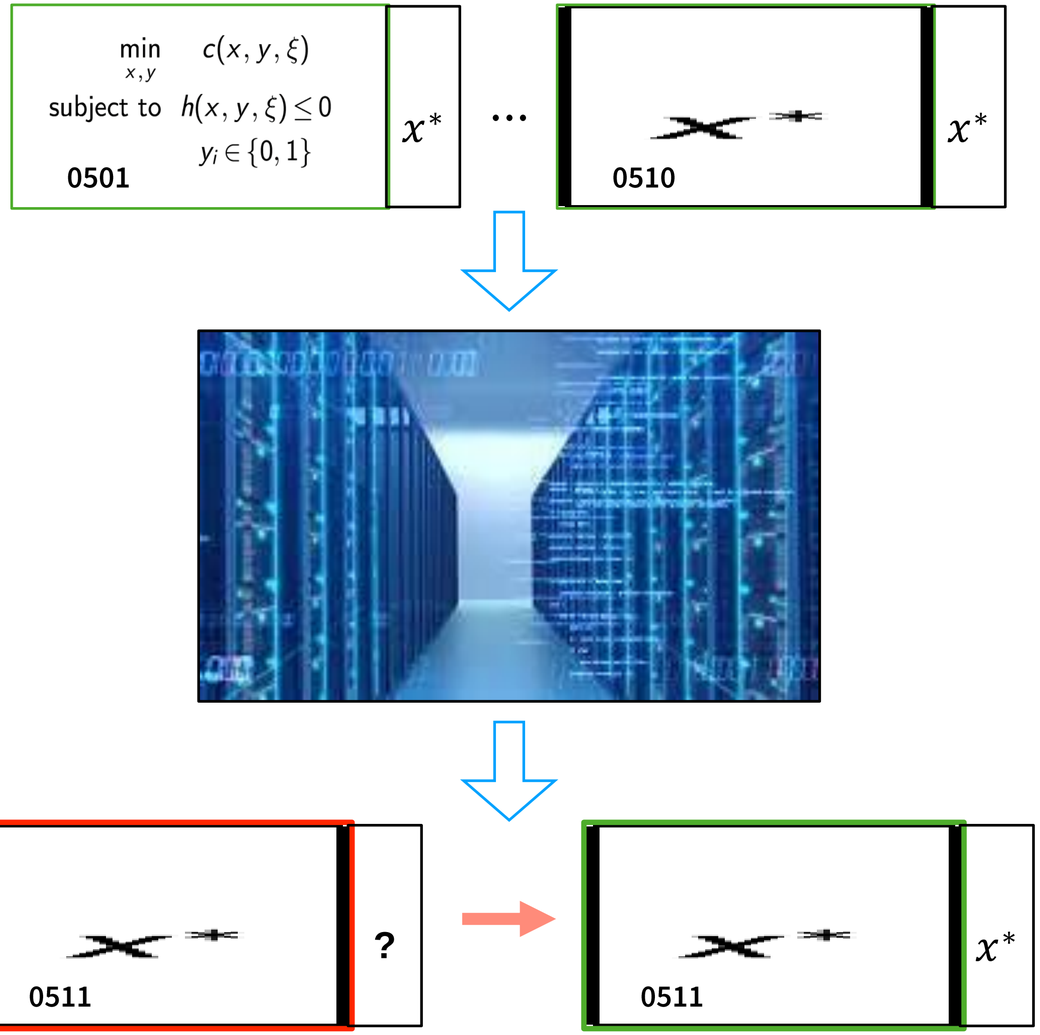
  Unit commitment; portfolio; scheduling…

