Recent Computational Progress on Linear Programming Solvers

LA/OPT Seminar

February 14, 2024

Yinyu Ye
Stanford University
Linear Programming and LP Giants

max or min \( \sum c_j x_j \)

s.t. \( \sum_j a_j x_j \leq b \),

\( 0 \leq x_j \leq 1 \quad \forall \ j = 1, \ldots, n \)
Today’s Talk

- LP Warm-Start: Online Helps Offline

- Smart Crossover: From an Interior Point to a Corner Points

- ABIP: Interior Point Method Meets ADMM

- cuPDLPC: How GPU Accelerates Solving LP

- Summary
Linear Programming as Combinatorial Classification

- Basic solution is one of the most important concept in LP

\[
\begin{align*}
\min_x & \quad c^T x \\
\text{subject to} & \quad Ax = b \\
& \quad x \geq 0
\end{align*}
\]

Knowledge of \( B \) reduces linear programming to a \textit{linear system}

- LP can be viewed a \textit{classification} task

Can we predict the basis?
Yes! Use the Dual
Classification using Duality

LP duality provides the most powerful classifier for LP

\[
\begin{align*}
\max_x & \quad c^T x \\
\text{subject to} & \quad Ax \leq b \\
& \quad 0 \leq x \leq u
\end{align*}
\]

Dual

\[
\begin{align*}
\min_{y,s} & \quad b^T y + u^T s \\
\text{subject to} & \quad s \geq c - A^T y \\
& \quad (y, s) \geq 0
\end{align*}
\]

If we get optimal \( y^* \), then optimality condition tells us

\[
x_j^* \in \begin{cases} 
\{0\}, & c_j - a_j^T y^* < 0 \\
[0, 1], & c_j - a_j^T y^* = 0 \\
\{1\}, & c_j - a_j^T y^* > 0
\end{cases}
\]

Dual solution tells us almost all about primal
Fast Training the Classifier $y^*$

- But solving dual problem is no easier than the primal

$$x_j^* \in \begin{cases} 
0, & c_j - a_j^\top y^* < 0 \\
[0, 1], & c_j - a_j^\top y^* = 0 \\
1, & c_j - a_j^\top y^* > 0 
\end{cases}$$

Dual solution tells us almost all about primal

- No matrix factorization
- No explicit matrix multiplication
- $O(\text{nnz}(A))$ flops
- Reasonable accuracy

The overall budget is only several MatVec

How can we fulfill the goals simultaneously?

Ans: Estimate on the fly by Online Linear Programming (OLP)

[Gao et al. ICML, 2023]
What is Online Linear Programming

- Decision maker needs to decide $x_t$: how much resources are allocated/sold to each customer

\[
\begin{align*}
\text{max} & \quad \sum_{t=1}^{T} r_t x_t \\
\text{s.t.} & \quad \sum_{t=1}^{T} a_{it} x_t \leq b_i, \quad i = 1, \ldots, m \\
& \quad 0 \leq x_t \leq 1 \quad \text{or} \quad x_t \in \{0, 1\}, \quad t = 1, \ldots, T
\end{align*}
\]

- Online setting:

- Customers arrive sequentially and the decision needs to be made instantly upon the customer arrival: Sell or No-sell?

[Agrawal et al. 2010, 2014], [Kesselheim et al., 2014]
[Li/Y, 2019], [Li et al., 2020],
Online Learning of $y^*$

Re-write the dual as

$$\min_{y, s} \quad b^\top y + u^\top s$$
subject to
$$s \geq c - A^\top y$$
$$(y, s) \geq 0$$

$$\min_{y \geq 0} \quad b^\top y + \sum_{j=1}^{n} [c_j - a_j^\top y]_+$$

• The dual objective is a finite-sum problem with minimal constraints

• When $n$ is large, dual objective is the sample approximation of a stochastic program

• What’s the most efficient way for finite-sum problem?

