S7303 - FINDING PARALLELISM IN GENERAL-PURPOSE LINEAR PROGRAMMING

Daniel Thuerck\textsuperscript{1,2} (advisors Michael Goesele\textsuperscript{1,2} and Marc Pfetsch\textsuperscript{1})
Maxim Naumov\textsuperscript{3}

\textsuperscript{1} Graduate School of Computational Engineering, TU Darmstadt
\textsuperscript{2} Graphics, Capture and Massively Parallel Computing, TU Darmstadt
\textsuperscript{3} NVIDIA Research
INTRODUCTION TO LINEAR PROGRAMMING
Linear Programs

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

- Linear objective function
- Linear constraints

where \( A = \begin{bmatrix} a_1^T \\ \vdots \\ a_m^T \end{bmatrix} \) and \( b = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix} \)
Linear Programs: Applications

[Stefano Lucidi, U Roma]

[General Electric]

[3P Logistics]
Lower-Level Parallelism in LP

INTERNALS OF AN LP SOLVER
Solving LPs

\[
\min \quad c^\top x \\
\text{s.t.} \quad Ax = b \\
\quad x \geq 0
\]

- A is \( m \times n \) matrix, with \( m \ll n \)
- A is sparse and has full row-rank
- Variables may be bounded: \( l \leq x \leq u \)

“Standard” LP format
Solving LPs

Simplex

Interior Point
Solving LPs

Simplex

Interior Point (IPM)

\[ A_B = \begin{pmatrix} \end{pmatrix} \]

“Basis” (active set)

\[ A \begin{pmatrix} \end{pmatrix} D - 1 A^T \]

“Augmented (Newton) System”

\[ \begin{bmatrix} D & A^T \\ A & 0 \end{bmatrix} \]

“Normal Equations”

\[ AD^{-1} A^T \]
Solving LPs

### IPM / Aug. System

\[
\begin{bmatrix}
D & A^\top \\
A & \end{bmatrix}
\]

- \((m + n) \times (m + n)\), sparse
- Symmetric, indefinite
- Solution: Indefinite LDL^T or MINRES method

### IPM / Normal Equations

\[
AD^{-1}A^\top
\]

- \(m \times m\), SPD, **might be dense**
- Squared condition number
- Solution: Cholesky-factorization or CG method
Solving LPs

**IPM / Aug. System**

\[
\begin{bmatrix}
D & A^T \\
A &
\end{bmatrix}
\]

- \((m + n) \times (m + n)\), sparse
- Symmetric, indefinite
- Solution: Indefinite \(LDL^T\) or MINRES method

**IPM / Normal Equations**

\[AD^{-1}A^T\]

- \(m \times m\), SPD, **might be dense**
- Squared condition number
- Solution: Cholesky-factorization or CG method
Introducing culip-lp...

An ongoing implementation of Mehrotra’s Primal-Dual interior point algorithm [1], featuring...

✓ (Iterative) Linear Algebra based on the “Augmented Matrix” approach,

✓ Full-rank guarantees,

✓ Comprehensive preprocessing & scaling.

Towards solving large-scale LPs on the GPU as open source for everybody
Progress report

IMPLEMENTING CULIP-LP
Solver architecture

- Preprocess
- Scale
- Standardize
- IPM loop
Solver architecture

Input data:

- Constraints
  
  \[ A_{eq}x = b_{eq} \]

- Constraints
  
  \[ A_{le}x \leq b_{le} \]

- Objective vector
  
  \[ c \]

- Bounds (on some variables)
  
  \[ l, u \]
Solver architecture

Storage format: CSR

- Compressed sparse row format
- Provides efficient row-based access by 3 arrays:

\[ A_{eq}x = b_{eq} \]
\[ A_{le}x \leq b_{le} \]

\[ c \]
\[ l, u \]

\[ a \quad b \quad c \quad d \]
\[ e \]

\[ \text{row_ptr} \]
0 → (a, b, c, d)
2
3
4
5

\[ \text{col_ind} \]
0 1 1 2 0
a b c d e

\[ \text{val} \]
Solver architecture

- **Example**: LP “pb-simp-nonunif” (see [2])
  - Input matrix: 1,4 Mio x 23k with 4,36 Mio nonzeros
  - Removed 1 singleton inequality
  - Removed 3629 low-forcing constraints
  - Removed 1 fixed variable
  - Removed 1,1 Mio (!) singleton inequalities
  - Result: approx. 3,6 Mio nonzeros removed
**Solver architecture**

