The practical revised simplex method (Part 2)

Julian Hall

School of Mathematics

University of Edinburgh

January 25th 2007

Overview (Part 2)

- Practical implementation of the revised simplex method
  - Representation of B^{-1}

Overview (Part 2)

- Practical implementation of the revised simplex method
  - Representation of B^{-1}
  - Strategies for CHUZC

Overview (Part 2)

- Practical implementation of the revised simplex method
  - Representation of B^{-1}
  - Strategies for CHUZC
  - Partial pricing

Overview (Part 2)

- Practical implementation of the revised simplex method
  - Representation of B^{-1}
  - Strategies for CHUZC
  - Partial pricing
  - Hyper-sparsity

Overview (Part 2)

- Practical implementation of the revised simplex method
  - Representation of B^{-1}
  - Strategies for CHUZC
  - Partial pricing
  - Hyper-sparsity
- Parallel simplex

Overview (Part 2)

- Practical implementation of the revised simplex method
  - Representation of B^{-1}
  - Strategies for CHUZC
  - Partial pricing
  - Hyper-sparsity
- Parallel simplex
- Research frontiers

Representing B^{-1}

- The key to the efficiency of the revised simplex method is the representation of B^{-1}.

Representing B^{-1}

- The key to the efficiency of the revised simplex method is the representation of B^{-1}.
- Periodically: **INVERT** operation determines representation of B^{-1} using Gaussian elimination

Representing B^{-1}

- The key to the efficiency of the revised simplex method is the representation of B^{-1}.
- Periodically: **INVERT** operation determines representation of B^{-1} using Gaussian elimination
- Each iteration: **UPDATE** operation updates representation of B^{-1} according to

$$B^{-1} := \left(I - \frac{(\hat{\boldsymbol{a}}_q - \boldsymbol{e}_p)\boldsymbol{e}_p^T}{\hat{a}_{pq}} \right) B^{-1}$$

INVERT

- The representation of B^{-1} is based on LU decomposition from Gaussian elimination.
- Classical factors $B = LU$ fill in.
- Perform row and column interchanges to maintain sparsity: yields $PBQ = LU$.

INVERT

- The representation of B^{-1} is based on LU decomposition from Gaussian elimination.
- Classical factors $B = LU$ fill in.
- Perform row and column interchanges to maintain sparsity: yields $PBQ = LU$.
  - Markowitz pivot selection criterion (1957) preserves sparsity well.

INVERT

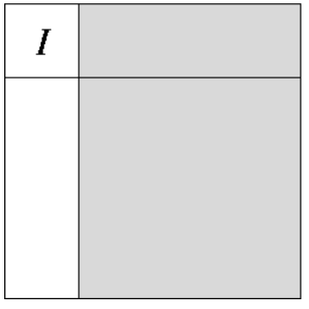
- The representation of B^{-1} is based on LU decomposition from Gaussian elimination.
- Classical factors $B = LU$ fill in.
- Perform row and column interchanges to maintain sparsity: yields $PBQ = LU$.
  - Markowitz pivot selection criterion (1957) preserves sparsity well.
  - Tomlin pivot selection criterion (1972) more efficient for most LP basis matrices.

INVERT

- The representation of B^{-1} is based on LU decomposition from Gaussian elimination.
- Classical factors $B = LU$ fill in.
- Perform row and column interchanges to maintain sparsity: yields $PBQ = LU$.
  - Markowitz pivot selection criterion (1957) preserves sparsity well.
  - Tomlin pivot selection criterion (1972) more efficient for most LP basis matrices.
- Permutations P and Q can be absorbed into the ordering of basic variables and constraints.