Ans: Online Sub-Gradient
Online Sub-Gradient Method

Solve finite-sum problem by OSG?

\[
\min_{y \geq 0} \quad b^T y + \sum_{j=1}^{n} [c_j - a_j^T y]_+
\]

On the dual side

• When read in a column \((c_j, a_j)\) data

Compute subgradient

\[
g_j = \frac{b}{n} - a_j I\{c_j > a_j^T y^j\}
\]

• Update \(y^j\) using (projected) subgradient

How to estimate \(\{x_j\}\)?

\[
x_j^* \in \begin{cases} 
{0}, & c_j - a_j^T y^* < 0 \\
[0, 1], & c_j - a_j^T y^* = 0 \\
{1}, & c_j - a_j^T y^* > 0
\end{cases}
\]

On the primal side

• Apply optimality condition on the fly

\[
x_j = I\{c_j > a_j^T y^j\}
\]

• May randomly sample columns multiple times and take average
Computational Results

Experiments on MIPLIB 2017 and MKP instances using Column-Generation

- 2x speedup on instances with many variables
- Simple, efficiently and almost no-cost
- Online LP helps pre-solving offline LP for **Warm Start**

<table>
<thead>
<tr>
<th>Data</th>
<th>Acc</th>
<th>Data</th>
<th>Acc</th>
</tr>
</thead>
<tbody>
<tr>
<td>scpm1</td>
<td>100%</td>
<td>rail507</td>
<td>90%</td>
</tr>
<tr>
<td>scp2</td>
<td>100%</td>
<td>rail516</td>
<td>88%</td>
</tr>
<tr>
<td>scpl4</td>
<td>100%</td>
<td>rail2586</td>
<td>94%</td>
</tr>
<tr>
<td>scp4</td>
<td>100%</td>
<td>rail4284</td>
<td>96%</td>
</tr>
</tbody>
</table>

Accuracy of classification
Today’s Talk

• LP Warm-Start: Online Helps Offline

• Smart Crossover: From an Interior Point to a Corner Point

• ABIP: Interior Point Method Meets ADMM

• cuPDLP-C: How GPU Accelerates Solving LP

• Summary
Linear Programming: the Need of Basic Feasible Solutions

$$\mathbf{x}^* = \arg\min_{\mathbf{x} \in P} \mathbf{c}^T \mathbf{x}$$

Crossover is the procedure from an interior-point solution to a BFS [Andersen/Y, 1996]
From an Interior Point to a Corner Point [Ge et al. 2021]

\[ \mathbf{x}^* = \arg\min_{x \in P} \mathbf{c}^\top \mathbf{x} \]

IPM Stops at \( x^k \)

Goal: Find a BFS that is in the sublevel set (enough for regular tolerance)

\[ P \cap \{ x : \mathbf{c}^\top \mathbf{x} \leq \mathbf{c}^\top x^k \} \]

Our approach: Solve a randomly-perturbed-objective problem

\[ \hat{x} = \arg\min_{x \in P} (\mathbf{c} + \Delta\mathbf{c})^\top \mathbf{x} \]

- If \( \Delta\mathbf{c} \) is too tiny, identifying the BFS \( \hat{x} \) is still hard
- If \( \Delta\mathbf{c} \) is too large, \( \hat{x} \) is no longer in the sublevel set

*We need theoretical guarantees to keep a balance on the size of \( \Delta\mathbf{c} \)!*
How Large Can the Perturbation be?

**Theorem:**
Let $x^k$ be any central-path solution of $\min_x c^T x$ s.t. $Ax = b, x \geq 0$. Then for any $\Delta c$ such that
\[
\|X_k \Delta c\|_2 \leq \frac{\|X_k (I - A^T (A X_k^2 A^T)^{-1} A X_k) c\|_2}{4n + 2},
\]
let $\hat{x}$ be the optimal solution of the perturbed problem, and then $c^T \hat{x} \leq c^T x^k$.

**Insight:**
We can generate the random perturbation $\Delta c$ within this range but as large as possible.
Flowchart of the Perturbation Crossover Method

Other heuristics:
1. Identify the feasibility problems.
2. Estimate the optimal face

Starting Interior-Point solution \((x^k, y^k)\)

Feasibility Problem?

Yes

Solve a Randomly Perturbed Problem

No

Solve the Perturbed Restricted Problem

Identify a Candidate Optimal Face \((\Theta^k_{\alpha}, \Theta^k_{\alpha'})\)

Compute a Perturbation that Satisfies (I)

Infeasible?

