

High performance solution of large-scale linear programming problems

Julian Hall

School of Mathematics
University of Edinburgh

jajhall@ed.ac.uk

Stirling

1 November 2019



THE UNIVERSITY
of EDINBURGH



- Introduction
 - What is LP?
 - Simplex method
 - Computational requirements
- Exploiting hyper-sparsity
- Impact
- Exploiting parallelism
- HiGHS

What is linear programming?



The diet problem

- Given
 - Costs of foodstuffs
 - Nutrient information and daily requirements
- What is the cheapest diet?

A linear programming model

- Let x_j be the amount of food j purchased ($x_j \geq 0$), for $j = 1, \dots, n$
- Cost c_j of food j gives total cost $f = \sum_{j=1}^n c_j x_j$ to be minimized
- Let a_{ij} be the amount of nutrient i in food j

Matching the daily requirement b_i gives $\sum_{j=1}^n a_{ij} x_j = b_i$, for $i = 1, \dots, m$

The general LP problem

- Given decision variables x_j for $j = 1, \dots, n$

- Costs c_j , $j = 1, \dots, n$ give **objective function** $f = \sum_{j=1}^n c_j x_j$

- **Constraints** are $\sum_{j=1}^n a_{ij} x_j = b_i$, $i = 1, \dots, m$

- **Bounds** are $x_j \geq 0$, $j = 1, \dots, n$

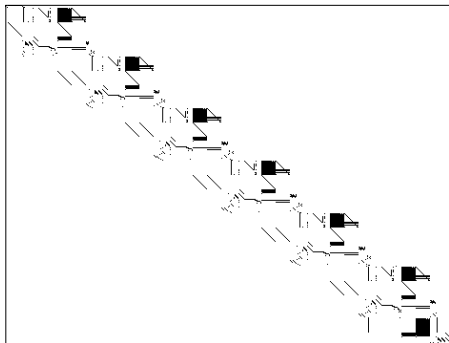
- General LP problem is

$$\text{maximize } f = \mathbf{c}^T \mathbf{x} \quad \text{such that} \quad A\mathbf{x} = \mathbf{b} \quad \text{and} \quad \mathbf{x} \geq \mathbf{0}$$

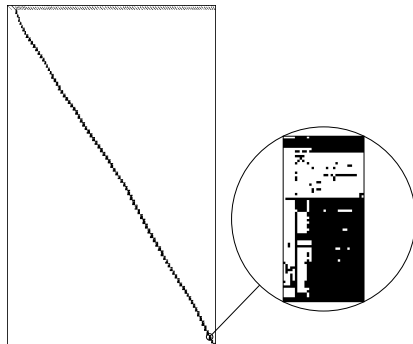
- Can assume $m < n$ and A has full rank
- For problems of practical interest, A is usually sparse

Practical LP problems

$$\text{maximize } f = \mathbf{c}^T \mathbf{x} \quad \text{such that} \quad \mathbf{Ax} = \mathbf{b} \quad \text{and} \quad \mathbf{x} \geq \mathbf{0}$$



STAIR: 356 rows; 467 columns; 3856 nonzeros



DCP1: 4950 rows; 3007 columns; 93853 nonzeros

Large-scale practical LP problems have $O(10^7)$ variables and constraints

Solving LP problems: The feasible region

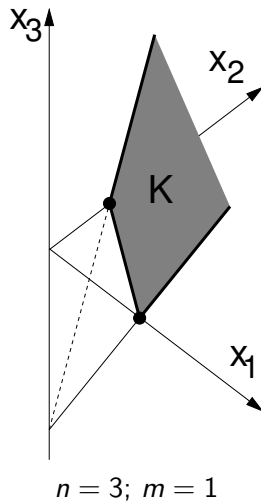
General LP problem is

maximize $f = \mathbf{c}^T \mathbf{x}$ such that $A\mathbf{x} = \mathbf{b}$ and $\mathbf{x} \geq \mathbf{0}$

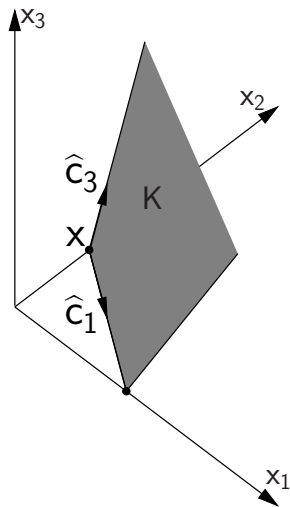
with m equations and n variables ($m < n$)

