Linear Programming solvers: the state of the art

Julian Hall

School of Mathematics, University of Edinburgh

4th ISM-ZIB-IMI MODAL Workshop
Mathematical Optimization and Data Analysis

Tokyo

27 March 2019

THE UNIVERSITY of EDINBURGH

Overview

- LP background
- Serial simplex
- Interior point methods
- Solvers
- Parallel simplex
  - For structured LP problems
  - For general LP problems
- A novel method

Solution of linear programming (LP) problems

$$\text{minimize} \quad f = \boldsymbol{c}^T\boldsymbol{x} \qquad \text{subject to} \quad A\boldsymbol{x} = \boldsymbol{b} \qquad \boldsymbol{x} \geq \boldsymbol{0}$$

Background

- Fundamental model in optimal decision-making
- Solution techniques
  - Simplex method (1947)
  - Interior point methods (1984)
  - Novel methods
- Large problems have
  - 10^3–10^8 variables
  - 10^3–10^8 constraints
- Matrix A is (usually) sparse

Example

STAIR: 356 rows, 467 columns and 3856 nonzeros

Solving LP problems: Necessary and sufficient conditions for optimality

$$\text{minimize} \quad f = \boldsymbol{c}^T\boldsymbol{x} \qquad \text{subject to} \quad A\boldsymbol{x} = \boldsymbol{b} \qquad \boldsymbol{x} \geq \boldsymbol{0}$$

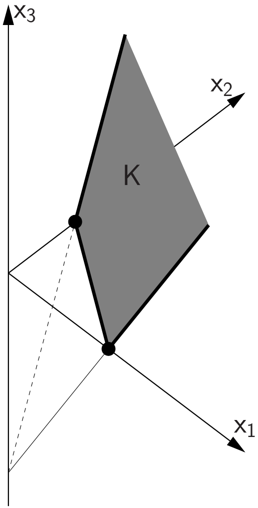
Karush-Kuhn-Tucker (KKT) conditions

$\boldsymbol{x}^*$ is an optimal solution $\iff$ there exist $\boldsymbol{y}^*$ and $\boldsymbol{s}^*$ such that

$$\begin{array}{rclcrclcrcll} A\boldsymbol{x} &=& \boldsymbol{b} & (1) & \boldsymbol{x} &\geq& \boldsymbol{0} & (3) & \boldsymbol{x}^T\boldsymbol{s} &=& 0 & (5) \\ A^T\boldsymbol{y} + \boldsymbol{s} &=& \boldsymbol{c} & (2) & \boldsymbol{s} &\geq& \boldsymbol{0} & (4) & & & & \end{array}$$

- For the **simplex algorithm** (1–2 and 5) always hold
  - **Primal** simplex algorithm: (3) holds and the algorithm seeks to satisfy (4)
  - **Dual** simplex algorithm: (4) holds and the algorithm seeks to satisfy (3)
- For **interior point methods** (1–4) hold and the method seeks to satisfy (5)

Solving LP problems: Characterizing the feasible region



minimize $\quad f = \boldsymbol{c}^T\boldsymbol{x} \qquad$ subject to $\quad A\boldsymbol{x} = \boldsymbol{b} \qquad \boldsymbol{x} \geq \boldsymbol{0}$

- $A \in \mathbb{R}^{m \times n}$ is full rank
- Solution of $A\boldsymbol{x} = \boldsymbol{b}$ is a $n - m$ dim. **hyperplane** in $\mathbb{R}^n$
- Intersection with $\boldsymbol{x} \geq \boldsymbol{0}$ is the **feasible region** K
- A vertex of K has
    - m **basic** components, $i \in \mathcal{B}$ given by $A\boldsymbol{x} = \boldsymbol{b}$
    - $n - m$ zero **nonbasic** components, $j \in \mathcal{N}$

  where $\mathcal{B} \cup \mathcal{N}$ partitions $\{1, \ldots, n\}$
- A solution of the LP occurs at a **vertex** of K

Solving LP problems: Optimality conditions at a vertex

$$\text{minimize} \quad f = \boldsymbol{c}^T\boldsymbol{x} \qquad \text{subject to} \quad A\boldsymbol{x} = \boldsymbol{b} \qquad \boldsymbol{x} \geq \boldsymbol{0}$$

Karush-Kuhn-Tucker (KKT) conditions

$\boldsymbol{x}^*$ is an optimal solution $\iff$ there exist $\boldsymbol{y}^*$ and $\boldsymbol{s}^*$ such that

$$\begin{aligned} A\boldsymbol{x} &= \boldsymbol{b} \quad (1) & \boldsymbol{x} &\geq \boldsymbol{0} \quad (3) & \boldsymbol{x}^T\boldsymbol{s} &= 0 \quad (5) \\ A^T\boldsymbol{y} + \boldsymbol{s} &= \boldsymbol{c} \quad (2) & \boldsymbol{s} &\geq \boldsymbol{0} \quad (4) & & \end{aligned}$$

- Given $\mathcal{B} \cup \mathcal{N}$, partition A as $\begin{bmatrix} B & N \end{bmatrix}$, $\boldsymbol{x}$ as $\begin{bmatrix} \boldsymbol{x}_B \\ \boldsymbol{x}_N \end{bmatrix}$, $\boldsymbol{c}$ as $\begin{bmatrix} \boldsymbol{c}_B \\ \boldsymbol{c}_N \end{bmatrix}$ and $\boldsymbol{s}$ as $\begin{bmatrix} \boldsymbol{s}_B \\ \boldsymbol{s}_N \end{bmatrix}$
- If $\boldsymbol{x}_N = \boldsymbol{0}$ and $\boldsymbol{x}_B = \widehat{\boldsymbol{b}} \equiv B^{-1}\boldsymbol{b}$ then $B\boldsymbol{x}_B + N\boldsymbol{x}_N = \boldsymbol{b}$ so $A\boldsymbol{x} = \boldsymbol{b}$ (1)
- For (2)

$$\begin{bmatrix} B^T \\ N^T \end{bmatrix} \boldsymbol{y} + \begin{bmatrix} \boldsymbol{s}_B \\ \boldsymbol{s}_N \end{bmatrix} = \begin{bmatrix} \boldsymbol{c}_B \\ \boldsymbol{c}_N \end{bmatrix}$$