Yes

Decrease \(\gamma\)

No

BFS \(\hat{x}\)

\(c^\top \hat{x} - b^\top y^k < \varepsilon\)

Reoptimize

BFS \(x^*\)

End

End

No

BFS \(x^*\)

End
# Computational Results on some LP relaxations in MIPLIB

## LP relaxation of some max cut packing problems:

<table>
<thead>
<tr>
<th>Problem</th>
<th>Dimension of optimal face</th>
<th>Gurobi Barrier Method (seconds)</th>
<th>Gurobi Crossover (seconds)</th>
<th>Perturbation Crossover (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>graph20-20-1rand</td>
<td>2035</td>
<td>0.01</td>
<td>0.05</td>
<td>0.04</td>
</tr>
<tr>
<td>graph20-80-1rand</td>
<td>15912</td>
<td>0.05</td>
<td>2.42</td>
<td>1.11</td>
</tr>
<tr>
<td>graph40-20-1rand</td>
<td>20773</td>
<td>0.09</td>
<td>15.82</td>
<td>8.33</td>
</tr>
<tr>
<td>graph40-40-1rand</td>
<td>101700</td>
<td>0.41</td>
<td>323.41</td>
<td>50.79</td>
</tr>
<tr>
<td>graph40-80-1rand</td>
<td>282112</td>
<td>1.4</td>
<td>&gt;10000</td>
<td>872.07</td>
</tr>
</tbody>
</table>

Our crossover is much faster especially when the dimension of the optimal face is large.
More Experiments on the LP Benchmark Problems (LPopt)

Geometric Average Time for Obtaining an Optimal BFS

- **Gurobi's Crossover**
- **Our Perturbation Crossover**
- **The "Virtual Best"**

```
Some hard LP instances for crossover in Gurobi:

- datt256
  415.94 -> 18.19
- s82
  881.19 -> 0.53
- set_cover_model
  281.14 -> 1.28
- ...
```

"Optimal": the regular relative objective gap < 1e-8
Today’s Talk

• LP Warm-Start: Online Helps Offline

• Smart Crossover: from an Interior Point to a Corner Point

• ABIP: Interior Point Method Meets ADMM

• cuPDLP-C: How GPU Accelerates Solving LP

• Summary
ABIP [Lin et al., 2021]

- An ADMM based interior point method solver for LP problems

- The primal-dual pair of LP:

\[
\begin{align*}
\min_{(P)} & \quad c^T x \\
\text{s.t.} & \quad A x = b \\
\max_{(D)} & \quad b^T y \\
\text{s.t.} & \quad A^T y + s = c \\
& \quad x \geq 0 \\
& \quad s \geq 0
\end{align*}
\]

- For IPM, initial feasible interior solutions are hard to find

- So we consider homogeneous and self-dual (HSD) LP here!

\[
\begin{align*}
\min & \quad \beta(n + 1)\theta + 1(r = 0) + 1(\xi = -n - 1) \\
\text{s.t.} & \quad Qu = v, \\
& \quad y \text{ free, } x \geq 0, \tau \geq 0, \theta \text{ free, } s \geq 0, \kappa \geq 0
\end{align*}
\]

where

\[
Q = \begin{bmatrix}
0 & A & -b & \bar{b} \\
-A^T & 0 & c & -\bar{c} \\
\bar{b} & -\bar{c}^T & 0 & \bar{z} \\
-b & c & -\bar{z} & 0
\end{bmatrix}, \quad u = \begin{bmatrix} y \\ x \\ \tau \\ \theta \end{bmatrix}, \quad v = \begin{bmatrix} r \\ s \\ \kappa \end{bmatrix}, \quad \bar{b} = b - Ae, \quad \bar{c} = c - e, \quad \bar{z} = c^T e + 1
\]
ABIP – Subproblem

- Add log-barrier penalty for HSD LP and solve

\[
\begin{align*}
\min & \quad B(u, v, \mu) \\
\text{s.t.} & \quad Qu = v
\end{align*}
\]

- Traditional IPM applies Newton’s method to solve the subproblem, which can be too expensive when problem is large!

