

High performance computational techniques for the simplex method

Julian Hall

School of Mathematics
University of Edinburgh

jajhall@ed.ac.uk

CO@Work 2020

16 September 2020



THE UNIVERSITY
of EDINBURGH



- Revision of LP theory and simplex algorithms
- Computational techniques for serial simplex
 - Bound-flipping ratio test (dual simplex)
 - Hyper-sparsity
 - Cost perturbation (dual simplex)
- Computational techniques for parallel simplex
 - Structured LP problems
 - General LP problems

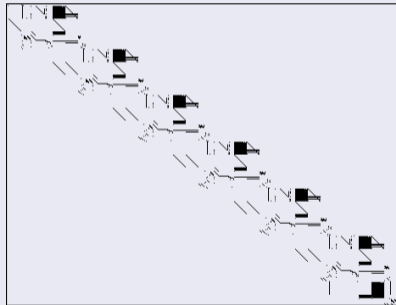
Solving LP problems: Background

$$\text{minimize } f = \mathbf{c}^T \mathbf{x} \quad \text{subject to } \mathbf{Ax} = \mathbf{b} \quad \mathbf{x} \geq 0$$

Background

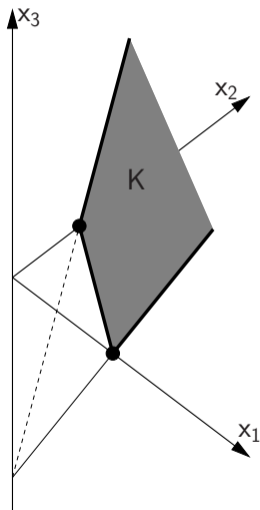
- Fundamental model in optimal decision-making
- Solution techniques
 - Simplex method (1947)
 - Interior point methods (1984)
- Large problems have
 - 10^3 – 10^8 variables
 - 10^3 – 10^8 constraints
- Matrix A is usually **sparse** and may be **structured**

Example



STAIR: 356 rows, 467 columns and 3856 nonzeros

Solving LP problems: Background



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

- A **vertex** of the **feasible region** $K \subset \mathbb{R}^n$ has
 - m **basic** components, $i \in \mathcal{B}$
 - $n - m$ zero **nonbasic** components, $j \in \mathcal{N}$
- The equations and \mathbf{x} are partitioned according to $\mathcal{B} \cup \mathcal{N}$

$$B\mathbf{x}_B + N\mathbf{x}_N = \mathbf{b} \Rightarrow \mathbf{x}_B = B^{-1}(\mathbf{b} - N\mathbf{x}_N) = \hat{\mathbf{b}} - \hat{N}\mathbf{x}_N$$

since the **basis matrix** B is nonsingular

- Reduced objective is then $f = \hat{f} + \hat{\mathbf{c}}_N^T \mathbf{x}_N$, where $\hat{f} = \mathbf{c}_B^T \hat{\mathbf{b}}$ and $\hat{\mathbf{c}}_N^T = \mathbf{c}_N^T - \mathbf{c}_B^T B^{-1} N$
- For $\mathbf{x}_N = 0$, partition yields an optimal solution if there is **Primal feasibility** $\hat{\mathbf{b}} \geq 0$; **Dual feasibility** $\hat{\mathbf{c}}_N \geq 0$

Solving dual LP problems: Optimality and the dual simplex algorithm

- Consider the **dual problem**

$$\text{maximize } f_D = \mathbf{b}^T \mathbf{y} \quad \text{subject to } A^T \mathbf{y} + \mathbf{s} = \mathbf{c} \quad \mathbf{s} \geq 0$$

- Equations, \mathbf{s} and \mathbf{c} partitioned according to $\mathcal{B} \cup \mathcal{N}$ as

$$\begin{bmatrix} B^T \\ N^T \end{bmatrix} \mathbf{y} + \begin{bmatrix} \mathbf{s}_B \\ \mathbf{s}_N \end{bmatrix} = \begin{bmatrix} \mathbf{c}_B \\ \mathbf{c}_N \end{bmatrix} \quad \Rightarrow \quad \begin{cases} \mathbf{y} = B^{-T}(\mathbf{c}_B - \mathbf{s}_B) \\ \mathbf{s}_N = \hat{\mathbf{c}}_N + \hat{N}^T \mathbf{s}_B \end{cases}$$

