High performance numerical linear algebra for the revised simplex method

Julian Hall

School of Mathematics
University of Edinburgh

jajhall@ed.ac.uk

Workshop on Linear Algebra for PDEs and Optimization, Edinburgh

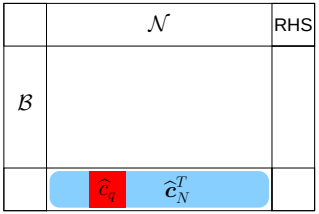
4 September 2017

THE UNIVERSITY *of* EDINBURGH

Overview

- Primal revised simplex method
  - Solving linear systems (FTRAN and BTRAN) with focus on exploiting hyper-sparsity
  - Exploiting $\bar{B} = B + (\boldsymbol{a}_q - B\boldsymbol{e}_p)\boldsymbol{e}_p^T$ (UPDATE-BASIS)
    - Traditional techniques
    - Novel techniques
- Exploiting parallelism
  - Background
  - Data parallelism by exploiting LP structure
  - Task parallelism for general LP problems
- Conclusions

The simplex algorithm: Choosing a column

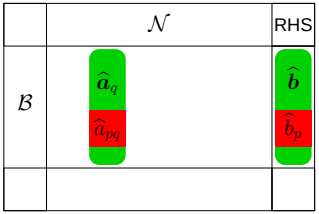


Primal algorithm: Assume $\widehat{\boldsymbol{b}} \geq \boldsymbol{0}$ Seek $\widehat{\boldsymbol{c}}_N \geq \boldsymbol{0}$

Scan $\widehat{c}_j < 0$, $i \in \mathcal{N}$, for a good candidate q to leave $\mathcal{N}$ CHUZC

Scan $\widehat{b}_i / \widehat{a}_{iq} > 0$, $i \in \mathcal{B}$, for a good candidate p to leave $\mathcal{B}$ CHUZR

The simplex algorithm: Choosing a row

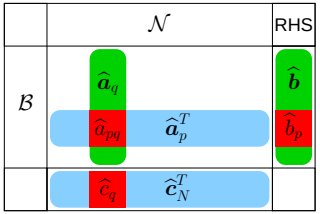


Primal algorithm: Assume $\widehat{\boldsymbol{b}} \geq \mathbf{0}$ Seek $\widehat{\boldsymbol{c}}_N \geq \mathbf{0}$

Scan $\widehat{c}_j < 0$, $i \in \mathcal{N}$, for a good candidate q to leave $\mathcal{N}$ CHUZC

Scan $\widehat{b}_i/\widehat{a}_{iq} > 0$, $i \in \mathcal{B}$, for a good candidate p to leave $\mathcal{B}$ CHUZR

The simplex algorithm: Update



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-PRIMAL

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

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 $\left| \dfrac{\widehat{b}_p \times \widehat{c}_q}{\widehat{a}_{pq}} \right|$ 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: $\widehat{N} := \widehat{N} - \frac{1}{\widehat{a}_{pq}} \widehat{\boldsymbol{a}}_q \widehat{\boldsymbol{a}}_p^T$

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

Simplex method: Hyper-sparsity

Recall: major computational components

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

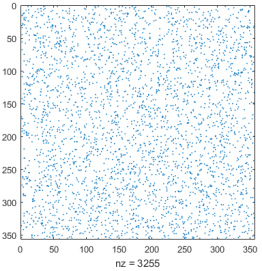
Phenomenon of hyper-sparsity

- Vectors $\boldsymbol{e}_p$ and $\boldsymbol{a}_q$ are sparse
- Results $\boldsymbol{\pi}_p$, $\widehat{\boldsymbol{a}}_p^T$ and $\widehat{\boldsymbol{a}}_q$ may be sparse—because B^{-1} is sparse
  - In BTRAN, $\boldsymbol{\pi}_p$ is a row of B^{-1}
  - In PRICE, $\widehat{\boldsymbol{a}}_p^T$ is a linear combination of a few rows of N
  - In FTRAN, $\widehat{\boldsymbol{a}}_q$ is a linear combination of a few columns of B^{-1}

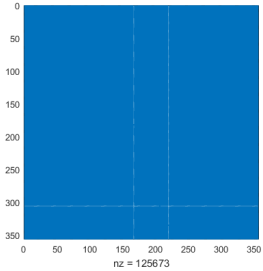
Simplex method: Hyper-sparsity

Simplex method: Inverse of a sparse matrix and solution of $B\boldsymbol{x} = \boldsymbol{b}$