- Apply ADMM (with splitting) to solve the kth subproblem inexactely

\[
\begin{align*}
\min & \quad 1(Q\bar{u} = \bar{v}) + B(u, v, \mu^k) \\
\text{s.t.} & \quad (\bar{u}, \bar{v}) = (u, v)
\end{align*}
\]

where the augmented Lagrangian function

\[
\mathcal{L}_\beta(\bar{u}, \bar{v}, u, v, \mu^k, p, q) := 1(Q\bar{u} = \bar{v}) + B(u, v, \mu^k) - \langle \beta(p, q), (\bar{u}, \bar{v}) - (u, v) \rangle + \frac{\beta}{2} \| (\bar{u}, \bar{v}) - (u, v) \|^2
\]
## ABIP+ – Enhancements [Deng et al., 2022]

<table>
<thead>
<tr>
<th>Motivation</th>
<th>Enhancement</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADMM</td>
<td>Rescaling</td>
</tr>
<tr>
<td></td>
<td>Restart</td>
</tr>
<tr>
<td></td>
<td>Half-update</td>
</tr>
<tr>
<td>IPM</td>
<td>Adaptive barrier parameter</td>
</tr>
<tr>
<td>Practice</td>
<td>Inner loop convergence check</td>
</tr>
<tr>
<td></td>
<td>Strategy integration</td>
</tr>
<tr>
<td>Extension</td>
<td>Quadratic conic programming</td>
</tr>
</tbody>
</table>

Various enhancements significantly improve ABIP!
ABIP+ – Restart

• Idea: Let the **uniform average** of the past *few* points be the new starting point

• ABIP (or first-order method in general) tends to induce a spiral trajectory

• After restart, ABIP moves more aggressively and converges faster (reduce **almost 70% ADMM iterations**)!
Computational Results on Netlib

- Selected 105 Netlib instances
- $\epsilon = 10^{-6}, 10^6$ max ADMM iterations

<table>
<thead>
<tr>
<th>Method</th>
<th># Solved</th>
<th># IPM</th>
<th># ADMM</th>
<th>Avg. Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABIP</td>
<td>65</td>
<td>74</td>
<td>265418</td>
<td>87.07</td>
</tr>
<tr>
<td>+ restart</td>
<td>68</td>
<td>74</td>
<td>88257</td>
<td>23.63</td>
</tr>
<tr>
<td>+ rescale</td>
<td>84</td>
<td>72</td>
<td>77925</td>
<td>20.44</td>
</tr>
<tr>
<td>+ hybrid $\mu$ (=ABIP+)</td>
<td>86</td>
<td>22</td>
<td>73738</td>
<td>14.97</td>
</tr>
</tbody>
</table>

- Hybrid $\mu$: If $\mu > \epsilon$ use the aggressive strategy, otherwise use the LOQO strategy
- ABIP+ decreases both # IPM iterations and # ADMM iterations significantly
Computational Results on PageRank Problems

• 117 instances, generated from sparse matrix datasets: DIMACS10, Gleich, Newman and SNAP, where Second order methods in commercial solver fail in most of these instances.

• $\epsilon = 10^{-4}$, 5000 max ADMM iterations.

<table>
<thead>
<tr>
<th>Method</th>
<th># Solved</th>
<th>SGM</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDL (Julia)</td>
<td>117</td>
<td>1</td>
</tr>
<tr>
<td>ABIP+</td>
<td>114</td>
<td>1.28</td>
</tr>
</tbody>
</table>

• In staircase matrix case (# nodes = # edges), ABIP+ is significantly faster than PDL!

<table>
<thead>
<tr>
<th># nodes</th>
<th>PDL (Julia)</th>
<th>ABIP+</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^4$</td>
<td>8.60</td>
<td>0.93</td>
</tr>
<tr>
<td>$10^5$</td>
<td>135.67</td>
<td>10.36</td>
</tr>
<tr>
<td>$10^6$</td>
<td>2248.40</td>
<td>60.32</td>
</tr>
</tbody>
</table>

[PDLP, Applegate et al., 2021, 2023]
Today’s Talk

• Online Warm-Start: Online Helps Offline

• Smart Crossover: From an Interior Point to a Corner Point

• ABIP: Interior Point Method meets ADMM

• cuPDLPC: How GPU Accelerates Solving LP

• Summary
Drawbacks for the simplex method and IPMs

Factorization is memory demanding
- A sparse matrix may induce dense decomposition
- Factorization is difficult for huge-size problems (>10^9 variables)