Reduced objective is $f_D = \hat{f} - \hat{\mathbf{b}}^T \mathbf{s}_B$

- For $\mathbf{s}_B = 0$, partition yields an optimal solution if there is
Primal feasibility $\hat{\mathbf{b}} \geq 0$; **Dual feasibility** $\hat{\mathbf{c}}_N \geq 0$
- **Dual simplex algorithm** for an LP is *primal algorithm* applied to the *dual problem*
- Structure of dual equations allows dual simplex algorithm to be applied to primal simplex tableau

Primal simplex algorithm

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

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

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

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

Dual simplex algorithm: Choose a row

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

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



	\mathcal{N}	RHS
\mathcal{B}		$\hat{\mathbf{b}}$ \hat{b}_p

Dual simplex algorithm: Choose a column

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

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

Scan $\hat{c}_j / \hat{a}_{pj} < 0$ for q to leave \mathcal{N}

	\mathcal{N}	RHS
\mathcal{B}		
		

Dual simplex algorithm: Update reduced costs and RHS

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

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

Scan $\hat{c}_j / \hat{a}_{pj} < 0$ for q to leave \mathcal{N}

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

Update $\hat{\mathbf{b}} := \hat{\mathbf{b}} - \alpha_P \hat{\mathbf{a}}_q$ $\alpha_P = \hat{b}_p / \hat{a}_{pq}$

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

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

Dual simplex algorithm: Data required

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

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

Scan $\hat{c}_j/\hat{a}_{pj} < 0$ for q to leave \mathcal{N}

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

Update $\hat{\mathbf{b}} := \hat{\mathbf{b}} - \alpha_P \hat{\mathbf{a}}_q$ $\alpha_P = \hat{b}_p/\hat{a}_{pq}$

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

Data required

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

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

Dual simplex algorithm

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

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

Scan $\hat{c}_j/\hat{a}_{pj} < 0$ for q to leave \mathcal{N}

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

Update $\hat{\mathbf{b}} := \hat{\mathbf{b}} - \alpha_P \hat{\mathbf{a}}_q$ $\alpha_P = \hat{b}_p/\hat{a}_{pq}$

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

Data required

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

Why does it work?

Objective improves by $\frac{\hat{b}_p \times \hat{c}_q}{\hat{a}_{pq}}$ each iteration

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

Solving LP problems: Primal or dual simplex?

Primal simplex algorithm

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

Dual simplex algorithm

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

Simplex method: Computation

Standard simplex method (SSM): Major computational component

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

Update of tableau: $\hat{N} := \hat{N} - \frac{1}{\hat{a}_{pq}} \hat{\mathbf{a}}_q \hat{\mathbf{a}}_p^T$

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

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

Revised simplex method (RSM): Major computational components

Pivotal row via $B^T \boldsymbol{\pi}_p = \mathbf{e}_p$ **BTRAN** and $\hat{\mathbf{a}}_p^T = \boldsymbol{\pi}_p^T N$ **PRICE**

Pivotal column via $B \hat{\mathbf{a}}_q = \mathbf{a}_q$ **FTRAN** Represent B^{-1} **INVERT**

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

Representing B^{-1} : **INVERT**

Form $B = LU$ using sparsity-exploiting Markowitz technique

- L unit lower triangular
- U upper triangular

Representing B^{-1} : **UPDATE**

- Exploit $\bar{B} = B + (\mathbf{a}_q - B\mathbf{e}_p)\mathbf{e}_p^T$ to limit refactorization
- Many schemes: simplest is **product form**

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

where E is easily invertible

Simplex method: Mittelmann test set

Industry standard set of 40 LP problems

	Rows	Cols	Nonzeros	$\frac{\text{Rows}}{\text{Cols}}$	$\frac{\text{Nonzeros}}{\text{Rows} \times \text{Cols}}$	$\frac{\text{Nonzeros}}{\max(\text{Rows}, \text{Cols})}$
Min	960	1560	38304	1/255	0.0005%	2.2
Geomean	54256	72442	910993	0.75	0.02%	6.5
Max	986069	1259121	11279748	85	16%	218.0

Mittelmann solution time measure

- Unsolved problems given “timeout” solution time
- Shift all solution times up by 10s
- Compute geometric mean of logs of shifted times
- Solution time measure is exponent of geometric mean shifted down by 10s