- If $\boldsymbol{y} = B^{-T}\boldsymbol{c}_B$ and $\boldsymbol{s}_B = \boldsymbol{0}$ then $B^T\boldsymbol{y} + \boldsymbol{s}_B = \boldsymbol{c}_B$
- If $\boldsymbol{s}_N = \widehat{\boldsymbol{c}}_N \equiv \boldsymbol{c}_N - N^T\boldsymbol{y}$ then (2) holds
- Finally, $\boldsymbol{x}^T\boldsymbol{s} = \boldsymbol{x}_B^T\boldsymbol{s}_B + \boldsymbol{x}_N^T\boldsymbol{s}_N = 0$ (5)
- Need $\widehat{\boldsymbol{b}} \geq \boldsymbol{0}$ for (3) and $\widehat{\boldsymbol{c}}_N \geq \boldsymbol{0}$ for (4)

Solving LP problems: Simplex and interior point methods

Simplex method (1947)

- Given $\mathcal{B} \cup \mathcal{N}$ so (1–2 and 5) hold
- Primal simplex method

  Assume $\hat{\boldsymbol{b}} \geq \mathbf{0}$ (3)

  Force $\hat{\boldsymbol{c}}_N \geq \mathbf{0}$ (4)
- Dual simplex method

  Assume $\hat{\boldsymbol{c}}_N \geq \mathbf{0}$ (4)

  Force $\hat{\boldsymbol{b}} \geq \mathbf{0}$ (3)
- Modify $\mathcal{B} \cup \mathcal{N}$
- **Combinatorial approach**
- Cost $O(2^n)$ iterations

  Practically: $O(m+n)$ iterations

Interior point method (1984)

- Replace $\boldsymbol{x} \geq 0$ by **log barrier**
- Solve

$$\begin{array}{ll} \text{maximize} & f = \boldsymbol{c}^T\boldsymbol{x} + \mu \sum_{j=1}^{n} \ln(x_j) \\ \text{subject to} & A\boldsymbol{x} = \boldsymbol{b} \end{array}$$

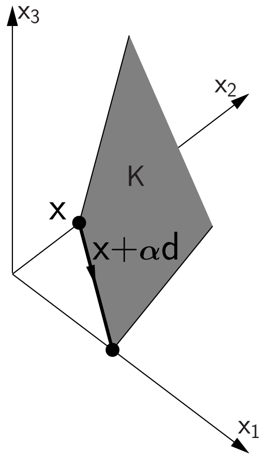
- KKT (5) changes:

  Replace $\boldsymbol{x}^T\boldsymbol{s} = 0$ by $XS = \mu\boldsymbol{e}$

  X and S have $\boldsymbol{x}$ and $\boldsymbol{s}$ on diagonal
- KKT (1–4) hold
- Satisfy (5) by forcing $XS = \mu\boldsymbol{e}$ as $\mu \to 0$
- **Iterative approach**
- Practically: $O(\sqrt{n})$ iterations

Simplex method

The simplex algorithm: Definition



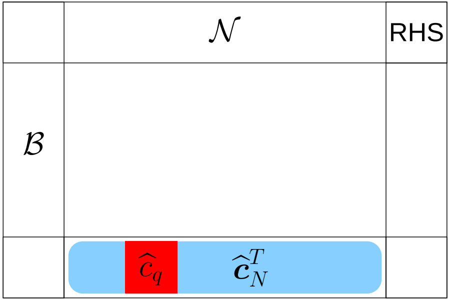
At a feasible vertex $\boldsymbol{x} = \begin{bmatrix} \widehat{\boldsymbol{b}} \\ \boldsymbol{0} \end{bmatrix}$ corresponding to $\mathcal{B} \cup \mathcal{N}$

1. If $\widehat{\boldsymbol{c}}_N \geq \boldsymbol{0}$ then **stop: the solution is optimal**
2. Scan $\widehat{c}_j < 0$ for q to leave $\mathcal{N}$
3. Let $\widehat{\boldsymbol{a}}_q = B^{-1} N \boldsymbol{e}_q$ and $\boldsymbol{d} = \begin{bmatrix} -\widehat{\boldsymbol{a}}_q \\ \boldsymbol{e}_q \end{bmatrix}$
4. Scan $\widehat{b}_i / \widehat{a}_{iq} > 0$ for α and p to leave $\mathcal{B}$
5. Exchange p and q between $\mathcal{B}$ and $\mathcal{N}$
6. Go to 1

Primal simplex algorithm: Choose a column

Assume $\widehat{\boldsymbol{b}} \geq \mathbf{0}$ Seek $\widehat{\boldsymbol{c}}_N \geq \mathbf{0}$

Scan $\widehat{c}_i < 0$ for q to leave $\mathcal{N}$

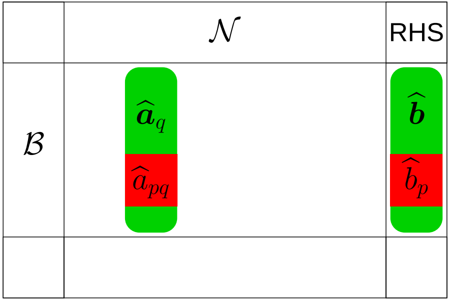


Primal simplex algorithm: Choose a row

Assume $\widehat{\boldsymbol{b}} \geq \mathbf{0}$ Seek $\widehat{\boldsymbol{c}}_N \geq \mathbf{0}$

Scan $\widehat{c}_j < 0$ for q to leave $\mathcal{N}$

Scan $\widehat{b}_i/\widehat{a}_{iq} > 0$ for p to leave $\mathcal{B}$



Primal simplex algorithm: Update cost and RHS

Assume $\widehat{\boldsymbol{b}} \geq \mathbf{0}$ Seek $\widehat{\boldsymbol{c}}_N \geq \mathbf{0}$

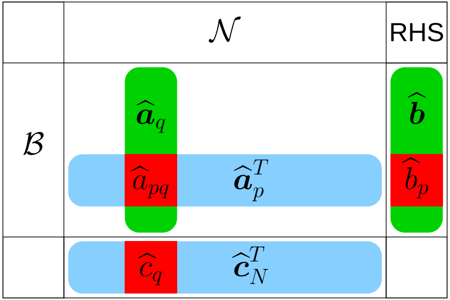
Scan $\widehat{c}_j < 0$ for q to leave $\mathcal{N}$