Recent progresses
- Parallelizing first-order methods for Linear programming on GPU
- Utilizing matrix-vector products on GPU
- Julia prototype: cuPDLP.jl (Lu/Yang, 2023)
- C implementation and solver enhancements: cuPDLP-C (Lu et al., 2024)

Difficult for GPU and parallelization
- Factorization is not as efficient on GPU
- Operations like pivoting are hard to parallelize
- CPU and GPU communication problems
Primal-Dual Hybrid Gradient for Linear Programming

- cuPDLPhp uses the saddle-point formulation of LP

\[
\min_{x \in \mathbb{R}^n} c^\top x \\
\text{s.t.} \quad Gx \geq h \\
\quad Ax = b \\
\quad l \leq x \leq u
\]

\[
\min_{x \in X} \max_{y \in Y} L(x, y) := c^\top x - y^\top K x + q^\top y
\]

\[
\begin{aligned}
x^{t+1} &\leftarrow \text{proj}_X (x^t - \tau (c - K^\top y^t)) \\
y^{t+1} &\leftarrow \text{proj}_Y (y^t + \sigma (q - K (2x^{t+1} - x^t)))
\end{aligned}
\]

An Iteration of PDHG [Esser et al. 2010]:

- Computing \( Kx, K^\top y \) by sparse matrix-vector product (spmv)
- Choosing step sizes: \( \tau, \sigma \)
- PDLP Adaptive line-search: Applegate et al. (2021,2023), Lu/Yang (2023)
- All operations can be done on GPU!
## Selected MIPLIB Instances

<table>
<thead>
<tr>
<th>Instances</th>
<th>Variables</th>
<th>Constraints</th>
<th>Non-zeros</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Packing Cuts in Undirected Graphs.</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>graph20-80-1rand</td>
<td>16263</td>
<td>55107</td>
<td>191997</td>
</tr>
<tr>
<td>graph40-20-1rand</td>
<td>31243</td>
<td>99067</td>
<td>345557</td>
</tr>
<tr>
<td>graph40-40-1rand</td>
<td>102600</td>
<td>360900</td>
<td>1260900</td>
</tr>
<tr>
<td>graph40-80-1rand</td>
<td>283648</td>
<td>1050112</td>
<td>3671552</td>
</tr>
<tr>
<td><strong>Open Pit Mining over a cube considering multiple time periods and two knapsack constraints per period.</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rmine11</td>
<td>12292</td>
<td>97389</td>
<td>241240</td>
</tr>
<tr>
<td>rmine13</td>
<td>23980</td>
<td>197155</td>
<td>485784</td>
</tr>
<tr>
<td>rmine15</td>
<td>42438</td>
<td>358395</td>
<td>879732</td>
</tr>
<tr>
<td>rmine21</td>
<td>162547</td>
<td>1441651</td>
<td>3514884</td>
</tr>
<tr>
<td>rmine25</td>
<td>326599</td>
<td>2953849</td>
<td>7182744</td>
</tr>
<tr>
<td><strong>Unit Commitment problems (electricity production planning problems)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ucase7</td>
<td>33020</td>
<td>47132</td>
<td>335644</td>
</tr>
<tr>
<td>ucase8</td>
<td>37413</td>
<td>53709</td>
<td>214625</td>
</tr>
<tr>
<td>ucase9</td>
<td>33242</td>
<td>49565</td>
<td>332316</td>
</tr>
<tr>
<td>ucase10</td>
<td>110818</td>
<td>196498</td>
<td>787045</td>
</tr>
<tr>
<td>ucase12</td>
<td>62529</td>
<td>121161</td>
<td>419447</td>
</tr>
</tbody>
</table>
## Computational Results on Selected MIPLIB instances