Dual simplex: Bound-flipping Ratio Test (BFRT)

Dual simplex: Bound-flipping Ratio Test (BFRT)

- General bounded equality problem is

$$\text{minimize } f = \mathbf{c}^T \mathbf{x} \quad \text{subject to } [A \quad I] \mathbf{x} = \mathbf{b} \quad \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}$$

- At a vertex, nonbasic variables \mathbf{x}_N take values \mathbf{v}_N of lower or upper bounds
- Equations and \mathbf{x} partitioned according to $\mathcal{B} \cup \mathcal{N}$ as

$$B\mathbf{x}_B + N\mathbf{x}_N = \mathbf{b} \quad \Rightarrow \quad \mathbf{x}_B = B^{-1}(\mathbf{b} - N\mathbf{x}_N) = \hat{\mathbf{b}} - \hat{N}\delta_N$$

where $\mathbf{x}_N = \mathbf{v}_N + \delta_N$ and $\hat{\mathbf{b}} = B^{-1}(\mathbf{b} - N\mathbf{v}_N)$

- For $\delta_N = 0$, the partition yields an optimal solution if there is

$$\text{Primal feasibility } \mathbf{l}_B \leq \hat{\mathbf{b}} \leq \mathbf{u}_B \quad \text{Dual feasibility } \begin{cases} \hat{c}_j \geq 0 & x_j = l_j \\ \hat{c}_j \leq 0 & x_j = u_j \end{cases} \quad j \in \mathcal{N}$$

Dual simplex: Bound-flipping Ratio Test (BFRT)

- Reduced objective is

$$f_D = \hat{f} - (\hat{\mathbf{b}} - \mathbf{I})^T \mathbf{s}_B^+ - (\mathbf{u} - \hat{\mathbf{b}})^T \mathbf{s}_B^-$$

- Suppose $p \in \mathcal{B}$ is chosen such that $\hat{b}_p < l_p$ so s_p is increased from zero

- As f_D increases, some s_j , $j \in \mathcal{N}$ is zeroed

- If x_j “flips bound” then \hat{b}_p increases

- If still have $\hat{b}_p < l_p$, then

- s_j changes sign
- s_p can be increased further

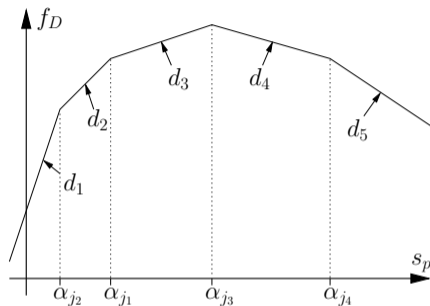
- In general

- Find $\{\alpha_1, \alpha_2, \dots\}$ (easily)

- Sort breakpoints as $\{\alpha_{j_1}, \alpha_{j_2}, \dots\}$

- Analyse $d_1 = l_p - \hat{b}_p > 0$ and (by recurrence) $\{d_2, d_3, \dots\}$ for sign change

- Multiple iteration “progress”, with only one basis change and set of B/FTRANs



Simplex method: Exploiting hyper-sparsity

Simplex method: Exploiting hyper-sparsity

Recall: major computational components

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

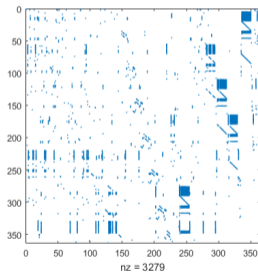
Phenomenon of hyper-sparsity

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

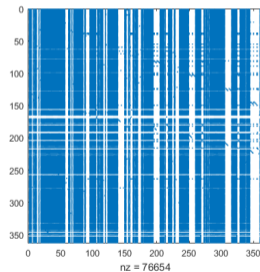
Simplex method: Exploiting hyper-sparsity

Simplex method: Inverse of a sparse matrix and solution of $Bx = b$

Optimal B for LP problem stair
has density 2.5%



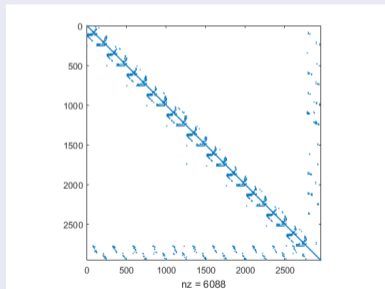
B^{-1} has density of 58%, so $B^{-1}b$
is typically dense