Scan $\widehat{b}_i/\widehat{a}_{iq} > 0$ for p to leave $\mathcal{B}$

Update: Exchange p and q between $\mathcal{B}$ and $\mathcal{N}$

Update $\widehat{\boldsymbol{b}} := \widehat{\boldsymbol{b}} - \alpha_P \widehat{\boldsymbol{a}}_q$ $\qquad \alpha_P = \widehat{b}_p/\widehat{a}_{pq}$

Update $\widehat{\boldsymbol{c}}_N^T := \widehat{\boldsymbol{c}}_N^T + \alpha_D \widehat{\boldsymbol{a}}_p^T$ $\qquad \alpha_D = -\widehat{c}_q/\widehat{a}_{pq}$



Primal simplex algorithm: Data required

Assume $\widehat{\boldsymbol{b}} \geq \boldsymbol{0}$ Seek $\widehat{\boldsymbol{c}}_N \geq \boldsymbol{0}$

Scan $\widehat{c}_j < 0$ for q to leave $\mathcal{N}$

Scan $\widehat{b}_i / \widehat{a}_{iq} > 0$ for p to leave $\mathcal{B}$

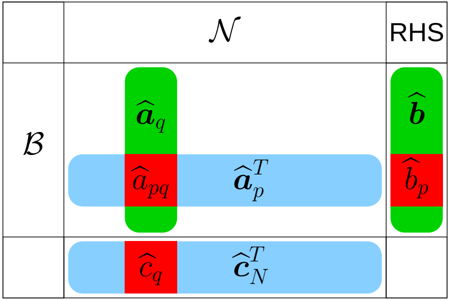
Update: Exchange p and q between $\mathcal{B}$ and $\mathcal{N}$

Update $\widehat{\boldsymbol{b}} := \widehat{\boldsymbol{b}} - \alpha_P \widehat{\boldsymbol{a}}_q$ $\quad \alpha_P = \widehat{b}_p / \widehat{a}_{pq}$

Update $\widehat{\boldsymbol{c}}_N^T := \widehat{\boldsymbol{c}}_N^T + \alpha_D \widehat{\boldsymbol{a}}_p^T$ $\quad \alpha_D = -\widehat{c}_q / \widehat{a}_{pq}$

Data required

- Pivotal row $\widehat{\boldsymbol{a}}_p^T = \boldsymbol{e}_p^T B^{-1} N$
- Pivotal column $\widehat{\boldsymbol{a}}_q = B^{-1} \boldsymbol{a}_q$

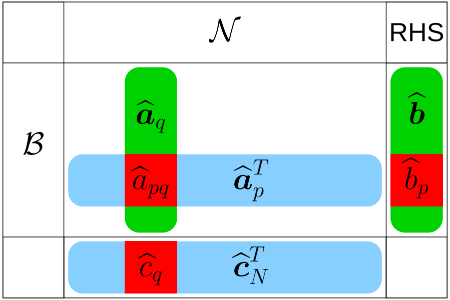


Primal simplex algorithm

Assume $\widehat{\boldsymbol{b}} \geq \boldsymbol{0}$ Seek $\widehat{\boldsymbol{c}}_N \geq \boldsymbol{0}$

Scan $\widehat{c}_j < 0$ for q to leave $\mathcal{N}$

Scan $\widehat{b}_i / \widehat{a}_{iq} > 0$ for p to leave $\mathcal{B}$



Update: Exchange p and q between $\mathcal{B}$ and $\mathcal{N}$

Update $\widehat{\boldsymbol{b}} := \widehat{\boldsymbol{b}} - \alpha_P \widehat{\boldsymbol{a}}_q$ $\qquad \alpha_P = \widehat{b}_p / \widehat{a}_{pq}$

Update $\widehat{\boldsymbol{c}}_N^T := \widehat{\boldsymbol{c}}_N^T + \alpha_D \widehat{\boldsymbol{a}}_p^T$ $\qquad \alpha_D = -\widehat{c}_q / \widehat{a}_{pq}$

Data required

- Pivotal row $\widehat{\boldsymbol{a}}_p^T = \boldsymbol{e}_p^T B^{-1} N$
- Pivotal column $\widehat{\boldsymbol{a}}_q = B^{-1} \boldsymbol{a}_q$

Why does it work?

Objective improves by $-\dfrac{\widehat{b}_p \times \widehat{c}_q}{\widehat{a}_{pq}}$ each iteration

Simplex method: Computation

Standard simplex method (SSM): Major computational component

	$\mathcal{N}$	RHS
$\mathcal{B}$	$\widehat{N}$	$\widehat{\boldsymbol{b}}$
	$\widehat{\boldsymbol{c}}_N^T$	

Update of tableau: $\widehat{N} := \widehat{N} - \dfrac{1}{\widehat{a}_{pq}}\widehat{\boldsymbol{a}}_q\widehat{\boldsymbol{a}}_p^T$

where $\widehat{N} = B^{-1}N$

- Hopelessly inefficient for sparse LP problems
- Prohibitively expensive for large LP problems

Revised simplex method (RSM): Major computational components

Pivotal row via	$B^T\boldsymbol{\pi}_p = \boldsymbol{e}_p$	BTRAN	and	$\widehat{\boldsymbol{a}}_p^T = \boldsymbol{\pi}_p^T N$	PRICE
Pivotal column via	$B\,\widehat{\boldsymbol{a}}_q = \boldsymbol{a}_q$	FTRAN		Represent B^{-1}	INVERT

Update B^{-1} exploiting $\bar{B} = B + (\boldsymbol{a}_q - B\boldsymbol{e}_p)\boldsymbol{e}_p^T$ UPDATE

Serial simplex: Hyper-sparsity

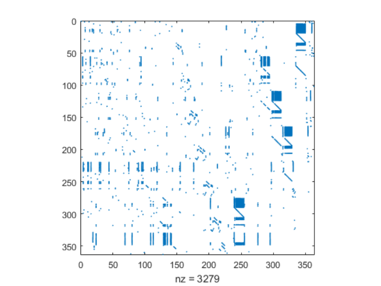
Serial simplex: Solve $B\boldsymbol{x} = \boldsymbol{r}$ for sparse $\boldsymbol{r}$