<table>
<thead>
<tr>
<th>Instances</th>
<th>cuPDLP.jl V100</th>
<th>cuPDLP.jl H100</th>
<th>cuPDLP-C H100</th>
<th>Gurobi Barrier</th>
<th>COPT Barrier 1th, 16G</th>
<th>COPT Barrier 12th, 128G</th>
</tr>
</thead>
<tbody>
<tr>
<td>graph20-80-1rand</td>
<td>1.16</td>
<td>0.86</td>
<td>0.13</td>
<td>0.21</td>
<td>0.04</td>
<td>0.04</td>
</tr>
<tr>
<td>graph40-20-1rand</td>
<td>1.16</td>
<td>0.87</td>
<td>0.15</td>
<td>0.36</td>
<td>0.06</td>
<td>0.06</td>
</tr>
<tr>
<td>graph40-40-1rand</td>
<td>1.19</td>
<td>0.84</td>
<td>0.30</td>
<td>1.62</td>
<td>0.12</td>
<td>0.14</td>
</tr>
<tr>
<td>graph40-80-1rand</td>
<td>1.73</td>
<td>1.02</td>
<td>0.88</td>
<td>5.72</td>
<td>0.43</td>
<td>0.44</td>
</tr>
<tr>
<td>rmine11</td>
<td>42.81</td>
<td>32.80</td>
<td>16.70</td>
<td>9.79</td>
<td>5.06</td>
<td>2.26</td>
</tr>
<tr>
<td>rmine13</td>
<td>28.35</td>
<td>56.62</td>
<td>12.09</td>
<td>38.31</td>
<td>15.23</td>
<td>4.20</td>
</tr>
<tr>
<td>rmine15</td>
<td>35.14</td>
<td>32.02</td>
<td>22.40</td>
<td>149.59</td>
<td>68.90</td>
<td>13.55</td>
</tr>
<tr>
<td>rmine21</td>
<td>441.16</td>
<td>830.18</td>
<td>148.49</td>
<td>2674.46</td>
<td>1361.07</td>
<td>207.33</td>
</tr>
<tr>
<td>rmine25</td>
<td>1411.57</td>
<td>409.39</td>
<td>246.33</td>
<td>&gt; 3600.00</td>
<td>&gt; 3600.00</td>
<td>1839.05</td>
</tr>
<tr>
<td>uccase7</td>
<td>62.26</td>
<td>82.04</td>
<td>38.34</td>
<td>3.98</td>
<td>2.57</td>
<td>1.66</td>
</tr>
<tr>
<td>uccase8</td>
<td>14.57</td>
<td>14.92</td>
<td>7.04</td>
<td>2.62</td>
<td>1.86</td>
<td>1.18</td>
</tr>
<tr>
<td>uccase9</td>
<td>66.49</td>
<td>58.31</td>
<td>13.40</td>
<td>4.46</td>
<td>3.09</td>
<td>2.04</td>
</tr>
<tr>
<td>uccase10</td>
<td>65.49</td>
<td>99.36</td>
<td>20.76</td>
<td>2.68</td>
<td>1.22</td>
<td>0.90</td>
</tr>
<tr>
<td>uccase12</td>
<td>45.53</td>
<td>37.41</td>
<td>20.22</td>
<td>1.53</td>
<td>0.59</td>
<td>0.62</td>
</tr>
</tbody>
</table>

- GPU solver is less influenced by problem sizes
### Strengthening with other LP Techniques

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Optimizer</th>
<th>Presolver</th>
<th>Tol.</th>
<th>SGM10</th>
<th>Solved</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIPLIB (383)</td>
<td>COPT</td>
<td>-</td>
<td>$10^{-8}$</td>
<td>3.11</td>
<td>383</td>
</tr>
<tr>
<td></td>
<td>cuPDLP-C</td>
<td>HiGHS</td>
<td>$10^{-4}$</td>
<td>6.12</td>
<td>373</td>
</tr>
<tr>
<td></td>
<td>CLP</td>
<td></td>
<td>$10^{-4}$</td>
<td>7.95</td>
<td>372</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No Presolve</td>
<td>$10^{-4}$</td>
<td>10.28</td>
<td>370</td>
</tr>
<tr>
<td></td>
<td>cuPDLP.jl</td>
<td>No Presolve</td>
<td>$10^{-4}$</td>
<td>17.49</td>
<td>370</td>
</tr>
<tr>
<td>Mittelmann (49)</td>
<td>COPT</td>
<td>-</td>
<td>$10^{-8}$</td>
<td>13.81</td>
<td>48</td>
</tr>
<tr>
<td></td>
<td>cuPDLP-C</td>
<td>HiGHS</td>
<td>$10^{-4}$</td>
<td>31.84</td>
<td>46</td>
</tr>
<tr>
<td></td>
<td>CLP</td>
<td></td>
<td>$10^{-4}$</td>
<td>33.97</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No Presolve</td>
<td>$10^{-4}$</td>
<td>57.54</td>
<td>43</td>
</tr>
</tbody>
</table>