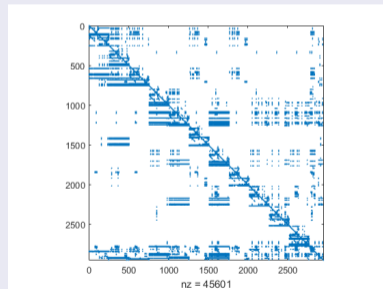
Simplex method: Exploiting hyper-sparsity

Simplex method: Inverse of a sparse matrix and solution of $Bx = b$

Optimal B for LP problem pds-02
has density 0.07%



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



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

Simplex method: Exploiting hyper-sparsity

Recall: Solve $Lx = \mathbf{b}$ using

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

```
   $\mathbf{r} = \mathbf{b}$ 
```

```
  for all  $j \in \{1, \dots, m\}$  do
```

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

```
       $r_i = r_i - L_{ij}r_j$ 
```

```
   $\mathbf{x} = \mathbf{r}$ 
```

When \mathbf{b} is **sparse**

- Inefficient until \mathbf{r} fills in

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)

Simplex method: Exploiting 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)

Simplex method: Exploiting hyper-sparsity - effectiveness

Testing environment

- Mittelman test set of 40 LPs
- HiGHS dual simplex solver
- Time limit of 10,000 seconds

Results

- When exploiting hyper-sparsity: solves 37 problems
- When not exploiting hyper-sparsity (in BTRAN, FTRAN and PRICE): solves 34 problems

	Min	Geomean	Max
Iteration count increase	0.75	1.08	3.17
Solution time increase	0.83	2.31	67.13
Iteration speed decrease	0.92	2.14	66.43
Mittelman measure	2.57		

Dual simplex: Cost perturbation

Dual simplex: Cost perturbation

Dual degeneracy

- If some nonbasic dual values $\mathbf{c}_N^T - \mathbf{c}_B^T B^{-1} N$ are zero, the vertex is **dual degenerate**
- At a dual degenerate vertex, an iteration of the dual simplex algorithm may not lead to a strict increase in the dual objective
- Stalling or cycling may occur

Cost perturbation

- Add a small random value to some/all of the cost coefficients \mathbf{c}
- Nonbasic dual values then (at worst) take small positive values
- An iteration of the dual simplex algorithm yields (at least) a small positive increase in the dual objective
- When optimal, remove perturbations
- May require simplex iterations to regain optimality

Results using Mittelmann test set

- With cost perturbation: HiGHS solves 37/40 problems
- Without cost perturbation: solves 27 problems

	Min	Geomean	Max
Iteration count increase	0.80	1.36	7.21
Solution time increase	0.57	1.46	13.31
Iteration speed decrease	0.49	1.07	4.02
Mittelmann measure	3.80		

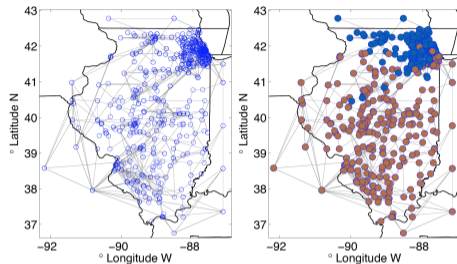
Computational techniques for parallel simplex: Structured LP problems

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

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

- 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
 - Solve nodes using parallel dual simplex solver PIPS-S

Lubin, Petra *et al.* (2011)

Convenient to permute the LP thus:

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

PIPS-S: Exploiting problem structure

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

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

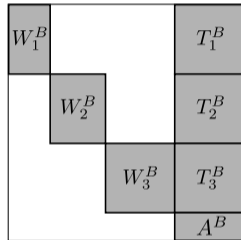
- W_i^B are columns corresponding to n_i^B basic variables in scenario i

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

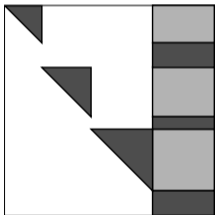
$$B = \begin{bmatrix} W_1^B & & & T_1^B \\ & \ddots & & \vdots \\ & & W_N^B & T_N^B \\ & & & A^B \end{bmatrix}$$