Random sparse matrix B
$m = 356$ and density 2.5%



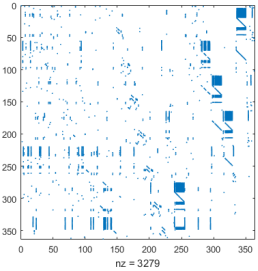
B^{-1} has density of 99%



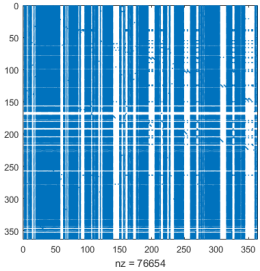
Simplex method: Hyper-sparsity

Simplex method: Inverse of a sparse matrix and solution of $B\boldsymbol{x} = \boldsymbol{b}$

Optimal B for LP problem `stair`
$m = 356$ and density 2.5%



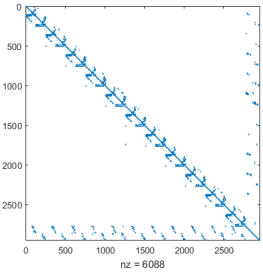
B^{-1} has density of 58%
$B^{-1}\boldsymbol{b}$ for sparse $\boldsymbol{b}$ is typically dense



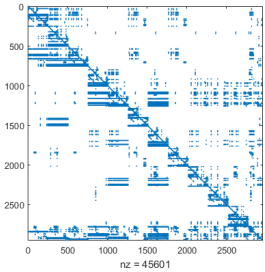
Simplex method: Hyper-sparsity

Simplex method: Inverse of a sparse matrix and solution of $B\boldsymbol{x} = \boldsymbol{b}$

Optimal B for LP problem pds-02
$m = 2953$ and density 0.07%



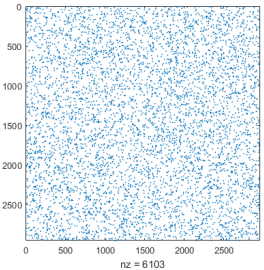
B^{-1} has density of 0.52%
$B^{-1}\boldsymbol{b}$ for sparse $\boldsymbol{b}$ is typically sparse



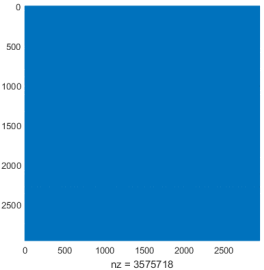
Simplex method: Hyper-sparsity

Simplex method: Inverse of a sparse matrix and solution of $B\boldsymbol{x} = \boldsymbol{b}$

Random matrix
$m = 2953$ and density 0.07%



B^{-1} has density of 100%



Simplex method: Hyper-sparsity

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

Simplex method: Hyper-sparsity

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

function `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

Simplex method: 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)
COAP best paper prize (2005)

Simplex method: UPDATE-BASIS

- **Each iteration:**
  - Solve $B^T \boldsymbol{\pi}_p = \boldsymbol{e}_p$
  - Solve $B \widehat{\boldsymbol{a}}_q = \boldsymbol{a}_q$
  - Column p of B replaced by $\boldsymbol{a}_q$ to give $\bar{B} = B + (\boldsymbol{a}_q - B\boldsymbol{e}_p)\boldsymbol{e}_p^T$
- **Sparsity-exploiting decomposition:** $PBQ = LU$
- **Challenge:** Solve systems involving $\bar{B}$ at minimal cost

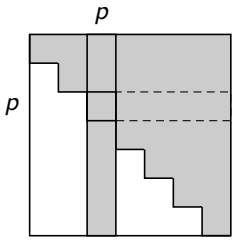
Simplex method: Product form update (PFI) for B

- Given
$$\bar{B} = B + (\boldsymbol{a}_q - B\boldsymbol{e}_p)\boldsymbol{e}_p^T$$
- Take B out as a factor on the left
$$\bar{B} \;=\; B[I + (B^{-1}\boldsymbol{a}_q - \boldsymbol{e}_p)\boldsymbol{e}_p^T] \;=\; BE$$
$$\text{where} \quad E \;=\; I + (\widehat{\boldsymbol{a}}_q - \boldsymbol{e}_p)\boldsymbol{e}_p^T \;=\; \begin{bmatrix} 1 & & \eta_1 & & \\ & \ddots & \vdots & & \\ & & \mu & & \\ & & \vdots & \ddots & \\ & & \eta_m & & 1 \end{bmatrix}$$
$\mu = \widehat{a}_{pq}$ is the **pivot**; remaining entries in $\widehat{\boldsymbol{a}}_q$ form the **eta vector** $\boldsymbol{\eta}$
- Can solve $\bar{B}\boldsymbol{x} = \boldsymbol{r}$ as $B\boldsymbol{x} = \boldsymbol{r}$ then $\boldsymbol{x} := E^{-1}\boldsymbol{x}$ as
$$x_p := x_p/\mu \qquad \text{then} \qquad \boldsymbol{x} := \boldsymbol{x} - x_p\boldsymbol{\eta}$$