- Julia Prototype: cuPDLP.jl (Lu/Yang, 2023)
- C Implementation: cuPDLP-C (Lu et al., 2024)
- LP scaling and presolving techniques significantly improve the GPU solver
- cuPDLP-C with HiGHS backend are open-sourced at:
  - [github.com/COPT-Public/cuPDLP-C](https://github.com/COPT-Public/cuPDLP-C)
Milestones of Solving a Well-Known “Intractable” Instance

In a workshop in January 2008 on the Perspectives in Interior Point Methods for Solving Linear Programs, the instance zib03 with 29,128,799 columns, 19,731,970 rows and 104,422,573 non-zeros was made public. As it turned out, the simplex algorithm was not suitable to solve it and barrier methods needed at least about 256 GB of memory, which was not easily available at that time. The first to solve it was Christian Bliedt in April 2009, running CPLEX out-of-core with eight threads and converging in 12,035,375 seconds (139 days) to solve the LP without crossover. Each iteration took 56 hours! Using modern codes on a machine with 2 TB memory and 4 E7-8880v4 CPUs @ 2.20 GHz with a total of 88 cores, this instance can be solved in 59,432 seconds = 16.5 hours with just 10% of the available memory used. This is a speed-up of 200 within 10 years. However, when the instance was introduced in 2008, none of the codes was able to solve it. Therefore there was infinite progress in the first year. Furthermore, 2021 was the first time we were able to compute an optimal basis solution.

2008: Instance zib03
29,128,799 variables
19,731,970 constraints
104,422,573 non-zeros
Presolve can’t really reduce it

2009: Cplex Barrier (without crossover)
139 days (56 hours/IPM-iteration)

2019: IPM on a more advanced machine
16.5 hours

2023-24: cuPDLP-C (to 1e-6 tolerance)
1.7 hours on NVIDIA A6000
27 minutes on NVIDIA H100!

Today’s Talk

• LP Warm-Start: Online Helps Offline

• Smart Crossover: From an Interior Point to a Corner Point

• ABIP: Interior Point Method Meets ADMM

• cuPDLP-C: How GPU Accelerates Solving LP

• Summary
Scientific Research Drives (Conic) LP Solver Development

**COPT** Barrier solver [User guide Ge at al. 2022]
- Added in **COPT** 1.4, October 2020
- Leading in Barrier Benchmark since June 2021 (**COPT** 2)
- Continue to lead in new LP benchmarks since October 2022

There are 49 public and 16 undisclosed LP problems in new LP benchmark.

**COPT** is the only solver that can solve all of them in time. Barrier is more often the best choice for solving LP.

**Key Features**
- High performance presolver
- Deterministic Parallel Cholesky
- # threads-independent behaviors
- Parallel crossover
- Smart crossover
Performance Advances **COPT 1 – 7 on Solving LP**

- Tested on 49 public LP benchmark problems from Hans Mittelmann, using time limit 15000.
- The PDLP GPU version also solves to optimal basis, where the crossover is finished on CPU.
- COPT 7 + GPU* = Best of COPT 7 and PDLP with GPU support.
- Hardware: CPU: AMD 5900X (12 Threads) with 128G memory and NVIDIA 4090 with 24G memory.