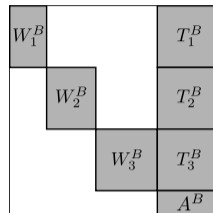
- B is nonsingular so
 - W_i^B are “tall”: full column rank
 - $[W_i^B \quad T_i^B]$ are “wide”: full row rank
 - A^B is “wide”: full row rank
- Scope for parallel inversion is immediate and well known

PIPS-S: Exploiting problem structure

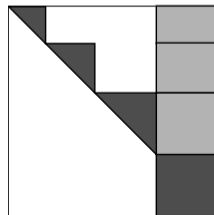
- Eliminate sub-diagonal entries in each W_i^B (independently)



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



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



Scope for data parallelism

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

Implementation

- Written in C++ with MPI
- Dual simplex
- Based on NLA routines in c1p
- Distribute problem data over processors
- Perform data-parallel BTRAN, FTRAN and PRICE over processors

Lubin, H, Petra and Anitescu (2013)

On Fusion cluster: Performance relative to c1p

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

Computational techniques for parallel simplex: General LP problems

Overview

- Written in C++ to study parallel simplex
- Dual simplex with standard algorithmic enhancements
- Efficient numerical linear algebra

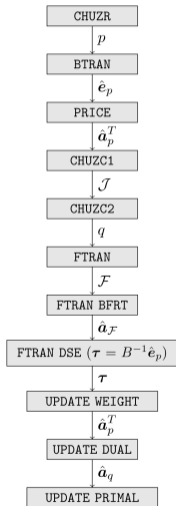
Concept

- High performance serial solver (`hso1`)
- 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`)

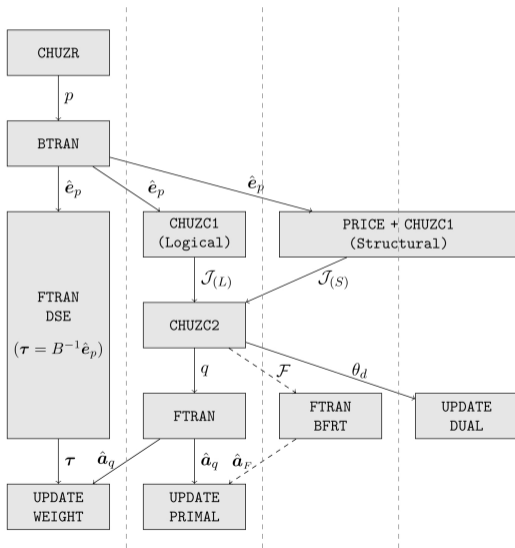
Huangfu and H

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



- Overlap the expensive FTRAN with PRICE, CHUZC, and then the subsequent (cheaper) FTRANs
- Exploit data parallelism in PRICE to form $\hat{\mathbf{a}}_p^T = \boldsymbol{\pi}_p^T N$, and in the expensive pass of CHUZC
- Overlap the cheaper FTRANs, and then the operations to update edge weights and the reduced RHS
- Other components performed serially
- Only four worthwhile threads unless $n \gg m$ so PRICE dominates

Huangfu and H (2014)

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*

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

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

Huangfu and H (2011–2014)

HiGHS: Multiple iteration parallelism with pami - effectiveness

Serial overhead of pami

- HiGHS pami solver in serial: solves 34/40 problems

	Min	Geomean	Max
Iteration count increase	0.43	1.02	2.98
Solution time increase	0.31	1.62	5.36
Iteration speed decrease	0.69	1.59	5.11
Mittelmann measure	2.08		

Parallel speed-up of pami with 8 threads

	Min	Geomean	Max
Iteration count decrease	1.00	1.00	1.00
Solution time decrease	1.15	1.88	2.39
Iteration speed increase	1.15	1.88	2.39

HiGHS: Multiple iteration parallelism with pami - effectiveness

Performance enhancement using parallel pami with 8 threads

	Min	Geomean	Max
Iteration count decrease	0.34	0.98	2.34
Solution time decrease	0.34	1.16	6.44
Iteration speed increase	0.38	1.18	2.75
Mittelman measure	1.21		

Observations

- There is significant scope to improve pami performance further
- Use pami tactically: switch it off if it is ineffective

Commercial impact

- Huangfu applied the parallel dual simplex techniques within the xpress solver
- For much of 2013–2018 the xpress simplex solver was the best in the world

High performance simplex

- Many (more) algorithmic and computational tricks in serial
- Parallel simplex has some impact on performance
- Tune techniques to problem at run-time



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.