- Given $B = LU$, solve
$$L\boldsymbol{y} = \boldsymbol{r}; \quad U\boldsymbol{x} = \boldsymbol{y}$$
- In revised simplex method, $\boldsymbol{r}$ is sparse: consequences?
  - If B is irreducible then $\boldsymbol{x}$ is full
  - If B is highly reducible then $\boldsymbol{x}$ can be **sparse**
- Phenomenon of **hyper-sparsity**
  - Exploit it when *forming* $\boldsymbol{x}$
  - Exploit it when *using* $\boldsymbol{x}$

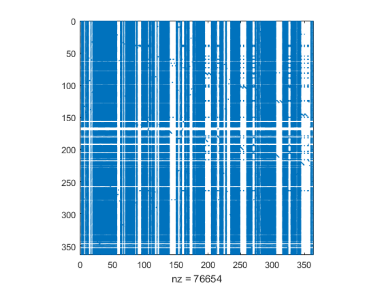
Serial simplex: Hyper-sparsity

Inverse of a sparse matrix and solution of $B\boldsymbol{x} = \boldsymbol{r}$

Optimal B for LP problem `stair`



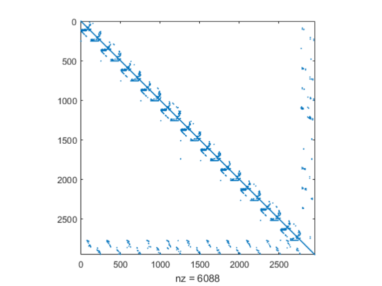
B^{-1} has density of 58%, so $B^{-1}\boldsymbol{r}$ is typically dense



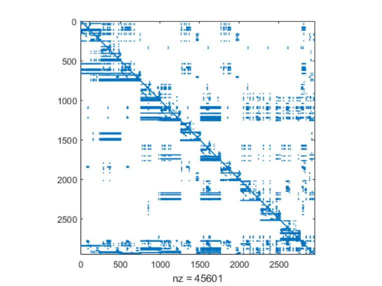
Serial simplex: Hyper-sparsity

Inverse of a sparse matrix and solution of $B\boldsymbol{x} = \boldsymbol{r}$

Optimal B for LP problem `pds-02`



B^{-1} has density of 0.52%, so $B^{-1}\boldsymbol{r}$ is typically sparse—when $\boldsymbol{r}$ is sparse



Serial simplex: Hyper-sparsity

- Use solution of $Lx = \boldsymbol{b}$
  - To illustrate the phenomenon of hyper-sparsity
  - To demonstrate how to exploit hyper-sparsity
- Apply principles to other triangular solves in the simplex method

Serial simplex: Hyper-sparsity

Recall: Solve $L\boldsymbol{x} = \boldsymbol{b}$ using

function $\texttt{ftranL}(L, \boldsymbol{b}, \boldsymbol{x})$

$\boldsymbol{r} = \boldsymbol{b}$

for all $j \in \{1, \ldots, m\}$ **do**

  for all $i : L_{ij} \neq 0$ **do**

    $r_i = r_i - L_{ij} r_j$

$\boldsymbol{x} = \boldsymbol{r}$

When $\boldsymbol{b}$ is **sparse**

- Inefficient until $\boldsymbol{r}$ fills in

Serial simplex: Hyper-sparsity

Better: Check r_j for zero

function `ftranL`(L, $\boldsymbol{b}$, $\boldsymbol{x}$)

$\boldsymbol{r} = \boldsymbol{b}$

for all $j \in \{1, \ldots, m\}$ **do**

  if $r_j \neq 0$ **then**

    for all $i : L_{ij} \neq 0$ **do**

      $r_i = r_i - L_{ij} r_j$

$\boldsymbol{x} = \boldsymbol{r}$

When $\boldsymbol{x}$ is **sparse**

- Few values of r_j are nonzero
- Check for zero dominates
- Requires more efficient identification of set $\mathcal{X}$ of indices j such that $r_j \neq 0$

Gilbert and Peierls (1988)
H and McKinnon (1998–2005)

Serial simplex: Hyper-sparsity

Recall: major computational components

- FTRAN: Form $\hat{\mathbf{a}}_q = B^{-1}\mathbf{a}_q$
- BTRAN: Form $\boldsymbol{\pi}_p = B^{-T}\mathbf{e}_p$
- PRICE: Form $\hat{\mathbf{a}}_p^T = \boldsymbol{\pi}_p^T N$

BTRAN: Form $\boldsymbol{\pi}_p = B^{-T}\mathbf{e}_p$

- Transposed triangular solves
- $L^T\mathbf{x} = \mathbf{b}$ has $x_i = b_i - \mathbf{l}_i^T\mathbf{x}$
  - Hyper-sparsity: $\mathbf{l}_i^T\mathbf{x}$ typically zero
  - Also store L (and U) row-wise and use FTRAN code

PRICE: Form $\hat{\mathbf{a}}_p^T = \boldsymbol{\pi}_p^T N$

- Hyper-sparsity: $\boldsymbol{\pi}_p^T$ is sparse
- Store N row-wise
- Form $\hat{\mathbf{a}}_p^T$ as a combination of rows of N for nonzeros in $\boldsymbol{\pi}_p^T$

H and McKinnon (1998–2005)

COAP best paper prize (2005)

Interior point methods

Interior point methods: Traditional

Replace $\boldsymbol{x} \geq 0$ by **log barrier** function and solve

$$\text{maximize} \quad f = \boldsymbol{c}^T\boldsymbol{x} + \mu \sum_{j=1}^{n} \ln(x_j) \quad \text{such that} \quad A\boldsymbol{x} = \boldsymbol{b}$$

- For small μ this has the same solution as the LP
- Solve for a decreasing sequence of values of μ, moving through *interior* of K
- Perform a small number of (expensive) iterations: each solves

$$\begin{bmatrix} -\Theta^{-1} & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\Delta x} \\ \boldsymbol{\Delta y} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{d} \end{bmatrix} \iff G\boldsymbol{\Delta y} = \boldsymbol{h}$$

  where $\boldsymbol{\Delta x}$ and $\boldsymbol{\Delta y}$ are steps in the primal and dual variables and $G = A\Theta A^T$
- Standard technique is to form the **Cholesky** decomposition $G = LL^T$ and perform triangular solves with L

Interior point methods: Traditional

Forming the **Cholesky** decomposition $G = LL^T$ and perform triangular solves with L