<table>
<thead>
<tr>
<th>Barrier</th>
<th>Time</th>
<th>Improvement</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>COPT 1</td>
<td>2020.10</td>
<td>Initial barrier LP solver release.</td>
<td></td>
</tr>
<tr>
<td>COPT 2</td>
<td>2021.05</td>
<td>Independently development efficient alternatives for MKL/Pardiso, allows for better parallelization and numerical handling.</td>
<td>Solves set-cover-model 1.95 times faster.</td>
</tr>
<tr>
<td>COPT 3</td>
<td>2021.10</td>
<td>Developed and Implemented smart crossover.</td>
<td>Solves datt256 18.8 times faster.</td>
</tr>
<tr>
<td>COPT 4</td>
<td>2022.01</td>
<td>Improved parallel crossover implementation.</td>
<td>Solves a2864-99blp 2.02 times faster.</td>
</tr>
<tr>
<td>COPT 5</td>
<td>2022.06</td>
<td>Improved barrier ordering.</td>
<td>Solves dlr1 36% faster.</td>
</tr>
<tr>
<td>COPT 6</td>
<td>2022.10</td>
<td>Improved LP presolver.</td>
<td>Solves rail02 28% faster.</td>
</tr>
<tr>
<td>COPT 7</td>
<td>2023.09</td>
<td>Revised starting point computation.</td>
<td>Solves s82 45% faster.</td>
</tr>
<tr>
<td>COPT 7 + GPU*</td>
<td>2024.01</td>
<td>Added PDLP with GPU support.</td>
<td>Solves thk_63 63% faster.</td>
</tr>
</tbody>
</table>
Performance Advances **COPT 5 – 7 on Solving SDP**

- Testing machine AMD 5900X with 128G memory.
- Testing time limit 40000s.
- **COPT 7.0** leads in the Mittelmann SDP benchmark (Feb. 1, 2024).

<table>
<thead>
<tr>
<th>SDP</th>
<th>Time</th>
<th>Improvement</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>COPT 5</td>
<td>2022.06</td>
<td>Initial SDP solvers release with all of Primal-Dual, ABIP/ADMM and Dual method.</td>
<td>Solves theta12 7.5 times faster.</td>
</tr>
<tr>
<td>COPT 6</td>
<td>2022.10</td>
<td>• Rewrote and improved ABIP/ADMM implementation.</td>
<td>• Rewrote and improved Dual method implementation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Rewrote and improved Dual method implementation.</td>
<td>• Solves G55mc 6.85 times faster.</td>
</tr>
<tr>
<td>COPT 7</td>
<td>2023.09</td>
<td>Improved Primal-Dual method parallelism for large SDPs with many cones.</td>
<td>Solves Bex2_1_5 93% faster.</td>
</tr>
</tbody>
</table>

1 Feb 2024 -----------------------------------------------
Several SDP-codes on sparse and other SDP problems
-----------------------------------------------
Hans D. Mittelmann (mittelmann@asu.edu)

Scaled shifted geometric means of runtimes ("1" is fastest solver)

<table>
<thead>
<tr>
<th>Count of &quot;a&quot;</th>
<th>6</th>
<th>5</th>
<th>0</th>
<th>17</th>
<th>13</th>
<th>2</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solved of 75</td>
<td>75</td>
<td>70</td>
<td>73</td>
<td>61</td>
<td>69</td>
<td>62</td>
<td>70</td>
<td>75</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Problem</th>
<th>COPT</th>
<th>CSDP</th>
<th>MOSEK</th>
<th>SDPA</th>
<th>SDPT3</th>
<th>SeDuMi</th>
<th>HDSDP</th>
<th>MDOPT</th>
</tr>
</thead>
</table>

---
In 2019, COPT first stood on the solver stage with its high-performance LP simplex solver.

At present, COPT 7.0 has become one of the fastest solver in the world for various problem types.

**Benchmarks for Optimization Software**

http://plato.asu.edu/guide.html by Prof. Hans Mittelmann

**Simplex Benchmark, 2019**

**Problem Types**

<table>
<thead>
<tr>
<th>Linear Programming</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mixed Integer Linear Programming</td>
<td>2</td>
</tr>
<tr>
<td>Second-Order Cone Programming</td>
<td>1</td>
</tr>
<tr>
<td>Convex Quadratic Programming and Convex Quadratically Constrained Programming</td>
<td>1</td>
</tr>
<tr>
<td>Semi-Definite Programming</td>
<td>1</td>
</tr>
<tr>
<td>Mixed Integer Second-Order Cone Programming</td>
<td>1</td>
</tr>
<tr>
<td>Mixed Integer Convex Quadratic Programming</td>
<td>1</td>
</tr>
</tbody>
</table>
LP Real-World Applications (from Cardinal Operations)

Education and Academic Research
Energy and Electricity
Industry 4.0
Supply Chain

Aviation
Transportation
Finance
Warehouse and Logistics

Long Live – Linear Programming