- **Large amount of data and solutions collected from the past**

  Future instances are similar to the past

- **A natural idea: use machine learning to learn from history**

  Known as pre-trained data-driven approaches

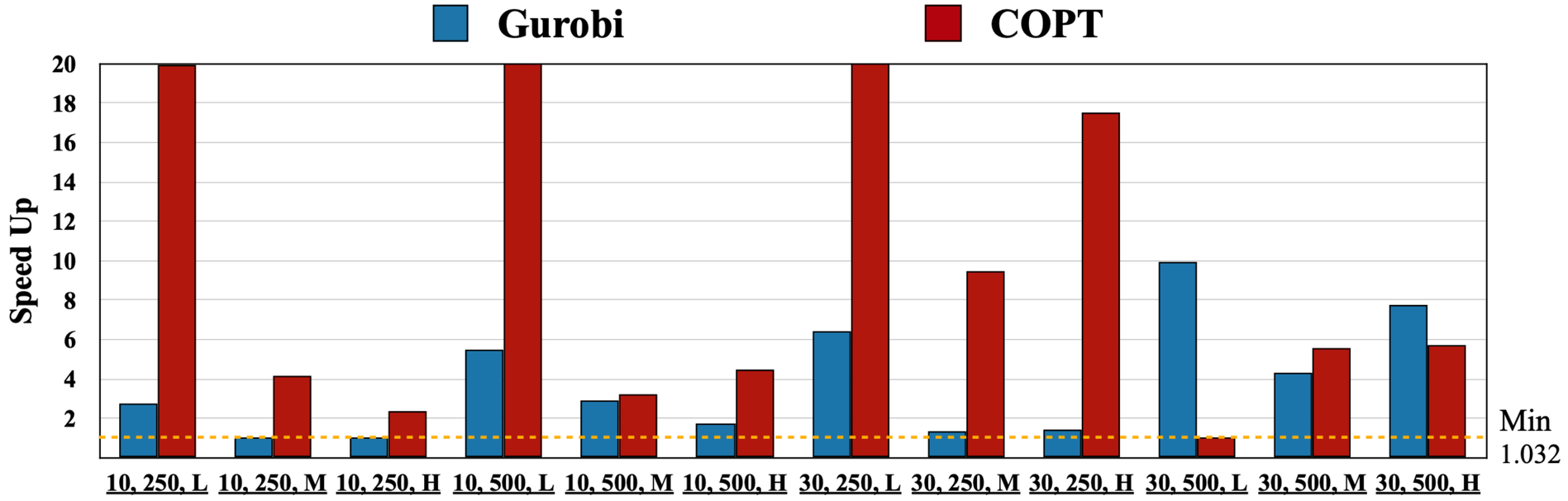


Numerical Test Results I

- **The method is tested on multi-knapsack, set-covering and unit-commitment problems**

  Train from 500 instances and test on 20 instances

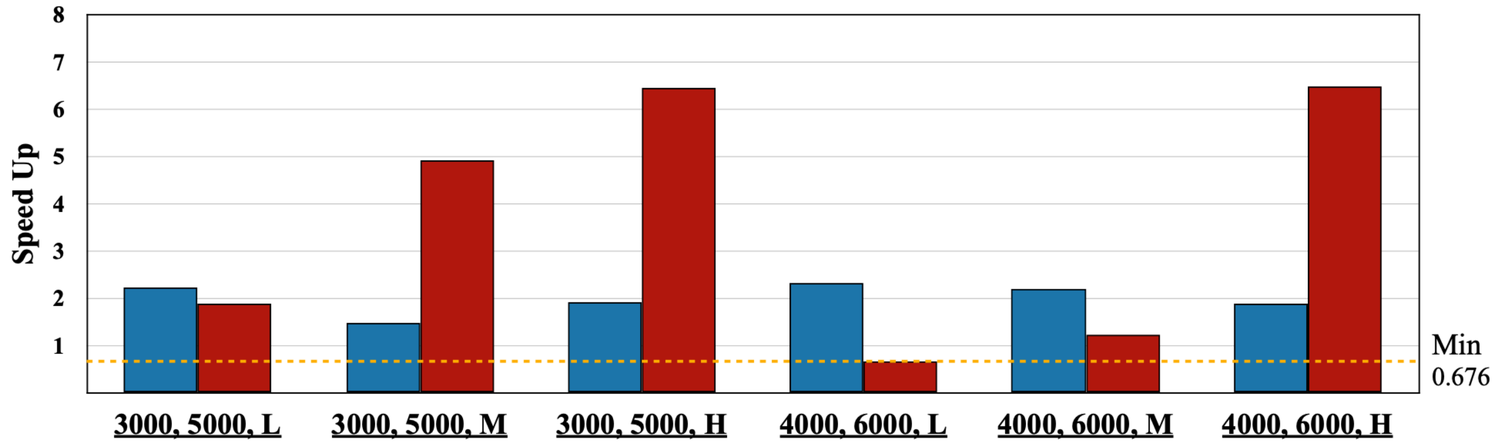
- **Measure the speedup of finding a good solution on in the region formed by two cuts**