- $G = A\Theta A^T = \sum_j \theta_j \boldsymbol{a}_j \boldsymbol{a}_j^T$ is generally sparse
  So long as dense columns of A are treated carefully
- Much effort has gone into developing efficient serial Cholesky codes
- Parallel codes exist: notably for (nested) block structured problems
  `OOPS` solved a QP with 10^9 variables
  Gondzio and Grothey (2006)
- **Disadvantage:** L can fill-in
  Cholesky can be prohibitively expensive for large n

Interior point methods: Matrix-free

Alternative approach to Cholesky: solve $G\boldsymbol{\Delta y} = \boldsymbol{h}$ using an **iterative** method

- Use **preconditioned conjugate gradient method** (PCG)
- For preconditioner, consider $G = \begin{bmatrix} L_{11} & \\ L_{21} & I \end{bmatrix} \begin{bmatrix} D_L & \\ & S \end{bmatrix} \begin{bmatrix} L_{11}^T & L_{21}^T \\ & I \end{bmatrix}$ where
  - $L = \begin{bmatrix} L_{11} \\ L_{21} \end{bmatrix}$ contains the first k columns of the Cholesky factor of G
  - D_L is a diagonal matrix formed by the k largest pivots of G
  - S is the Schur complement after k pivots
- Precondition $G\boldsymbol{\Delta y} = \boldsymbol{h}$ using $P = \begin{bmatrix} L_{11} & \\ L_{21} & I \end{bmatrix} \begin{bmatrix} D_L & \\ & D_S \end{bmatrix} \begin{bmatrix} L_{11}^T & L_{21}^T \\ & I \end{bmatrix}$ where
  - S_D is the diagonal of S
  - Avoids computing S or even G!

Gondzio (2009)

Interior point methods: Matrix-free

- Can solve problems intractable using direct methods
- Only requires "oracle" returning $\boldsymbol{y} = A\boldsymbol{x}$

Gondzio *et al.* (2014)

- Matrix-free IPM beats first order methods on speed and reliability for
  - ℓ_1-regularized sparse least-squares: $n = O(10^{12})$
  - ℓ_1-regularized logistic regression: $n = O(10^4 - 10^7)$
- How?
  - Preconditioner P is **diagonal**
  - $A\Theta A^T$ is near-diagonal!
- Says much about the "difficulty" of such problems!

Fountoulakis and Gondzio (2016)

- **Disadvantage:** Not useful for all problems!

Linear Programming solvers: software

Solvers

Commercial

- Xpress
- Gurobi
- Cplex
- Mosek
- SAS
- Matlab

Open-source

- Clp (COIN-OR)
- HiGHS
- Glop (Google)
- Soplex (ZIB)
- Glpk (GNU)
- Lpsolve

Simplex solvers

Solver	Gurobi	Xpress	Clp	Cplex	Mosek
Time	1	1.0	1.9	1.9	5.1

Mittelmann (25 April 2018)

Solver	Clp	Mosek	SAS	HiGHS	Glop	Matlab	Soplex	Glpk	Lpsolve
Time	1	2.8	3.2	5.3	6.4	6.6	10.1	26	112

Interior point solvers

Solver	Mosek	bpmpd	SAS	Matlab	Clp
Time	1	2.6	3.5	3.6	9.7

Parallel simplex for structured LP problems

PIPS-S

Overview

- Written in C++ to solve stochastic MIP relaxations in parallel
- Dual simplex
- Based on NLA routines in `Clp`
- Product form update

Concept

- Exploit data parallelism due to block structure of LPs
- Distribute problem over processes

Paper: Lubin, H, Petra and Anitescu (2013)

- COIN-OR INFORMS 2013 Cup
- COAP best paper prize (2013)

PIPS-S: Stochastic MIP problems

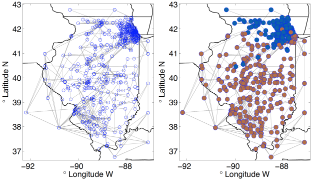
Two-stage stochastic LPs have column-linked block angular (BALP) structure

$$
\begin{array}{rccccccccccc}
\text{minimize} & \mathbf{c}_0^T\mathbf{x}_0 & + & \mathbf{c}_1^T\mathbf{x}_1 & + & \mathbf{c}_2^T\mathbf{x}_2 & + & \dots & + & \mathbf{c}_N^T\mathbf{x}_N & & \\
\text{subject to} & A\mathbf{x}_0 & & & & & & & & & = & \mathbf{b}_0 \\
& T_1\mathbf{x}_0 & + & W_1\mathbf{x}_1 & & & & & & & = & \mathbf{b}_1 \\
& T_2\mathbf{x}_0 & & & + & W_2\mathbf{x}_2 & & & & & = & \mathbf{b}_2 \\
& \vdots & & & & & & \ddots & & & & \vdots \\
& T_N\mathbf{x}_0 & & & & & & & + & W_N\mathbf{x}_N & = & \mathbf{b}_N \\
& \mathbf{x}_0 \geq \mathbf{0} & & \mathbf{x}_1 \geq \mathbf{0} & & \mathbf{x}_2 \geq \mathbf{0} & & \dots & & \mathbf{x}_N \geq \mathbf{0} & &
\end{array}
$$

- Variables $\mathbf{x}_0 \in \mathbb{R}^{n_0}$ are **first stage** decisions
- Variables $\mathbf{x}_i \in \mathbb{R}^{n_i}$ for $i = 1, \dots, N$ are **second stage** decisions
  Each corresponds to a **scenario** which occurs with modelled probability
- The objective is the expected cost of the decisions
- In stochastic MIP problems, some/all decisions are discrete

PIPS-S: Stochastic MIP problems

- Power systems optimization project at Argonne
- Integer second-stage decisions
- Stochasticity from wind generation
- Solution via branch-and-bound
    - Solve root using parallel IPM solver PIPS
      Lubin, Petra *et al.* (2011)
    - Solve nodes using parallel dual simplex solver PIPS-S