The feasible region

- Solution of $A\mathbf{x} = \mathbf{b}$ is a $n - m$ dimensional **hyperplane** in $\mathbb{R}^n$
- Intersection with $\mathbf{x} \geq \mathbf{0}$ is the **feasible region** K
 - K is a **polyhedron**
 - A vertex has
 - $n - m$ zero components
 - m components given by $A\mathbf{x} = \mathbf{b}$
- A solution of the LP occurs at a **vertex** of K



The simplex algorithm: Choosing where to move

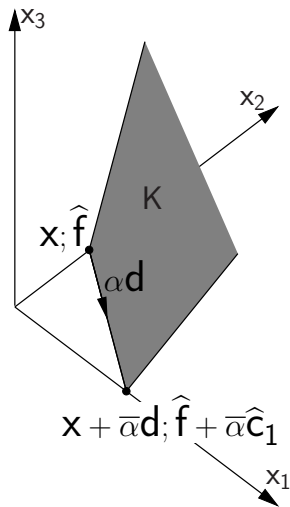


General LP problem is

$$\text{maximize } f = \mathbf{c}^T \mathbf{x} \quad \text{such that} \quad A\mathbf{x} = \mathbf{b} \quad \text{and} \quad \mathbf{x} \geq \mathbf{0}$$

- At a vertex $\mathbf{x}$, consider moving along an edge of K to improve the value of f
 - Corresponds to increasing a variable x_j from zero
 - Rate of change $\hat{c}_j$ of f with x_j is known
- If no positive $\hat{c}_j$ then $\mathbf{x}$ is optimal so **stop**
- Increase variable x_q with greatest $\hat{c}_j$

The simplex algorithm: Choosing how far to go



- Increase variable x_q with greatest $\hat{c}_j$
- Edge direction $\mathbf{d}$ corresponding x_q is known
- Move along $\mathbf{x} + \alpha \mathbf{d}$
- Stop when (first) component p of $\mathbf{x} + \alpha \mathbf{d}$ is zeroed
- Objective increases by $\bar{\alpha} \hat{c}_q > 0$
- Repeat!

The simplex algorithm: Computational requirements

Each simplex iteration requires

- Values of $\hat{c}_j$ for all zeroed variables
- Edge direction $\mathbf{d}$ corresponding to x_q

The revised simplex method

- Matrix A is generally **sparse**
- Find data for simplex algorithm by
 - Solving square systems of equations
 - Forming matrix-vector products
- Coefficient matrix B for systems is a square nonsingular submatrix of A
- Columns of A not in B form rectangular matrix N

Each iteration

- Solve $B \hat{\mathbf{a}}_q = \mathbf{a}_q$
- Solve $B^T \boldsymbol{\pi}_p = \mathbf{e}_p$
- Update $\hat{\mathbf{c}}$ using $\hat{\mathbf{a}}_p^T = \boldsymbol{\pi}_p^T N$
- Column p of B replaced by $\mathbf{a}_q$ to give $\bar{B} = B + (\mathbf{a}_q - B\mathbf{e}_p)\mathbf{e}_p^T$

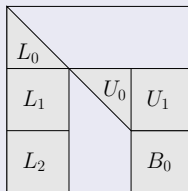
NLA challenge

- Form $PBQ = LU$
 - Solve $B\mathbf{x} = \mathbf{b}$ for **sparse** $\mathbf{b}$
- Solve $\bar{B}\mathbf{x} = \mathbf{b}$

NLA Challenge: Form $PBQ = LU$

Triangularisation

Identify row and column singletons until every active nonzero has positive Markowitz merit



Solve $B\mathbf{x} = \mathbf{r}$ as

$$L_0\mathbf{x}_L = \mathbf{r}_L$$

$$B_0\mathbf{x}_0 = \mathbf{r}_0 - L_2\mathbf{x}_L$$

$$U_0\mathbf{x}_U = \mathbf{r}_U - L_1\mathbf{x}_L - U_1\mathbf{x}_0$$

- Matrix B is typically **highly reducible**: $\dim(B_0) \ll m$
- For network flow problems B is provably triangularisable

NLA Challenge: Solve $\bar{B}\mathbf{x} = \mathbf{r}$ using the product form update (PFI)

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

- Column p of B replaced by $\mathbf{a}_q$ to give $\bar{B} = B + (\mathbf{a}_q - B\mathbf{e}_p)\mathbf{e}_p^T$
- Take B out as a factor on the left

$$\bar{B} = B[I + (B^{-1}\mathbf{a}_q - \mathbf{e}_p)\mathbf{e}_p^T] = BE$$

$$\text{where } E = I + (\hat{\mathbf{a}}_q - \mathbf{e}_p)\mathbf{e}_p^T = \begin{bmatrix} 1 & & & \eta_1 & & \\ & \ddots & & \vdots & & \\ & & \mu & & & \\ & & \vdots & \ddots & & \\ & & \eta_m & & 1 & \end{bmatrix}$$