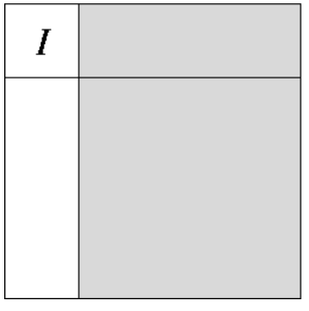
Tomlin INVERT

1 Find identity columns in B

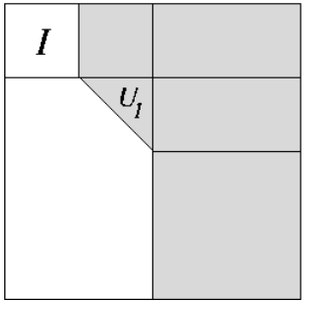


Tomlin INVERT

1 Find identity columns in B

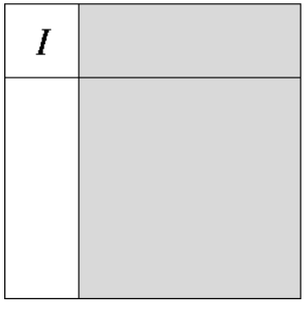


2a Find columns with only one nonzero

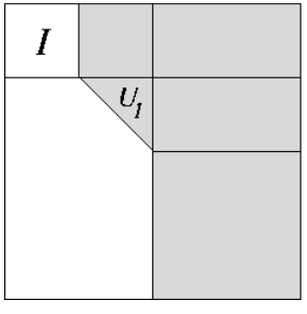


Tomlin INVERT

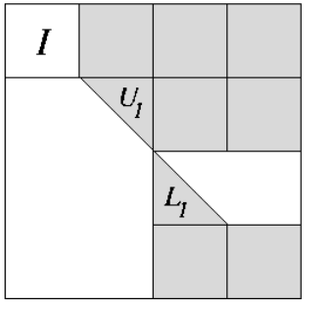
1 Find identity columns in B



2a Find columns with only one nonzero

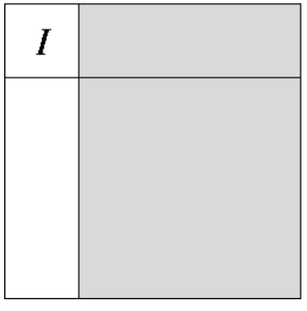


2b Find rows with only one nonzero

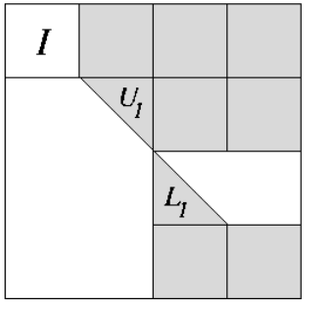


Tomlin INVERT

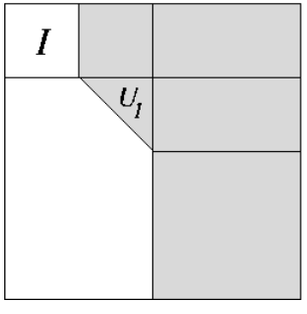
1 Find identity columns in B



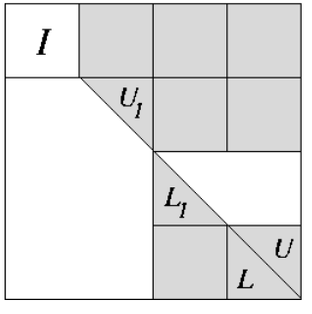
2b Find rows with only one nonzero



2a Find columns with only one nonzero

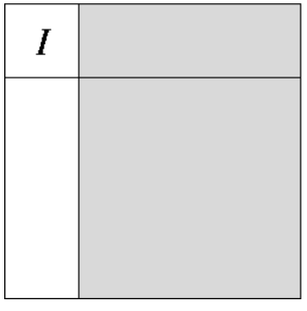


3 Factorise the "bump" using GE

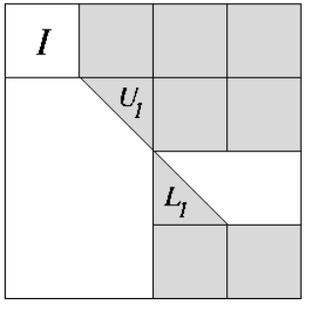


Tomlin INVERT

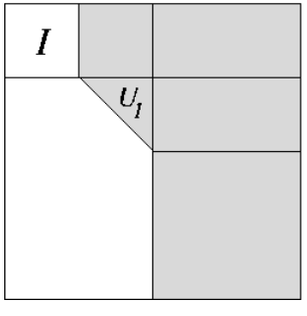
1 Find identity columns in B



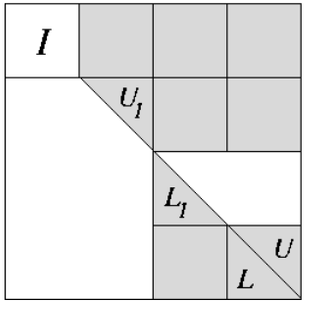
2b Find rows with only one nonzero



2a Find columns with only one nonzero



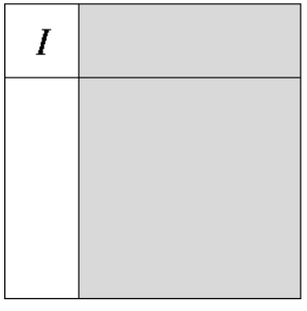
3 Factorise the "bump" using GE



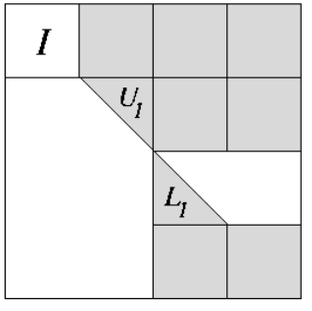
- Efficiency depends on the bump being small

Tomlin INVERT