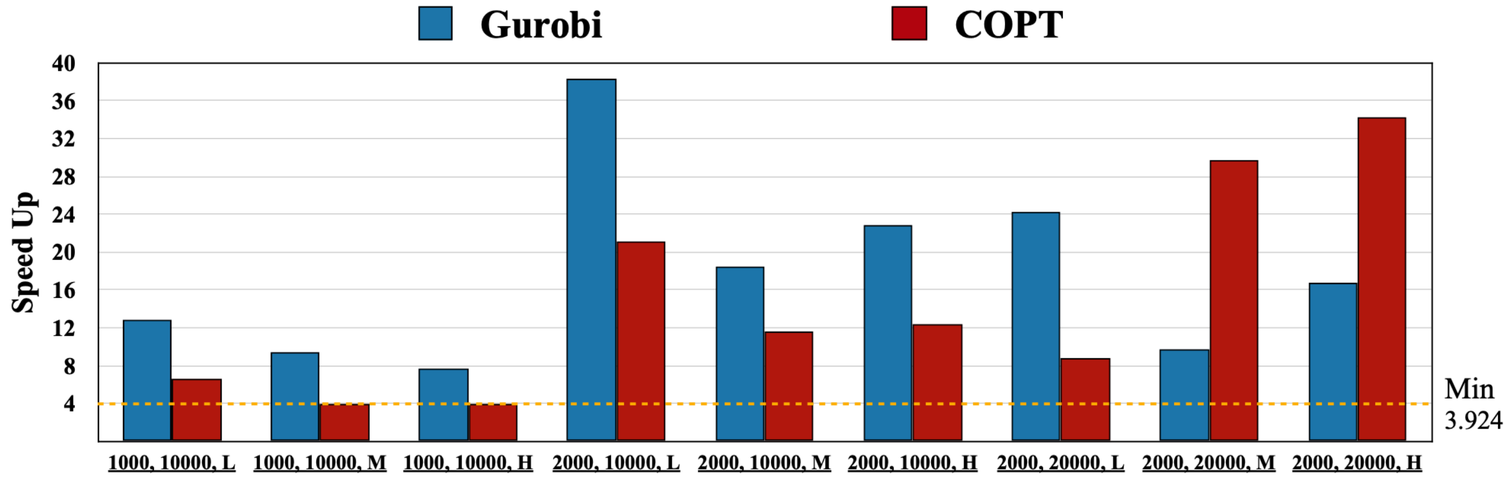
Average speedup on knapsack instances

Numerical Test Results II

- Acceleration by two lines of code
- Remarkable speedup on primal solution finding for both the state of art MIP solvers Gurobi and COPT
- No loss of optimality



Unit Commitment



Set-Covering

Takeaways

- **First-order potential reduction serves as a fast warm-start for high-precision second-order methods if needed**
- **Interior-point solutions provide prediction-power for cross-over and mixed-integer programming via statistical cardinality cut generation**

- THANK YOU