PIPS-S: Exploiting problem structure

Convenient to permute the LP thus:

$$
\begin{array}{rccccccccccc}
\text{minimize} & \boldsymbol{c}_1^T\boldsymbol{x}_1 & + & \boldsymbol{c}_2^T\boldsymbol{x}_2 & + & \dots & + & \boldsymbol{c}_N^T\boldsymbol{x}_N & + & \boldsymbol{c}_0^T\boldsymbol{x}_0 & & \\
\text{subject to} & W_1\boldsymbol{x}_1 & & & & & & & + & T_1\boldsymbol{x}_0 & = & \boldsymbol{b}_1 \\
 & & & W_2\boldsymbol{x}_2 & & & & & + & T_2\boldsymbol{x}_0 & = & \boldsymbol{b}_2 \\
 & & & & & \ddots & & & \vdots & & & \vdots \\
 & & & & & & & W_N\boldsymbol{x}_N & + & T_N\boldsymbol{x}_0 & = & \boldsymbol{b}_N \\
 & & & & & & & & & A\boldsymbol{x}_0 & = & \boldsymbol{b}_0 \\
 & \boldsymbol{x}_1 \geq \boldsymbol{0} & & \boldsymbol{x}_2 \geq \boldsymbol{0} & & \dots & & \boldsymbol{x}_N \geq \boldsymbol{0} & & \boldsymbol{x}_0 \geq \boldsymbol{0} & &
\end{array}
$$

PIPS-S: Exploiting problem structure

- Inversion of the basis matrix B is key to revised simplex efficiency

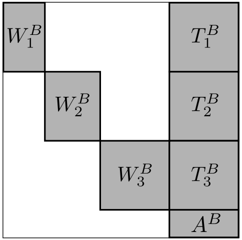
$$B = \begin{bmatrix} W_1^B & & & T_1^B \\ & \ddots & & \vdots \\ & & W_N^B & T_N^B \\ & & & A^B \end{bmatrix}$$

- W_i^B are columns corresponding to n_i^B basic variables in scenario i
- $\begin{bmatrix} T_1^B \\ \vdots \\ T_N^B \\ A^B \end{bmatrix}$ are columns corresponding to n_0^B basic first stage decisions

PIPS-S: Exploiting problem structure

- Inversion of the basis matrix B is key to revised simplex efficiency

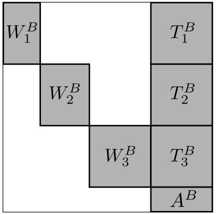
$$B = \begin{bmatrix} W_1^B & & & T_1^B \\ & \ddots & & \vdots \\ & & W_N^B & T_N^B \\ & & & A^B \end{bmatrix}$$



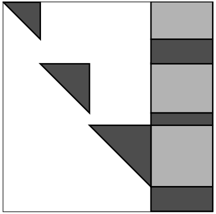
- B is nonsingular so
  - W_i^B are "tall": full column rank
  - $\begin{bmatrix} W_i^B & T_i^B \end{bmatrix}$ are "wide": full row rank
  - A^B is "wide": full row rank

- Scope for parallel inversion is immediate and well known

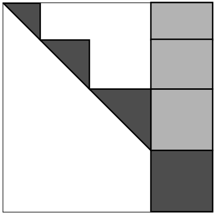
PIPS-S: Exploiting problem structure



- Eliminate sub-diagonal entries in each W_i^B (independently)



- Apply elimination operations to each T_i^B (independently)



- Accumulate non-pivoted rows from the W_i^B with A^B and complete elimination

PIPS-S: Overview

Scope for parallelism

- Parallel Gaussian elimination yields **block LU** decomposition of B
- Scope for parallelism in block forward and block backward substitution
- Scope for parallelism in `PRICE`

Implementation

- Distribute problem data over processes
- Perform data-parallel `BTRAN`, `FTRAN` and `PRICE` over processes
- Used MPI

Lubin, H, Petra and Anitescu (2013)
COIN-OR INFORMS 2013 Cup
COAP best paper prize (2013)

PIPS-S: Results

On Fusion cluster: Performance relative to Clp

Dimension	Cores	Storm	SSN	UC12	UC24
$m+n=O(10^6)$	1	0.34	0.22	0.17	0.08
	32	8.5	6.5	2.4	0.7
$m+n=O(10^7)$	256	299	45	67	68

On Blue Gene

- Instance of UC12
- $m+n=O(10^8)$
- Requires 1 TB of RAM
- Runs from an advanced basis

Cores	Iterations	Time (h)	Iter/sec
1024	Exceeded execution time limit		
2048	82,638	6.14	3.74
4096	75,732	5.03	4.18
8192	86,439	4.67	5.14

Parallel simplex for general LP problems

HiGHS: Past (2011–2014)

Overview

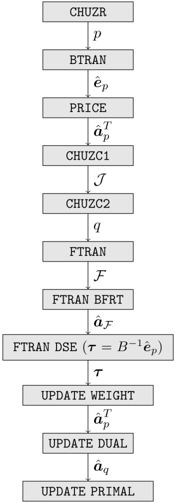
- Written in C++ to study parallel simplex
- Dual simplex with standard algorithmic enhancements
- Efficient numerical linear algebra
- No interface or utilities

Concept

- High performance serial solver (`hsol`)
- Exploit limited task and data parallelism in standard dual RSM iterations (`sip`)
- Exploit greater task and data parallelism via minor iterations of dual SSM (`pami`)

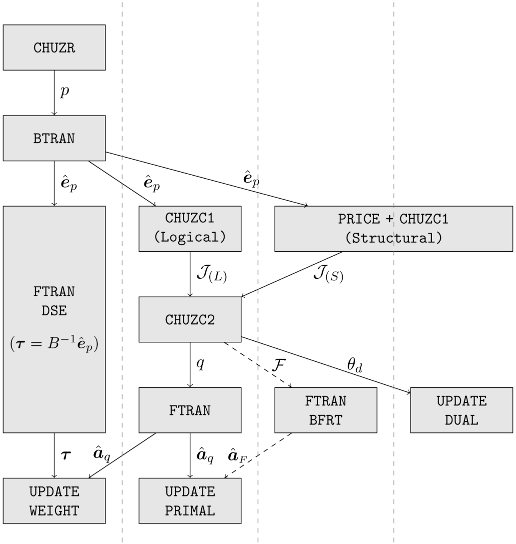
Huangfu and H

HiGHS: Single iteration parallelism with sip option



- Computational components appear sequential
- Each has highly-tuned sparsity-exploiting serial implementation
- Exploit “slack” in data dependencies