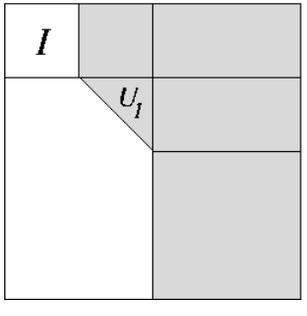
1 Find identity columns in B



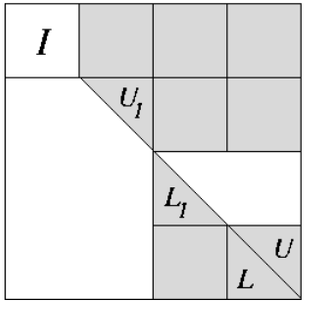
2b Find rows with only one nonzero



2a Find columns with only one nonzero



3 Factorise the "bump" using GE



- Efficiency depends on the bump being small
- For many practical LP problems there is little or no bump

Using $B = LU$ from Gaussian elimination

- Gaussian elimination yields

$$B = LU \quad \text{where} \quad L = \begin{bmatrix} 1 & & & \\ & \ddots & & \\ & & 1 & \\ \boldsymbol{l}_1 & \dots & \boldsymbol{l}_{n-1} & 1 \end{bmatrix} \quad \text{and} \quad U = \begin{bmatrix} u_{11} & \boldsymbol{u}_2 & \dots & \boldsymbol{u}_n \\ & u_{22} & & \\ & & \ddots & \\ & & & u_{nn} \end{bmatrix}$$

Using $B = LU$ from Gaussian elimination

- Gaussian elimination yields

$$B = LU \quad \text{where} \quad L = \begin{bmatrix} 1 & & & \\ & \ddots & & \\ & & 1 & \\ \boldsymbol{l}_1 & \dots & \boldsymbol{l}_{n-1} & 1 \end{bmatrix} \quad \text{and} \quad U = \begin{bmatrix} u_{11} & \boldsymbol{u}_2 & \dots & \boldsymbol{u}_n \\ & u_{22} & & \\ & & \ddots & \\ & & & u_{nn} \end{bmatrix}$$

- $B\boldsymbol{x} = \boldsymbol{r}$ is solved by solving $L\boldsymbol{y} = \boldsymbol{r}$ and $U\boldsymbol{x} = \boldsymbol{y}$.