$\mu = \hat{a}_{pq}$ is the **pivot**; remaining entries in $\hat{\mathbf{a}}_q$ form the **eta vector** $\boldsymbol{\eta}$

- Can solve $\bar{B}\mathbf{x} = \mathbf{r}$ as $B\mathbf{x} = \mathbf{r}$ then $\mathbf{x} := E^{-1}\mathbf{x}$ as

$$x_p := x_p / \mu \quad \text{then} \quad \mathbf{x} := \mathbf{x} - x_p \boldsymbol{\eta}$$

Dantzig and Orchard-Hays (1954)

NLA Challenge: Hyper-sparsity

NLA Challenge: Solve $B\mathbf{x} = \mathbf{r}$ for sparse $\mathbf{r}$

- Given $B = LU$, solve $B\mathbf{x} = \mathbf{r}$ as

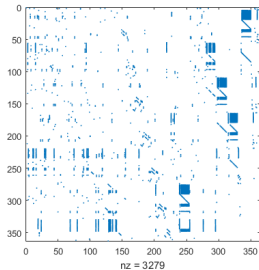
$$L\mathbf{y} = \mathbf{r}; \quad U\mathbf{x} = \mathbf{y}$$

- In revised simplex method, $\mathbf{r}$ is sparse: consequences?
 - If B is irreducible then $\mathbf{x}$ is full
 - If B is highly reducible then $\mathbf{x}$ can be **sparse**
- Phenomenon of **hyper-sparsity**
 - Exploit it when *forming* $\mathbf{x}$
 - Exploit it when *using* $\mathbf{x}$

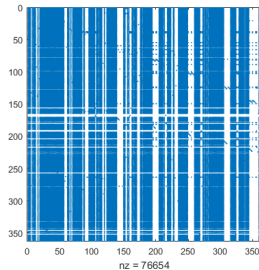
NLA Challenge: Hyper-sparsity

Inverse of a sparse matrix and solution of $Bx = r$

Optimal B for LP problem STAIR



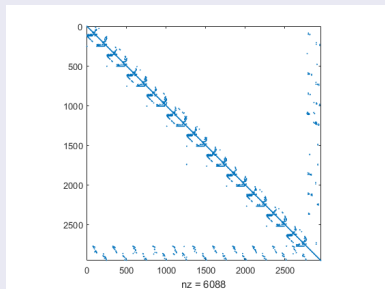
B^{-1} has density of 58%, so $B^{-1}r$ is typically dense



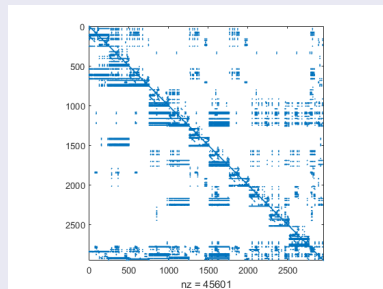
NLA Challenge: Hyper-sparsity

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

Optimal B for LP problem pds-02



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



NLA Challenge: Hyper-sparsity

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

NLA Challenge: Hyper-sparsity

Recall: Solve $Lx = b$ using

```
function ftranL( $L, b, x$ )  
   $r = b$   
  for all  $j \in \{1, \dots, m\}$  do  
    for all  $i : L_{ij} \neq 0$  do  
       $r_i = r_i - L_{ij}r_j$   
 $x = r$ 
```

When b is **sparse**

- Inefficient until r fills in

NLA Challenge: Hyper-sparsity

Better: Check r_j for zero

```
function ftranL( $L, \mathbf{b}, \mathbf{x}$ )  
   $\mathbf{r} = \mathbf{b}$   
  for all  $j \in \{1, \dots, m\}$  do  
    if  $r_j \neq 0$  then  
      for all  $i : L_{ij} \neq 0$  do  
         $r_i = r_i - L_{ij}r_j$   
  
   $\mathbf{x} = \mathbf{r}$ 
```

When $\mathbf{x}$ 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)

NLA Challenge: 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)

Impact: Feed formulation

Dantzig-Wolfe structure

Suppose N people are choosing diets with limits on the available food

- Person k has an individual LP $\min \mathbf{c}_k^T \mathbf{x}_k$ s.t. $A_k \mathbf{x}_k = \mathbf{b}_k$ and $\mathbf{x}_k \geq \mathbf{0}$
- Person k consumes $x_{kj} = [\mathbf{x}_k]_j$ of food j so $\sum_{k=1}^N x_{kj} \leq b_j$
- This **availability constraint** links the individual LPs

