Three high performance simplex solvers

Julian Hall[1] Qi Huangfu[2] Ivet Galabova[1] Others

[1]School of Mathematics, University of Edinburgh

[2]FICO

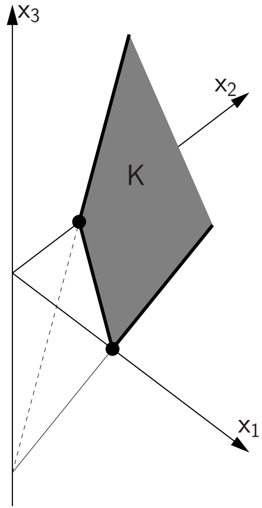
Argonne National Laboratory

16 May 2017

Overview

- Background
  - Dual vs primal simplex method
- Three high performance simplex solvers
  - `EMSOL` (1992–2004)
  - `PIPS-S` (2011–date)
  - `hsol` (2011–date)
- The future

Solving LP problems: Characterizing a basis



$$\begin{array}{ll} \text{minimize} & f = \boldsymbol{c}^T\boldsymbol{x} \\ \text{subject to} & A\boldsymbol{x} = \boldsymbol{b} \quad \boldsymbol{x} \geq \boldsymbol{0} \end{array}$$

- A **vertex** of the **feasible region** $K \subset \mathbb{R}^n$ 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 $\{1, \ldots, n\}$
- Equations partitioned according to $\mathcal{B} \cup \mathcal{N}$ as

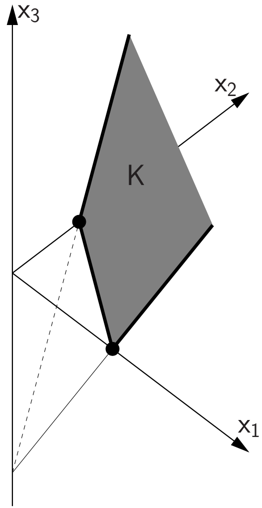
  $$B\boldsymbol{x}_B + N\boldsymbol{x}_N = \boldsymbol{b}$$

  with nonsingular **basis matrix** B
- Solution set of $A\boldsymbol{x} = \boldsymbol{b}$ characterized by

  $$\boldsymbol{x}_B = \widehat{\boldsymbol{b}} - B^{-1}N\boldsymbol{x}_N$$

  where $\widehat{\boldsymbol{b}} = B^{-1}\boldsymbol{b}$

Solving LP problems: Optimality conditions



$$\begin{array}{ll}\text{minimize} & f = \boldsymbol{c}^T\boldsymbol{x} \\ \text{subject to} & A\boldsymbol{x} = \boldsymbol{b} \quad \boldsymbol{x} \geq \boldsymbol{0}\end{array}$$

- Objective partitioned according to $\mathcal{B} \cup \mathcal{N}$ as

$$\begin{aligned} f &= \boldsymbol{c}_B^T\boldsymbol{x}_B + \boldsymbol{c}_N^T\boldsymbol{x}_N \\ &= \widehat{f} + \widehat{\boldsymbol{c}}_N^T\boldsymbol{x}_N \end{aligned}$$

  where $\widehat{f} = \boldsymbol{c}_B^T\widehat{\boldsymbol{b}}$ and $\widehat{\boldsymbol{c}}_N^T = \boldsymbol{c}_N^T - \boldsymbol{c}_B^T B^{-1}N$ is the vector of **reduced costs**

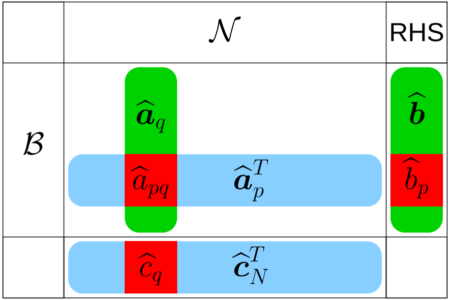
- Partition yields an optimal solution if there is
  - Primal feasibility $\widehat{\boldsymbol{b}} \geq \boldsymbol{0}$
  - Dual feasibility $\widehat{\boldsymbol{c}}_N \geq \boldsymbol{0}$

Primal simplex algorithm

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

Scan $\widehat{c}_j$ 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 $\widehat{\boldsymbol{a}}_q = B^{-1}\boldsymbol{a}_q$

Why does it work?