---

**Goal:** Reduce element variance in matrices

- Scaling [3] makes a difference
  1. Geometric scaling (1x - 4x)
     
    \[ A_{i,\text{r}} = \frac{A_{i,\text{r}}}{\max(|A_{i,\text{r}}|) \min(|A_{i,\text{r}}|)} \]
  
    \[ A_{i,\text{r}} = \frac{A_{i,\text{r}}}{\|A_{i,\text{r}}\|_2} \]

- Equilibration (1x)
Solver architecture

Goal: Format LP in standard form

- Shift variables:
  \[ 1 \leq x \leq u \rightarrow 0 \leq x' \leq u + l \]

- Split (free) variables
  \[ x \rightarrow x = x^+ - x^- \quad x^+, x^- \geq 0 \]

- Build std’ matrix:
  \[
  \begin{pmatrix}
  A_{le} & I \\
  A_{eq} & \end{pmatrix}
  \begin{pmatrix}
  x \\
  b_{Le}
  \end{pmatrix}
  \]

\[
\begin{aligned}
A_{eq}x &= b_{eq} \\
A_{le}x &\leq b_{le} \\
c &
\end{aligned}
\]

\[
\begin{aligned}
l, u 
\end{aligned}
\]
Solver architecture

Ensure A has full rank (symbolically)

\[ PAQ = \begin{cases} m_u \\ m_c \end{cases} \]

\[ Ax = b \]

\[ \begin{align*}
    c \\
    u
\end{align*} \]

\[ m_u \leq \text{rank}(A) \leq \text{structural rank}(A) \]
Goal: Solve KKT conditions by Newton steps

Steps:
- Augmented matrix assembly
- Solving the (indefinite) augmented matrix
- Solve twice: predictor and corrector
- Stepsize along $v = v_p + v_c$
Solving the augmented system

\[
\begin{bmatrix}
D & A^T \\
A & 0
\end{bmatrix}
\]

**Iterative strategy:**
- Symmetric, indefinite: use MINRES [4] (in parts)
- Equilibrate system implicitly
- Preconditioner: Experiments ongoing

**Direct strategy:**
- Symmetric, indefinite: use SPRAL SSIDS [5]
- Reordering by METIS [6]
- Scaling for large pivots

\[\sim 95\% \text{ of computation}\]
Intermediate findings

PERFORMANCE EVALUATION
## Benchmark problems

<table>
<thead>
<tr>
<th>Problem name [7]</th>
<th>M</th>
<th>N</th>
<th>NNZ</th>
</tr>
</thead>
<tbody>
<tr>
<td>ex9</td>
<td>40,962</td>
<td>10,404</td>
<td>517,112</td>
</tr>
<tr>
<td>ex10</td>
<td>696,608</td>
<td>17,680</td>
<td>1,162,000</td>
</tr>
<tr>
<td>neos-631710</td>
<td>169,576</td>
<td>167,056</td>
<td>834,166</td>
</tr>
<tr>
<td>bley_xl1</td>
<td>175,620</td>
<td>5831</td>
<td>869,391</td>
</tr>
<tr>
<td>map06</td>
<td>328,818</td>
<td>164,547</td>
<td>549,920</td>
</tr>
<tr>
<td>map10</td>
<td>328,818</td>
<td>164,547</td>
<td>549,920</td>
</tr>
<tr>
<td>nb10tb</td>
<td>150,495</td>
<td>73340</td>
<td>1,172,289</td>
</tr>
<tr>
<td>neos-142912</td>
<td>58,726</td>
<td>416,040</td>
<td>1,855,220</td>
</tr>
<tr>
<td>in</td>
<td>1,526,202</td>
<td>1,449,074</td>
<td>6,811,639</td>
</tr>
</tbody>
</table>
## Performance