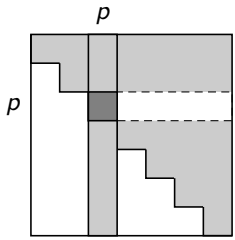
Dantzig and Orchard-Hays (1954)

Simplex method: Forrest-Tomlin update (FT) for B

- Given
$$\bar{B} = B + (\boldsymbol{a}_q - B\boldsymbol{e}_p)\boldsymbol{e}_p^T \quad \text{where (wlog)} \quad B = LU$$
- Multiply $\bar{B}$ by L^{-1} to give
$$L^{-1}\bar{B} \;=\; U + (L^{-1}\boldsymbol{a}_q - U\boldsymbol{e}_p)\boldsymbol{e}_p^T \;=\; U + (\widetilde{\boldsymbol{a}}_q - \boldsymbol{u}_p)\boldsymbol{e}_p^T \;=\; U' \qquad \text{(a)}$$
- Eliminate entries in row p to give $R^{-1}U' = \bar{U}$ (b)



(a) Spiked upper: U'



(b) After elimination: $\bar{U}$

- Yields $\bar{B} = LR\bar{U}$
- Compute $\widetilde{\boldsymbol{a}}_q$ when forming $\widehat{\boldsymbol{a}}_q$
- Represent R like E
- FT more efficient than PFI with respect to sparsity

Forrest and Tomlin (1972)

Simplex method: Multiple updates

- Suppose $B_0 = L_0 U_0$ and k updates are performed to obtain B_k
  - Pivot in rows $\{p_i\}_{i=1}^{k}$
  - Introduce columns $\{\boldsymbol{a}_{q_i}\}_{i=1}^{k}$
- PFI generalizes as

$$B_k = B_0 E_1 E_2 \dots E_k$$

- FT generalizes as

$$B_k = L_0 R_1 R_2 \dots R_k U_k$$

Eventually more computationally efficient or numerically prudent to reinvert some B_k

Novel update techniques for the revised simplex method: 1

Alternative product form update (APF)

- **Recall:** Column p of B is replaced by $\boldsymbol{a}_q$ to give $\bar{B} = B + (\boldsymbol{a}_q - B\boldsymbol{e}_p)\boldsymbol{e}_p^T$
  - Traditional PFI takes B out as a factor on the left so $\bar{B} = BE$
- **Idea:** Why not take it out on the right!

$$\begin{aligned} \bar{B} &= [I + (\boldsymbol{a}_q - B\boldsymbol{e}_p)\boldsymbol{e}_p^T B^{-1}]B &= TB \\ \text{where} \quad T &= I + (\boldsymbol{a}_q - \boldsymbol{a}_{p'})\widehat{\boldsymbol{e}}_p^T \end{aligned}$$

- T is formed of known data and readily invertible (like E for PFI)
  Naturally compute $\widehat{\boldsymbol{e}}_p$ when solving $B^T\boldsymbol{\pi}_p = \boldsymbol{e}_p$
- **But:** Is this useful?

Novel update techniques for the revised simplex method: 2

Middle product form update (MPF)

- **Recall:** Column p of B is replaced by $\boldsymbol{a}_q$ to give $\bar{B} = B + (\boldsymbol{a}_q - B\boldsymbol{e}_p)\boldsymbol{e}_p^T$
- **Idea:** Substitute $B = LU$ and take factors L on the left and U on the right!

$$\begin{aligned}
\bar{B} &= LU + (\boldsymbol{a}_q - B\boldsymbol{e}_p)\boldsymbol{e}_p^T \\
&= LU + LL^{-1}(\boldsymbol{a}_q - B\boldsymbol{e}_p)\boldsymbol{e}_p^T U^{-1}U \\
&= L[I + (\widetilde{\boldsymbol{a}}_q - U\boldsymbol{e}_p)\widetilde{\boldsymbol{e}}_p^T]U \\
&= LMU \qquad \text{where} \quad M = I + (\widetilde{\boldsymbol{a}}_q - \boldsymbol{u}_p)\widetilde{\boldsymbol{e}}_p^T
\end{aligned}$$

- M is formed of known data and readily invertible (like E for PFI)
  Naturally compute $\widetilde{\boldsymbol{a}}_q$ when solving $B\,\widehat{\boldsymbol{a}}_q = \boldsymbol{a}_q$ and $\widetilde{\boldsymbol{e}}_p$ when solving $B^T\boldsymbol{\pi}_p = \boldsymbol{e}_p$
- **But:** Is this useful?