Dantzig-Wolfe structure and decomposition method (1960)

$$\begin{array}{llllll} \min & f & = & \mathbf{c}_1^T \mathbf{x}_1 & + & \dots & + & \mathbf{c}_N^T \mathbf{x}_N \\ \text{s.t.} & & & A_{01} \mathbf{x}_1 & + & \dots & + & A_{0N} \mathbf{x}_N & \leq & \mathbf{b}_0 \\ & & & A_1 \mathbf{x}_1 & & & & & = & \mathbf{b}_1 \\ & & & & & \ddots & & & & \vdots \\ & & & & & & & A_N \mathbf{x}_N & = & \mathbf{b}_N \\ & & & \mathbf{x}_1 \geq \mathbf{0} & \dots & & & \mathbf{x}_N \geq \mathbf{0} & & \end{array}$$

Impact: Feed formulation

Animal feed formulation

- Animal feed is blended from many ingredients
 - Multiple animals; multiple diets
 - Shared raw materials
 - Formulate at minimum cost!
- Dantzig-Wolfe structure



Cargill (Format Solutions)

- 20-year relationship
- Format software blends half the world's manufactured animal food
 - Farm food: $O(\$1\text{bn})$
 - Pet food: $O(\$10\text{bn})$
 - Trade commodities
- Good for Impact!

NLA Challenge: Parallel solution of structured LP problems

NLA Challenge: Parallel solution of stochastic MIP problems

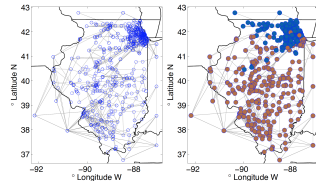
Two-stage stochastic LPs have column-linked block angular (BALP) structure

$$\begin{array}{llllllllllll} \text{minimize} & \mathbf{c}_0^T \mathbf{x}_0 & + & \mathbf{c}_1^T \mathbf{x}_1 & + & \mathbf{c}_2^T \mathbf{x}_2 & + & \dots & + & \mathbf{c}_N^T \mathbf{x}_N & & \\ \text{subject to} & \mathbf{A} \mathbf{x}_0 & & & & & & & & & = & \mathbf{b}_0 \\ & \mathbf{T}_1 \mathbf{x}_0 & + & \mathbf{W}_1 \mathbf{x}_1 & & & & & & & = & \mathbf{b}_1 \\ & \mathbf{T}_2 \mathbf{x}_0 & & & + & \mathbf{W}_2 \mathbf{x}_2 & & & & & = & \mathbf{b}_2 \\ & \vdots & & & & & & \ddots & & & \vdots \\ & \mathbf{T}_N \mathbf{x}_0 & & & & & & & + & \mathbf{W}_N \mathbf{x}_N & = & \mathbf{b}_N \\ & \mathbf{x}_0 \geq \mathbf{0} & & \mathbf{x}_1 \geq \mathbf{0} & & \mathbf{x}_2 \geq \mathbf{0} & & \dots & & \mathbf{x}_N \geq \mathbf{0} & & \end{array}$$

- 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

NLA Challenge: Parallel solution of 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
 - Solve nodes using parallel dual simplex solver PIPS-S
- Lubin, Petra *et al.* (2011)



PIPS-S: Exploiting problem structure

Convenient to permute the LP thus:

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

PIPS-S: Exploiting problem structure

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

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

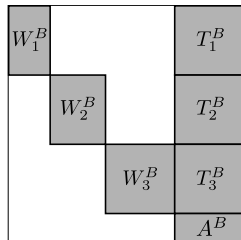
- W_i^B are columns corresponding to n_i^B basic variables in scenario i

- $\begin{bmatrix} T_1^B \\ \vdots \\ T_N^B \\ A^B \end{bmatrix}$ 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

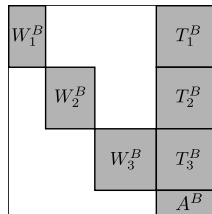
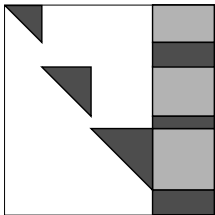
$$B = \begin{bmatrix} W_1^B & & & T_1^B \\ & \ddots & & \vdots \\ & & W_N^B & T_N^B \\ & & & A^B \end{bmatrix}$$