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ex9</td>
<td>517,112</td>
<td>X (NC)</td>
<td></td>
</tr>
<tr>
<td>ex10</td>
<td>1,162,000</td>
<td>X (NS)</td>
<td>141</td>
</tr>
<tr>
<td>neos-631710</td>
<td>834,166</td>
<td>172</td>
<td>478</td>
</tr>
<tr>
<td>bley_xl1</td>
<td>869,391</td>
<td>X (NS)</td>
<td>1,492</td>
</tr>
<tr>
<td>map06</td>
<td>549,920</td>
<td>X (NC)</td>
<td>466</td>
</tr>
<tr>
<td>map10</td>
<td>549,920</td>
<td>X (NC)</td>
<td>615</td>
</tr>
<tr>
<td>nb10tb</td>
<td>1,172,289</td>
<td>X (NC)</td>
<td>2,461</td>
</tr>
<tr>
<td>neos-142912</td>
<td>1,855,220</td>
<td>356</td>
<td>447</td>
</tr>
<tr>
<td>in</td>
<td>6,811,639</td>
<td>X (NS)</td>
<td>NC</td>
</tr>
</tbody>
</table>

X – failed, NS – did not start 1st iteration, NC – did not converged within 1 hour
Runtime breakdown

Problem: map10 [7]
Iterative vs. direct methods

MINRES Iterations

IPM step

Iterations

Predictor
Corrector

MINRES relative residual

IPM step

Relative Residual

Example: map10 [7]
Numerical difficulty

Condition of matrix

\[
\begin{bmatrix}
D & A^T \\
A & \end{bmatrix}
\]

- depends mainly on

\[D = \text{diag}(x) \cdot \text{diag}(s)\]

- with strong duality towards the end often yielding

\[
\frac{\max(x_i s_i)}{\min(x_i s_i)} \approx 10^{10}
\]

Remedies

- 2x2 pivoting in factorizations (e.g. \(LDL^T\) in SPRAL)
- Preconditioning for MINRES or GMRES

\text{expect speed-up here}

where

- \(x^T=[x_1,\ldots,x_n]\) are solution and
- \(s^T=[s_1,\ldots,s_n]\) are slack variables
What’s keeping you from optimizing your runtime?

LP Solver (a.k.a “the black box”)
Higher-Level Parallelism in LP

FEASIBILITY STUDY: LP DECOMPOSITIONS
Solving an LP: The usual setup

Large LP → LP Solver (a.k.a “the black box”) → Solution
LP-decompositions: feasibility

Decomposition works on structure of the constraint matrix $A$:

- Benders [9]
- Dantzig-Wolfe [10]
Higher-level parallelism by LP decomposition

1. Large LP
2. Apply LP decomposition
3. Master LP
4. Slave LP 1
5. Slave LP k
6. Assemble master/slave solutions
7. Solution
LP-decompositions: prototype

Implemented a Benders’ decomposition using hypergraph partitioning:

<table>
<thead>
<tr>
<th>Name</th>
<th>M</th>
<th>N</th>
<th>K</th>
<th># iterations</th>
<th># statics</th>
<th># coupling</th>
</tr>
</thead>
<tbody>
<tr>
<td>pigeon-10</td>
<td>1331</td>
<td>430</td>
<td>2</td>
<td>93</td>
<td>58</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>1331</td>
<td>430</td>
<td>4</td>
<td>102</td>
<td>71</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>1331</td>
<td>430</td>
<td>6</td>
<td>102</td>
<td>71</td>
<td>28</td>
</tr>
<tr>
<td>glass4</td>
<td>715</td>
<td>322</td>
<td>2</td>
<td>15</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>715</td>
<td>322</td>
<td>4</td>
<td>14</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>715</td>
<td>322</td>
<td>6</td>
<td>13</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>aflow40b (dual)</td>
<td>4170</td>
<td>2728</td>
<td>4</td>
<td>1048</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>4170</td>
<td>2728</td>
<td>6</td>
<td>1017</td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>
Acknowledgements

The work of Daniel Thuerck is supported by the 'Excellence Initiative' of the German Federal and State Governments and the Graduate School of Computational Engineering at Technische Universität Darmstadt.
References


References