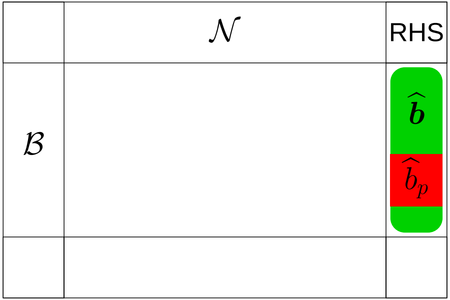
Objective improves by $-\dfrac{\widehat{b}_p \times \widehat{c}_q}{\widehat{a}_{pq}}$ each iteration

Dual simplex algorithm

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

Scan $\widehat{b}_i < 0$ for p to leave $\mathcal{B}$

Scan $\widehat{c}_j / \widehat{a}_{pj}$ for q to leave $\mathcal{N}$

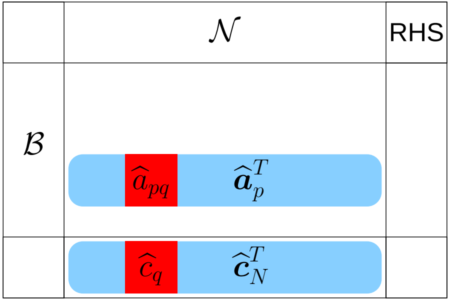


Dual simplex algorithm

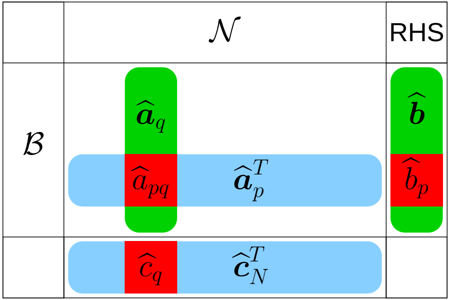
Assume $\widehat{\boldsymbol{c}}_N \geq \mathbf{0}$ Seek $\widehat{\boldsymbol{b}} \geq \mathbf{0}$

Scan $\widehat{b}_i < 0$ for p to leave $\mathcal{B}$

Scan $\widehat{c}_j / \widehat{a}_{pj}$ for q to leave $\mathcal{N}$



Dual simplex algorithm



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

Scan $\widehat{b}_i < 0$ for p to leave $\mathcal{B}$

Scan $\widehat{c}_j / \widehat{a}_{pj}$ for q to leave $\mathcal{N}$

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 $\widehat{\boldsymbol{a}}_q = B^{-1} \boldsymbol{a}_q$

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

- 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` Invert B

Simplex algorithm: Primal or dual?

Primal simplex algorithm

- Traditional variant
- Solution is generally not primal feasible when LP is tightened

Dual simplex algorithm

- Preferred variant
- Easier to get dual feasibility
- More progress in many iterations
- Solution is dual feasible when LP is tightened

EMSOL (1992–2004)

Overview

- Written in FORTRAN to study parallel simplex
- Primal simplex
- Product form update for $B := B + (\boldsymbol{a}_q - B\boldsymbol{e}_p)\boldsymbol{e}_p^T$
  - simple and easy to parallelize
  - inefficient and numerically unstable

Output: Parallel codes

- ASYNPLEX and PARSMI developed for Cray T3D
- Slave processors send attractive tableau columns to a master processor which keeps them up-to-date and determines basis changes
- Modest speed-up on general sparse LP problems
- Non-deterministic

H and McKinnon (1995–1998)

EMSOL: Theoretical and serial outputs

Output: Simplest LPs which cycle

- Each simplex iteration improves the objective by $-\frac{\widehat{b}_p \times \widehat{c}_q}{\widehat{a}_{pq}}$
  - Termination guaranteed if all $\widehat{b}_p > 0$
  - **Cycling** can occur if some $\widehat{b}_p = 0$ (degeneracy)
- Found the family of (provably) simplest LPs which cycle, *even* with the leading practical anti-degeneracy technique H and McKinnon (1996–2004)

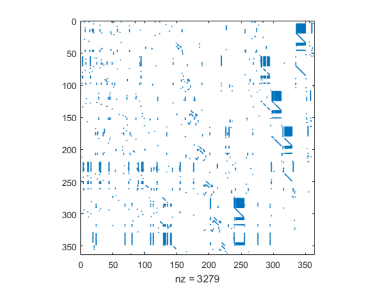
Output: Hyper-sparsity

- Solution of $B\boldsymbol{x} = \boldsymbol{r}$ is $\boldsymbol{x} = B^{-1}\boldsymbol{r}$
- In the simplex method, B and $\boldsymbol{r}$ are sparse
- For some LPs $\boldsymbol{x}$ is typically sparse: **hyper-sparsity**

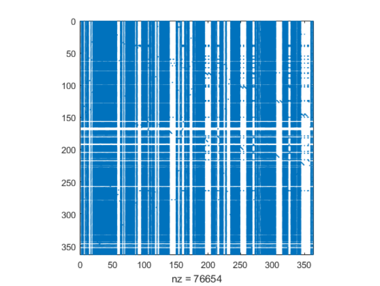
Remarkable?