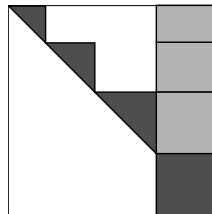
- B is nonsingular so
 - W_i^B are “tall”: full column rank
 - $[W_i^B \ T_i^B]$ are “wide”: full row rank
 - A^B is “wide”: full row rank
- Scope for parallel inversion is immediate and well known

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

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)
COAP best paper prize (2013)

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

NLA Challenge: Parallel solution of general LP problems

NLA Challenge: Parallel solution of general LP problems

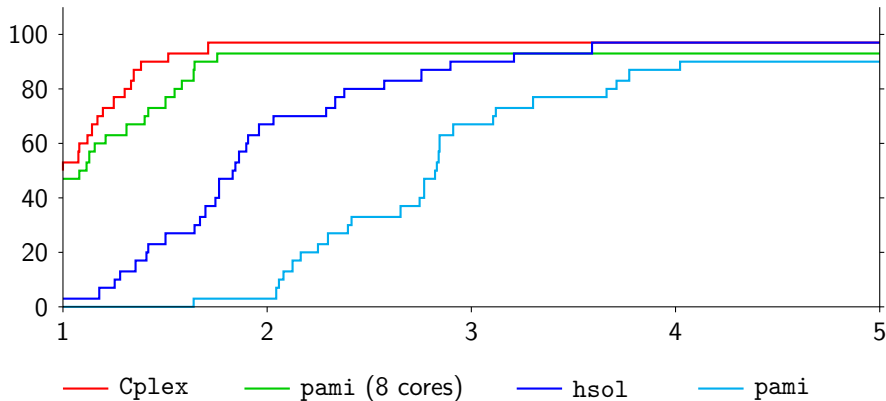
- Perform standard dual simplex minor iterations for rows in set $\mathcal{P}$ ($|\mathcal{P}| \ll m$)
- Suggested by Rosander (1975) but never implemented efficiently *in serial*

	$\mathcal{N}$	RHS
$\mathcal{B}$	$\hat{\mathbf{a}}_{\mathcal{P}}^T$	$\hat{\mathbf{b}}$
		$\hat{\mathbf{b}}_{\mathcal{P}}$
	$\hat{\mathbf{c}}_N^T$	

- Multiple independent linear systems and matrix-vector products
⇒ Scope for parallelism (pami)

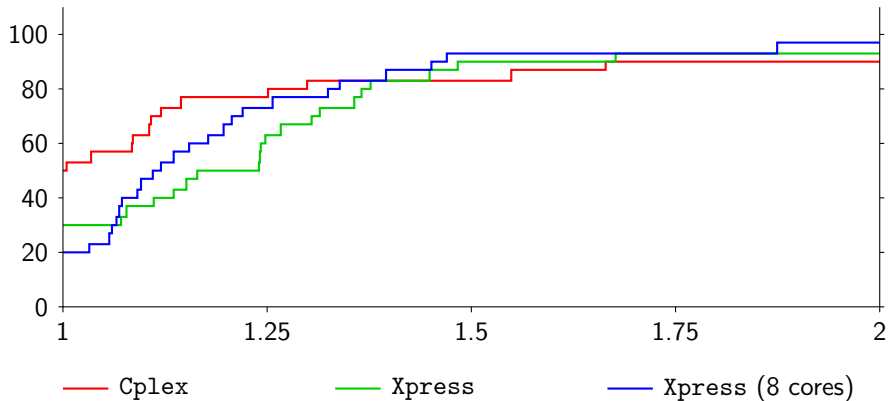
Huangfu and H (2011–2014)
COAP best paper prize (2015)
MPC best paper prize (2018)

HiGHS: Cplex vs pami vs hsol



- pami is less efficient than hsol in serial
- pami speedup more than compensates
- pami performance approaching Cplex

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: A high-performance linear optimizer

HiGHS: **H**all, **i**vet **G**alabova, **H**uangfu, **S**chork, (Feldmeier and Fogg)



Features

- Model management
- Interfaces
- Presolve
- Interior point method

Availability

- Open-source under MIT license
- No third-party code
- See <http://www.higs.dev/>

Aim: to become the world's leading open-source linear optimization software



- Solving large-scale LPs offers NLA challenges
- Novel techniques of practical value are hard to find but get noticed
- Having a high-performance solver is good for consultancy, Impact and kudos

Slides: <http://www.maths.ed.ac.uk/hall/Stirling19>

HiGHS: <http://www.hihs.dev/>



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.

Novel update techniques for the revised simplex method.

Computational Optimization and Applications, 60(4):587–608, 2015.



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. Animescu.

Parallel distributed-memory simplex for large-scale stochastic LP problems.

Computational Optimization and Applications, 55(3):571–596, 2013.