Using $B = LU$ from Gaussian elimination

- Gaussian elimination yields

$$B = LU \quad \text{where} \quad L = \begin{bmatrix} 1 & & & \\ & \ddots & & \\ & & 1 & \\ \boldsymbol{l}_1 & \dots & \boldsymbol{l}_{n-1} & 1 \end{bmatrix} \quad \text{and} \quad U = \begin{bmatrix} u_{11} & \boldsymbol{u}_2 & \dots & \boldsymbol{u}_n \\ & u_{22} & & \\ & & \ddots & \\ & & & u_{nn} \end{bmatrix}$$

- $B\boldsymbol{x} = \boldsymbol{r}$ is solved by solving $L\boldsymbol{y} = \boldsymbol{r}$ and $U\boldsymbol{x} = \boldsymbol{y}$.
- In practice, $\boldsymbol{r}$ is transformed into $\boldsymbol{x}$ as follows.

$$\begin{aligned} & & \boldsymbol{r} &:= \boldsymbol{r} - r_k \boldsymbol{l}_k & & k = 1, \dots, n-1; \\ r_k &:= \frac{r_k}{u_{kk}}, & \boldsymbol{r} &:= \boldsymbol{r} - r_k \boldsymbol{u}_k & & k = n, \dots, 2; \\ r_1 &:= \frac{r_1}{u_{11}}. & & & & \end{aligned}$$

LU factors as a product of elementary matrices

$B = LU$ may be written as

$$B = (\prod_{k=1}^{n-1} L_k)(\prod_{k=1}^{n} U_k)$$

where

$$L_k = \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & & 1 & \\ & & \boldsymbol{l}_k & & \ddots \\ & & & & & 1 \end{bmatrix} \quad \text{and} \quad U_k = \begin{bmatrix} 1 & & & & & \\ & \ddots & & \boldsymbol{u}_k & & \\ & & 1 & & & \\ & & & u_{kk} & & \\ & & & & 1 & \\ & & & & & \ddots \\ & & & & & & 1 \end{bmatrix}.$$

LU factors as a product of elementary matrices

$B = LU$ may be written as

$$B = (\prod_{k=1}^{n-1} L_k)(\prod_{k=1}^{n} U_k)$$

where

$$L_k = \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & & 1 & & \\ & & \boldsymbol{l}_k & & \ddots & \\ & & & & & 1 \end{bmatrix} \quad \text{and} \quad U_k = \begin{bmatrix} 1 & & & & & \\ & \ddots & & \boldsymbol{u}_k & & \\ & & 1 & & & \\ & & & u_{kk} & & \\ & & & & 1 & \\ & & & & & \ddots \\ & & & & & & 1 \end{bmatrix}.$$

So

$$B^{-1} = (\prod_{k=1}^{n} U_k^{-1})(\prod_{k=1}^{n-1} L_k^{-1})$$

Product form and the eta file

- In general, $B^{-1} = \prod_{k=K_I}^{1} E_k^{-1}$ where

$$E_k^{-1} = \begin{bmatrix} 1 & & & -\eta_1^k & & \\ & \ddots & & \vdots & & \\ & & 1 & -\eta_{p_k-1}^k & & \\ & & & 1 & & \\ & & & -\eta_{p_k+1}^k & 1 & \\ & & & \vdots & & \ddots \\ & & & -\eta_m^k & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & & \mu^k & & \\ & & & & 1 & \\ & & & & & \ddots \\ & & & & & 1 \end{bmatrix}.$$

Product form and the eta file

- In general, $B^{-1} = \prod_{k=K_I}^{1} E_k^{-1}$ where

$$E_k^{-1} = \begin{bmatrix} 1 & & & -\eta_1^k & & \\ & \ddots & & \vdots & & \\ & & 1 & -\eta_{p_k-1}^k & & \\ & & & 1 & & \\ & & & -\eta_{p_k+1}^k & 1 & \\ & & & \vdots & & \ddots \\ & & & -\eta_m^k & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & & \mu^k & & \\ & & & & 1 & \\ & & & & & \ddots \\ & & & & & 1 \end{bmatrix}.$$

- The **pivot** is μ_k; the index of the **pivotal row** is p_k.