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

Optimal B for LP problem `stair`

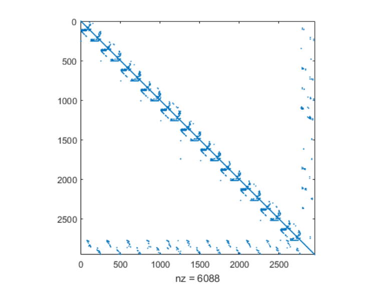


B^{-1} has density of 58%, so $B^{-1}\boldsymbol{r}$ is typically dense

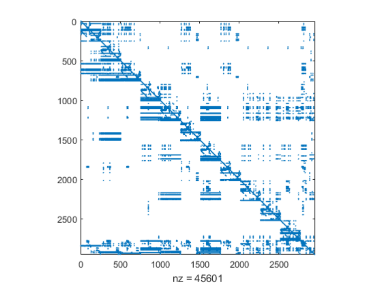


Inverse of a sparse matrix and solution of $Bx = r$

Optimal B for LP problem pds-02



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



EMSOL: Exploiting hyper-sparsity

To solve $B\boldsymbol{x} = \boldsymbol{r}$

- Using decomposition of B given by $\{p_k, \mu_k, \boldsymbol{\eta}_k\}_{k=1}^{K}$
- Traditional technique transforms RHS into solution $\boldsymbol{x}$

do $k = 1,\ K$

$$r_{p_k} := r_{p_k}/\mu_k$$
$$\boldsymbol{r} := \boldsymbol{r} - r_{p_k}\boldsymbol{\eta}_k$$

end do

- When initial RHS is sparse: inefficient until $\boldsymbol{r}$ fills in

EMSOL: Exploiting hyper-sparsity

To solve $B\boldsymbol{x} = \boldsymbol{r}$

- Using decomposition of B given by $\{p_k, \mu_k, \boldsymbol{\eta}_k\}_{k=1}^{K}$
- When $\boldsymbol{r}$ is sparse skip $\boldsymbol{\eta}_k$ if r_{p_k} is zero

```
do k = 1, K
   if (r_{p_k} .ne. 0) then
      r_{p_k} := r_{p_k}/μ_k
      r := r − r_{p_k} η_k
   end if
end do
```

- When $\boldsymbol{x}$ is sparse, the dominant cost is the test for zero
- Requires efficient identification of vectors $\boldsymbol{\eta}_k$ to be applied

Gilbert and Peierls (1988)
H and McKinnon (1998–2005)
COAP best paper prize (2005)

PIPS-S (2011–date)

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

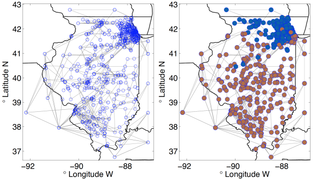
Two-stage stochastic LPs have column-linked block angular (BALP) structure

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

- 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: Stochastic MIP problems

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
- For column-linked BALP problems

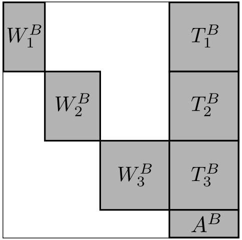
$$B = \begin{bmatrix} W_1^B & & & T_1^B \\ & \ddots & & \vdots \\ & & W_N^B & T_N^B \\ & & & A^B \end{bmatrix}$$

- 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

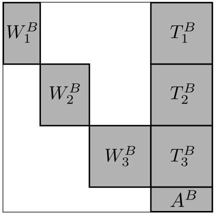
- Inversion of the basis matrix B is key to revised simplex efficiency
- For column-linked BALP problems

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

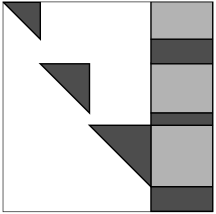


- B is nonsingular so
  - W_i^B are "tall": full column rank
  - $\begin{bmatrix} W_i^B & T_i^B \end{bmatrix}$ are "wide": full row rank
  - A^B is "wide": full row rank
- Scope for parallel inversion is immediate and well known

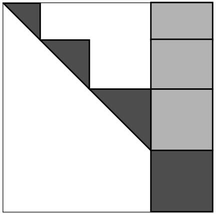
PIPS-S: Exploiting problem structure



- 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

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

`hsol` (2011–date)