Novel update techniques for the revised simplex method: 3

Forrest-Tomlin update

- **Recall:** Column p of B is replaced by $\boldsymbol{a}_q$ to give $\bar{B} = B + (\boldsymbol{a}_q - B\boldsymbol{e}_p)\boldsymbol{e}_p^T$
- **Idea:** Substitute $B = LU$ and take factor L on the left

$$\bar{B} \quad = \quad L[U + (L^{-1}\boldsymbol{a}_q - U\boldsymbol{e}_p)\boldsymbol{e}_p^T] \quad = \quad LU'$$

where $U' = U + (\widetilde{\boldsymbol{a}}_q - \boldsymbol{u}_p)\boldsymbol{e}_p^T$ is "spiked" upper triangular and then eliminate

Collective Forrest-Tomlin (CFT) update

- Update Forrest-Tomlin representation of B after multiple basis changes
- Don't have data to perform a sequence of standard FT updates
- Have to perform elimination corresponding to multiple spikes
- **But:** Is this useful?

Huangfu and H (2013)

Novel update techniques for the revised simplex method: Multiple updates

- Suppose $B_0 = L_0 U_0$ and k updates are performed to obtain B_k
- **Recall:** PFI generalizes as $B_k = B_0 E_1 E_2 \dots E_k$
- **Recall:** FT generalizes as $B_k = L_0 R_1 R_2 \dots R_k U_k$
- APF generalizes as

$$B_k = T_k \dots T_2 T_1 B$$

- MPF generalizes as

$$B_k = L_0 M_1 M_2 \dots M_k U_0$$

Novel update techniques for the revised simplex method: Results

- Test environment
  - 30 representative LP problems from standard test sets
  - Same sequence of basis changes for all update techniques
- Geometric mean time for operations to form and use update data

Update	PFI	FT	APF	MPF	CFT
Mean	1.00	0.30	0.95	0.45	0.29

Conclusions:

- FT much better than PFI
- APF little better than PFI
- MPF closer to FT than PFI
- CFT as good as FT

Parallelising the revised simplex method

Parallelising the simplex method

History

- SSM: Good parallel efficiency of $\widehat{N} := \widehat{N} - \frac{1}{\widehat{a}_{pq}} \widehat{\boldsymbol{a}}_q \widehat{\boldsymbol{a}}_p^T$ *was* achieved
  Many! (1988–date)
- Only parallel *revised* simplex is worthwhile: goal of H and McKinnon in 1992!

Parallel primal revised simplex method

- Overlap computational components for different iterations
  Wunderling (1996), H and McKinnon (1995-2005)
- Modest speed-up *was* achieved on general sparse LP problems

Parallel dual revised simplex method

- Only immediate parallelism is in forming $\boldsymbol{\pi}_p^T N$
- When $n \gg m$ significant speed-up *was* achieved Bixby and Martin (2000)

Parallelising the simplex method: Matrix-vector product $\boldsymbol{\pi}_p^T N$ (PRICE)

- **In theory:**
  - Partition $N = \begin{bmatrix} N_1 & N_2 & \dots & N_P \end{bmatrix}$, where P is the number of processes
  - Compute $\boldsymbol{\pi}_p^T N_j$ on process j for $j = 1, 2, \dots P$
  - Execution time is reduced by a factor P: **linear speed-up**
- **In practice**
  - On a distributed memory machine with N_j on processor j
    Linear speed-up achieved
  - On a shared memory machine
    - If N_j fits into cache for process j
      Linear speed-up achieved
    - Otherwise, computation is limited by speed of reading data from memory
      Speed-up limited to number of memory channels

Data parallelism by exploiting LP structure

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

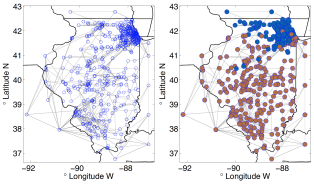
Two-stage stochastic LPs have column-linked block angular (BALP) structure