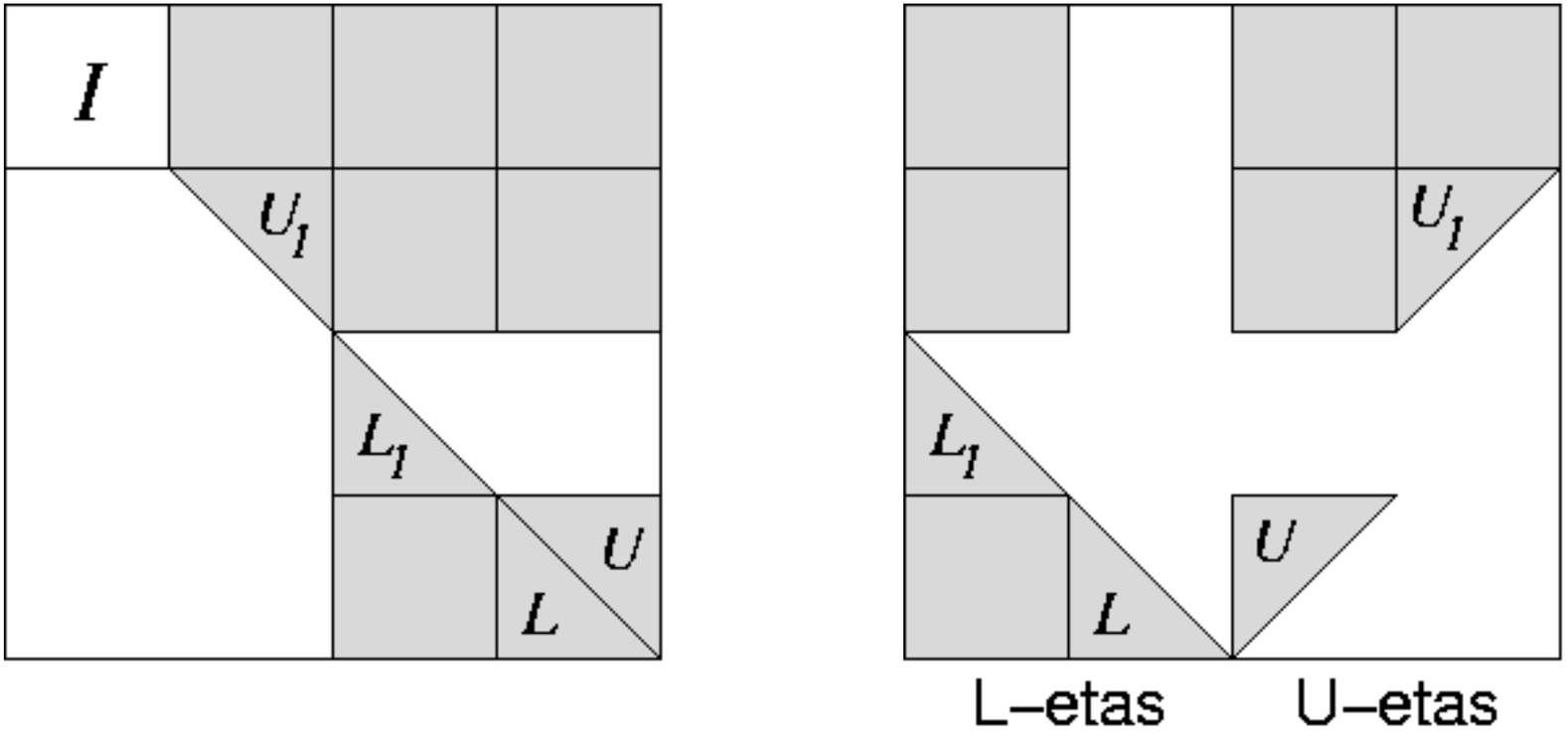
Product form and the eta file

- In general, $B^{-1} = \prod_{k=K_I}^{1} E_k^{-1}$ where

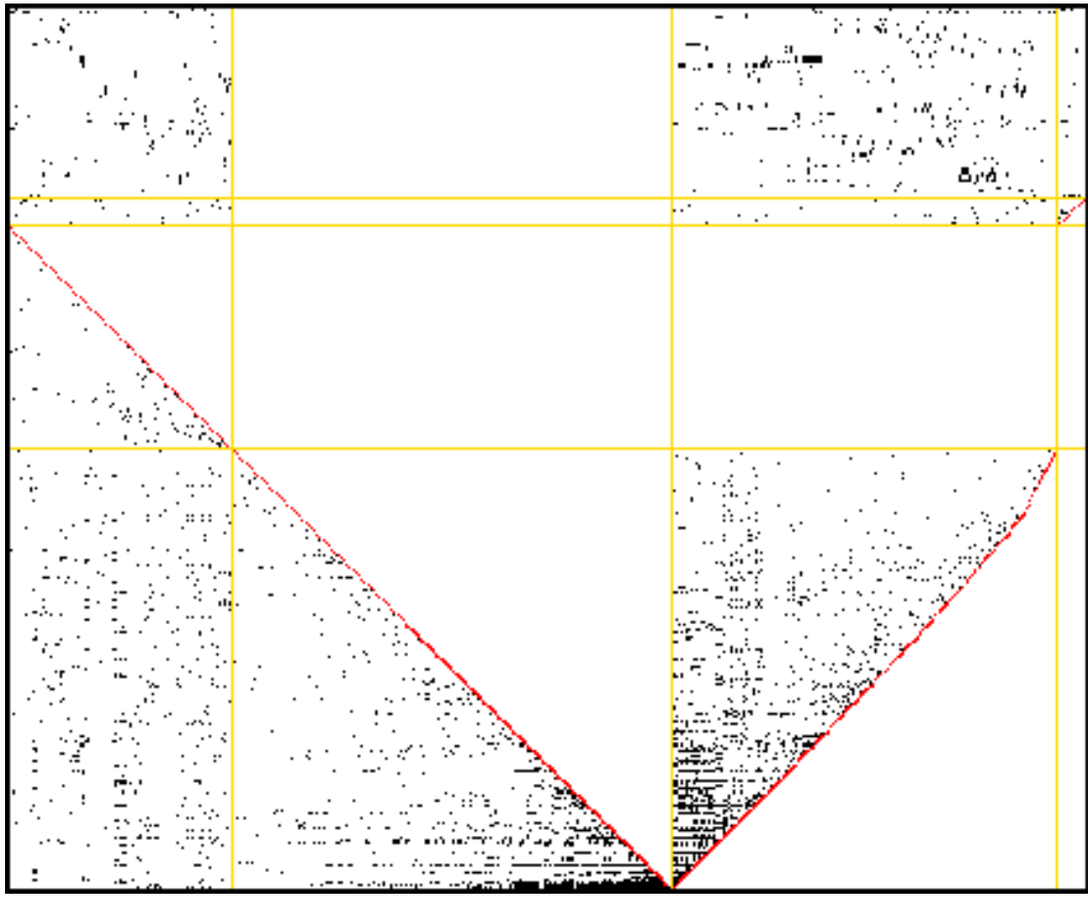
$$E_k^{-1} = \begin{bmatrix} 1 & & & -\eta_1^k & & \\ & \ddots & & \vdots & & \\ & & 1 & -\eta_{p_k-1}^k & & \\ & & & 1 & & \\ & & & -\eta_{p_k+1}^k & 1 & \\ & & & \vdots & & \ddots \\ & & & -\eta_m^k & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & & \mu^k & & \\ & & & & 1 & \\ & & & & & \ddots \\ & & & & & 1 \end{bmatrix}.$$

- The **pivot** is μ_k; the index of the **pivotal row** is p_k.
- The set $\{p_k, \mu_k, \boldsymbol{\eta}_k\}_{k=1}^{K_I}$ is known as the **eta file**.