Overview

- Written in C++ to study parallel simplex
- Dual simplex with steepest edge and BFRT
- Forrest-Tomlin update
  - complex and inherently serial
  - efficient and numerically stable

Concept

- 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`)
- Test-bed for research
- Work-horse for consultancy

Huangfu, H and Galabova (2011–date)

`hsol`: A high performance simplex solver

Features

- Model management: Add/delete/modify problem data
- Efficiency: High performance serial and parallel computational components
- Open-source C++

Presolve

- Presolve (and corresponding postsolve) has been implemented efficiently

  Remove redundancies in the LP to reduce problem dimension

  Galabova (2016)

Crash

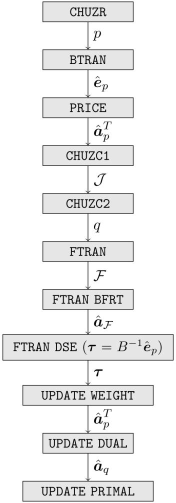
- Dual simplex "triangular basis" crash

  Can be valuable, even with dual steepest edge weights

- Alternative crash techniques being studied

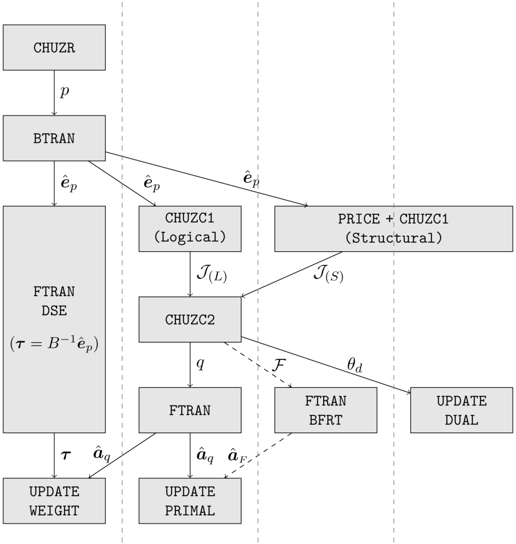
  H and Galabova (2016–date)

hsol: Single iteration parallelism with sip option



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

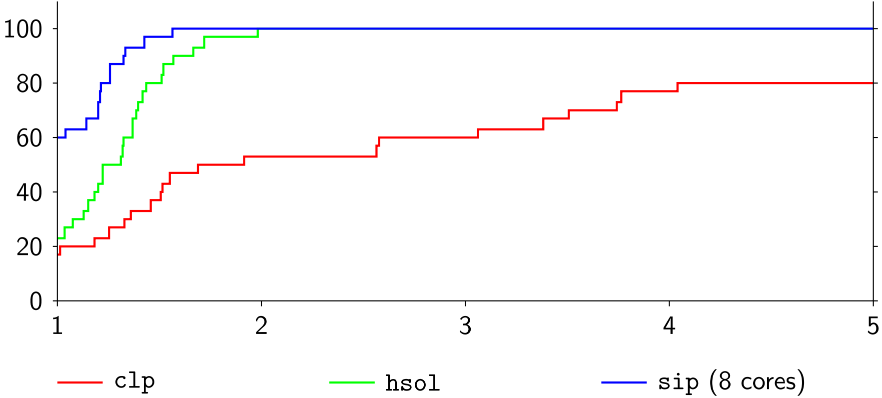
hsol: Single iteration parallelism with sip option



- Parallel PRICE to form $\hat{\mathbf{a}}_p^T = \boldsymbol{\pi}_p^T N$
- 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)

hsol: clp vs hsol vs sip

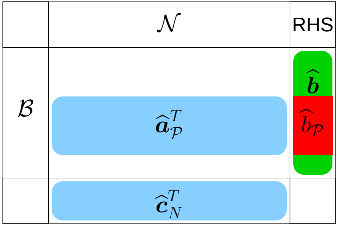


Performance on spectrum of 30 significant LP test problems

- hsol is 2.3 times faster than clp
- sip on 8 cores is 1.2 times faster than hsol; 2.6 times faster than clp

`hsol`: 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 $\boldsymbol{\pi}_{\mathcal{P}} = B^{-1}\boldsymbol{e}_{\mathcal{P}}$
- 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)

`hsol`: Performance and reliability

Extended testing using 159 test problems

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

Exclude 7 which are "hard"