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

- Variables $\boldsymbol{x}_0 \in \mathbb{R}^{n_0}$ are **first stage** decisions
- Variables $\boldsymbol{x}_i \in \mathbb{R}^{n_i}$ 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:

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

PIPS-S: Exploiting problem structure

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

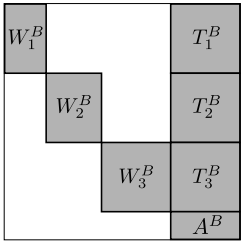
$$B = \begin{bmatrix} W_1^B & & & T_1^B \\ & \ddots & & \vdots \\ & & W_N^B & T_N^B \\ & & & A^B \end{bmatrix}$$

- 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

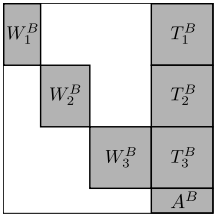
- Inversion of the basis matrix B is key to revised simplex efficiency
- For column-linked BALP problems

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

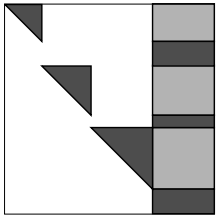


- B is nonsingular so
  - W_i^B are "tall": full column rank
  - $\begin{bmatrix} W_i^B & T_i^B \end{bmatrix}$ are "wide": full row rank
  - A^B is "wide": full row rank
- Scope for parallel inversion is immediate and well known

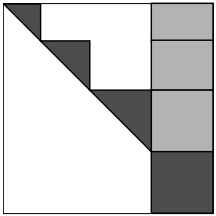
PIPS-S: Exploiting problem structure



- 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

Task parallelism for general LP problems

$h_2$gmp (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

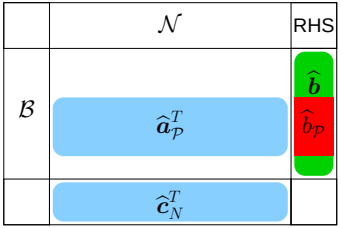
Concept

- 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)

h2gmp: 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)

Novel update procedures: are they useful?

Update 1: Alternative product form update (APF)

- **Recall:** $\bar{B} = TB$ with T easily invertible
- Used to get $\boldsymbol{\pi}_{\mathcal{P}} = \bar{B}^{-T}\boldsymbol{e}_{\mathcal{P}}$ from $B^{-T}\boldsymbol{e}_{\mathcal{P}}$ in `pami`
- Used to compute multiple $B^{-1}\boldsymbol{a}_F$ efficiently after multiple BFRT in `pami`

Update 2: Middle product form update (MPF)

- **Recall:** $\bar{B} = LMU$ with M easily invertible
- Not used by `pami`!
- Used by Google in `glop`

Update 3: Multiple Forrest-Tomlin update

Used to perform multiple Forrest-Tomlin updates after minor iterations in `pami`

h2gmp: Performance and reliability

Extended testing using 159 test problems

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

Exclude 7 which are "hard"

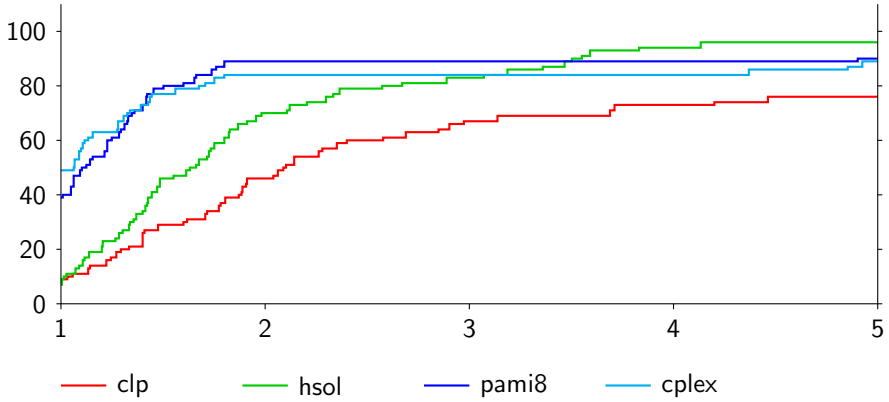
Performance

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

h2gmp: Performance



Conclusions

High performance NLA is essential for the revised simplex method

- Hyper-sparsity: advance in serial simplex
- Stochastic LP: large scale data-parallelism for special problems
- Novel updates: underpin open-source parallel simplex solver

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/NLAPDEO17/