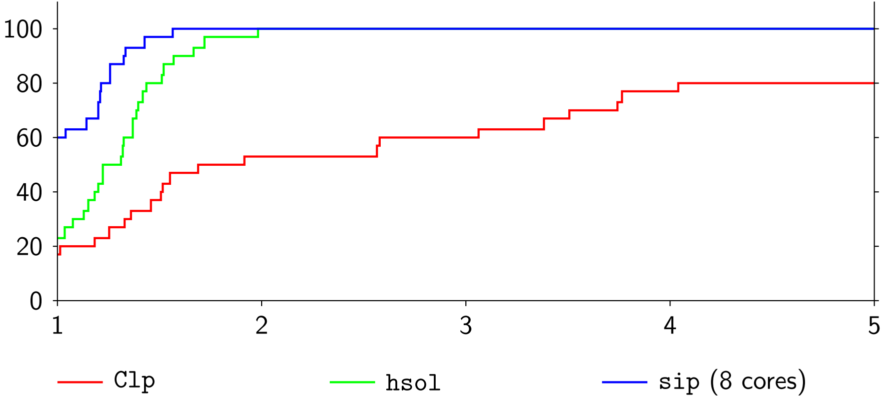
HiGHS: Single iteration parallelism with sip option



- Parallel PRICE to form $\hat{a}_p^T = \pi_p^T N$
- Other computational components serial
- Overlap any independent calculations
- Only four worthwhile threads unless $n \gg m$ so PRICE dominates
- More than Bixby and Martin (2000)
- Better than Forrest (2012)

Huangfu and H (2014)

HiGHS: Clp vs HiGHS vs sip

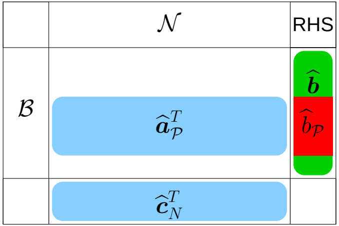


Performance on spectrum of 30 significant LP test problems

- sip on 8 cores is 1.15 times faster than HiGHS
- HiGHS (sip on 8 cores) is 2.29 (2.64) times faster than Clp

HiGHS: Multiple iteration parallelism with pami option

- Perform standard dual simplex minor iterations for rows in set $\mathcal{P}$ ($|\mathcal{P}| \ll m$)
- Suggested by Rosander (1975) but never implemented efficiently *in serial*



- Task-parallel multiple BTRAN to form $\boldsymbol{\pi}_{\mathcal{P}} = B^{-T}\boldsymbol{e}_{\mathcal{P}}$
- Data-parallel PRICE to form $\widehat{\boldsymbol{a}}_{p}^{T}$ (as required)
- Task-parallel multiple FTRAN for primal, dual and weight updates

Huangfu and H (2011–2014)
COAP best paper prize (2015)
MPC best paper prize (2018)

`HiGHS`: Performance and reliability

Extended testing using 159 test problems

- 98 Netlib
- 16 Kennington
- 4 Industrial
- 41 Mittelmann

Exclude 7 which are "hard"

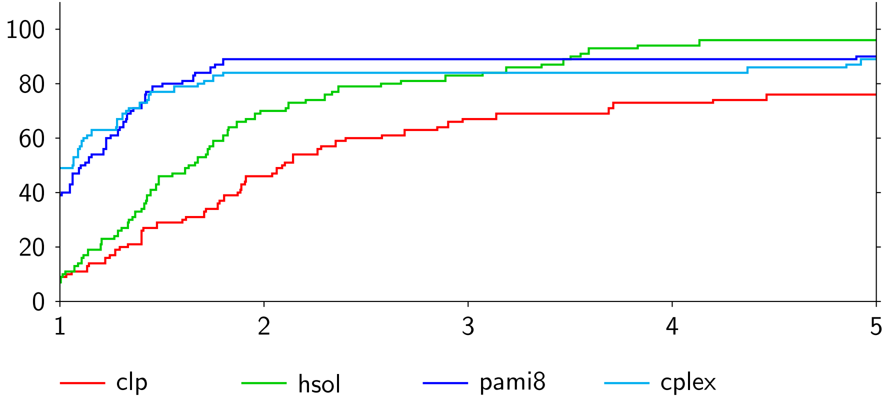
Performance

Benchmark against `clp` (v1.16) and `cplex` (v12.5)

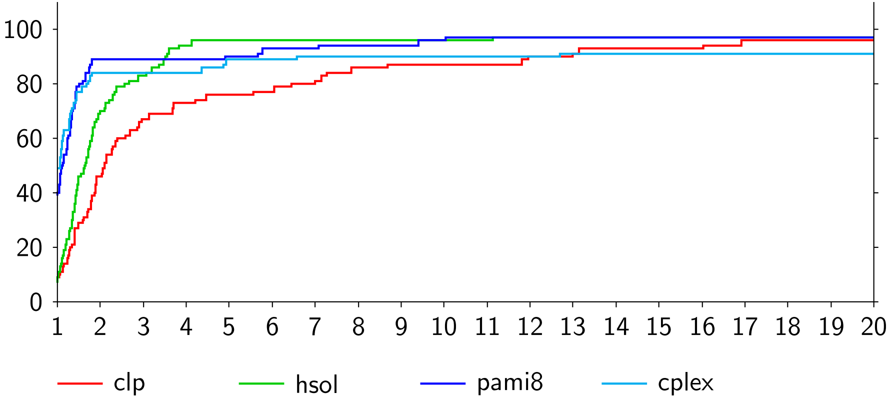
- Dual simplex
- No presolve
- No crash

Ignore results for 82 LPs with minimum solution time below 0.1s

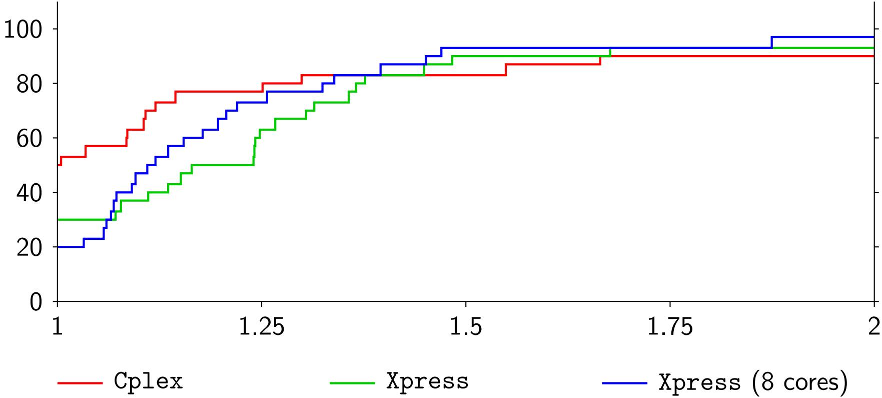
HiGHS: Performance



HiGHS: Reliability



HiGHS: Impact



- pami ideas incorporated in FICO Xpress (Huangfu 2014)
- Xpress has been the fastest simplex solver for most of the past five years