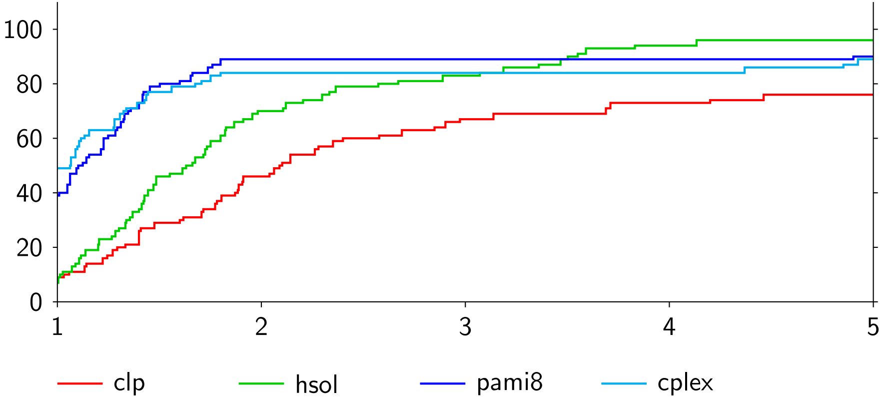
Performance

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

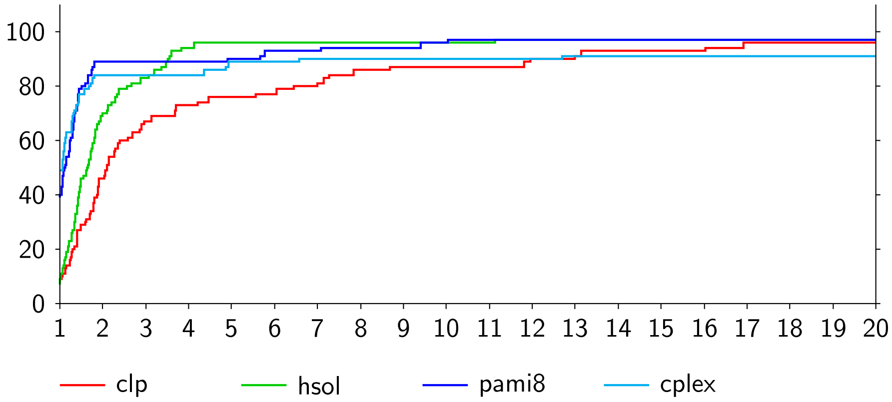
- Dual simplex
- No presolve
- No crash

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

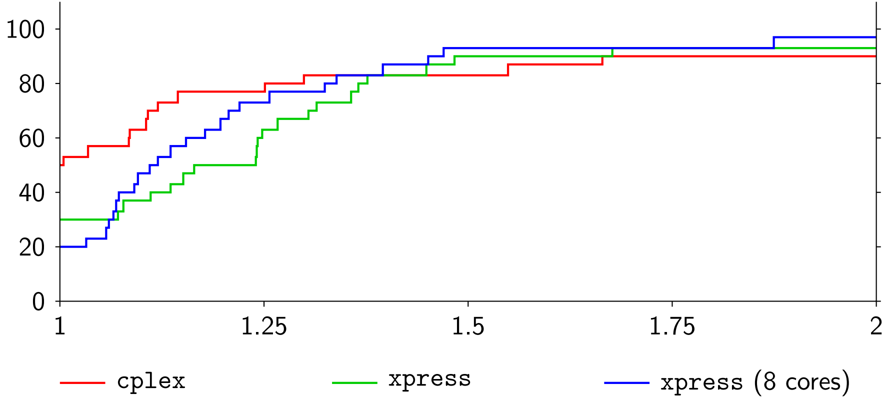
hsol: Performance



hsol: Reliability



hsol: Impact



- pami ideas incorporated in FICO Xpress (Huangfu 2014)
- Xpress simplex solver now fastest commercial simplex solver

`hsol`: Future

For 2017

- SCIP interface
- QP solver
- MIP solver

For 2018

- MIP presolve
- MIQP solver

Long term

Replacement for `clp`?

Commercial involvement

- Cargill (feed formulation)
- Google (techniques in `glop`)
- Financial services

Conclusions

- `EMSOL`: Parallel variants; Advance in serial simplex
- `PIPS-S`: Large scale data-parallelism for special problems
- `hsol`: Task and data parallelism; test-bed for further research

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.
Technical Report ERGO-14-011, School of Mathematics, University of Edinburgh, 2014.
Accepted for publication in Mathematical Programming Computation.

Q. Huangfu and J. A. J. Hall.
Novel update techniques for the revised simplex method.
Computational Optimization and Applications, 60(4):587–608, 2015.

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.

Slides: http://www.maths.ed.ac.uk/hall/Argonne17