HiGHS: an open-source high-performance linear optimizer

HiGHS: Present (2016–date)

Features

- Model management: Add/delete/modify problem data
- Interfaces

Presolve

- Presolve (and corresponding postsolve) has been implemented efficiently

  Remove redundancies in the LP to reduce problem dimension

  Galabova

Crash

- Dual simplex "triangular basis" crash
- Alternative crash techniques being studied

  H and Galabova

Interior point method

- Reliable "Matrix-free" implementation: Solve normal equations iteratively

  Schork

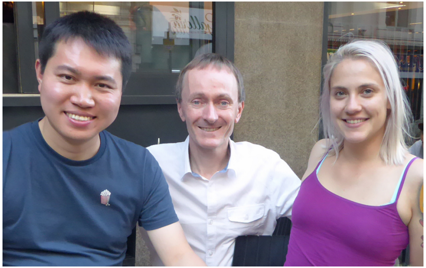
HiGHS: The team

What's in a name?

HiGHS: **H**all, **i**vet **G**alabova, **H**uangfu and **S**chork

Team HiGHS

- Julian Hall: Reader (1990–date)
- Ivet Galabova
  - PhD (2016–date)
  - Google (2018)
- Qi Huangfu
  - PhD (2009–2013)
  - FICO Xpress (2013–2018)
  - MSc (2018–date)
- Lukas Schork: PhD (2015–2018)
- Michael Feldmeier: PhD (2018–date)



HiGHS: Access

Availability

- Open source (MIT license)
- GitHub: ERGO-Code/HiGHS
- COIN-OR: Replacement for `Clp`?

Interfaces

- Existing
  - C++ `HiGHS` class
  - Load from `.mps`
  - Load from `.lp`
  - OSI (almost!)
  - SCIP (almost!)
- Prototypes
  - `Python`
  - `FORTRAN`
  - `GAMS`
  - `Julia`
- Planned
  - `AMPL`
  - `MATLAB`
  - `R`

A novel method: Fast approximate solution of LP problems

Fast approximate solution of LP problems

- **Aim:** Get an approximate solution of an LP problem faster than simplex or interior point methods
- **What for?**
  - Advanced start for the simplex method
  - Fast approximate solution may be good enough!

"Idiot" crash (Forrest)

For $j = 1, \ldots, n$ (repeatedly)

$$\text{Solve} \quad \min g_j(\delta) = \mu(c_j + \sum_{i=1}^{m} a_{ij}\lambda_i)\delta + \sum_{i=1}^{m}(r_i + a_{ij}\delta)^2 \quad \text{where } r_i = \boldsymbol{a}_i^T \boldsymbol{x} - b_i$$

$$\text{Set} \quad x_j := \max(0, x_j + \delta)$$

Modify μ and $\boldsymbol{\lambda}$ "intelligently" and hope that $\boldsymbol{x}$ converges to something useful!

Idiot crash: Application to quadratic assignment problem linearizations

Results: Performance after (up to) 200 Idiot iterations

Model	Rows	Columns	Optimum	Residual	Objective	Error	Time
NUG05	210	225	50.00	9.4×10^{-9}	50.01	1.5×10^{-4}	0.04
NUG06	372	486	86.00	7.8×10^{-9}	86.01	1.2×10^{-4}	0.11
NUG07	602	931	148.00	7.9×10^{-9}	148.64	4.3×10^{-3}	0.25
NUG08	912	1613	203.50	7.0×10^{-9}	204.41	4.5×10^{-3}	0.47
NUG12	3192	8856	522.89	8.8×10^{-9}	523.86	1.8×10^{-3}	2.58
NUG15	6330	22275	1041.00	8.9×10^{-9}	1041.38	3.7×10^{-4}	5.13
NUG20	15240	72600	2182.00	7.5×10^{-9}	2183.03	4.7×10^{-4}	14.94
NUG30	52260	379350	4805.00	1.1×10^{-8}	4811.41	1.3×10^{-3}	82.28

- Solution of NUG30 intractable using simplex or IPM on the same machine
- Idiot crash consistently yields near-optimal solutions

Fast approximate solution of LP problems

Idiot crash: Performance

For a few problems, notably QAP linearizations, $\boldsymbol{x} \to \boldsymbol{x}^c \approx \boldsymbol{x}^*$

- No proof of near-optimality when $\boldsymbol{x}^c \approx \boldsymbol{x}^*$
- Great advanced start for simplex (`Clp`)

H and Galabova (2018)

Future aims

- Apply to dual LP to give confidence interval for $\boldsymbol{x}^c \approx \boldsymbol{x}^*$
- Aim to develop more successful algorithms for fast approximate solution of LPs

To close

Conclusions

- LP solvers crucial to decision-making
- Classical methods very highly developed
- Look for alternative algorithms for fast (approximate) solution of LPs

Slides:
http://www.maths.ed.ac.uk/hall/Tokyo19

Code:
https://github.com/ERGO-Code/HiGHS

I. L. Galabova and J. A. J. Hall.
A quadratic penalty algorithm for linear programming and its application to linearizations of quadratic assignment problems.
Technical Report ERGO-18-009, School of Mathematics, University of Edinburgh, 2018.

J. A. J. Hall and K. I. M. McKinnon.
Hyper-sparsity in the revised simplex method and how to exploit it.
Computational Optimization and Applications, 32(3):259–283, December 2005.

Q. Huangfu and J. A. J. Hall.
Parallelizing the dual revised simplex method.
Mathematical Programming Computation, 10(1):119–142, 2018.

M. Lubin, J. A. J. Hall, C. G. Petra, and M. Anitescu.
Parallel distributed-memory simplex for large-scale stochastic LP problems.
Computational Optimization and Applications, 55(3):571–596, 2013.

L. Schork and J. Gondzio.
Implementation of an interior point method with basis preconditioning.
Technical Report ERGO-18-014, School of Mathematics, University of Edinburgh, 2018.