Eta file from Tomlin INVERT

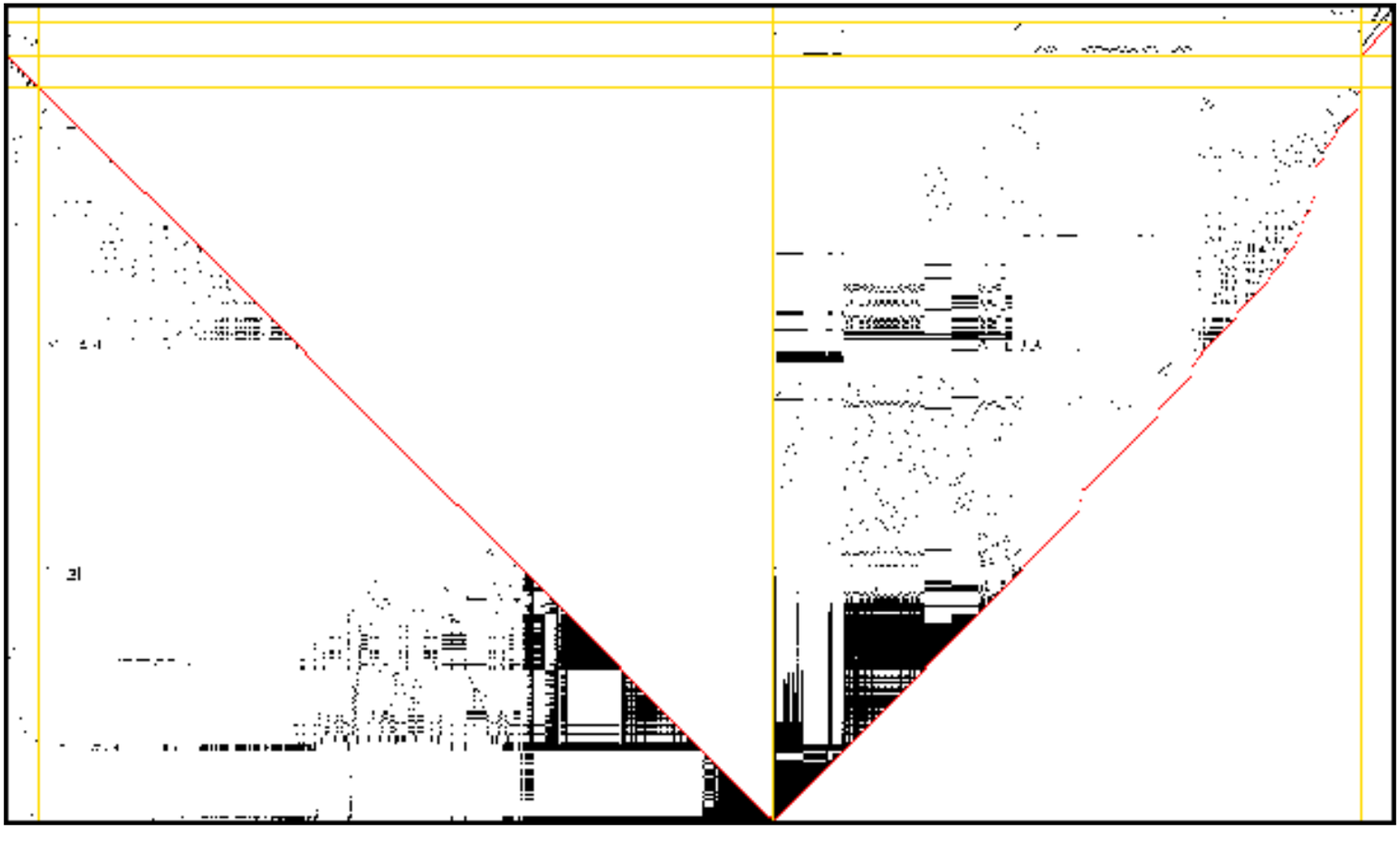


Eta file (25fv47)



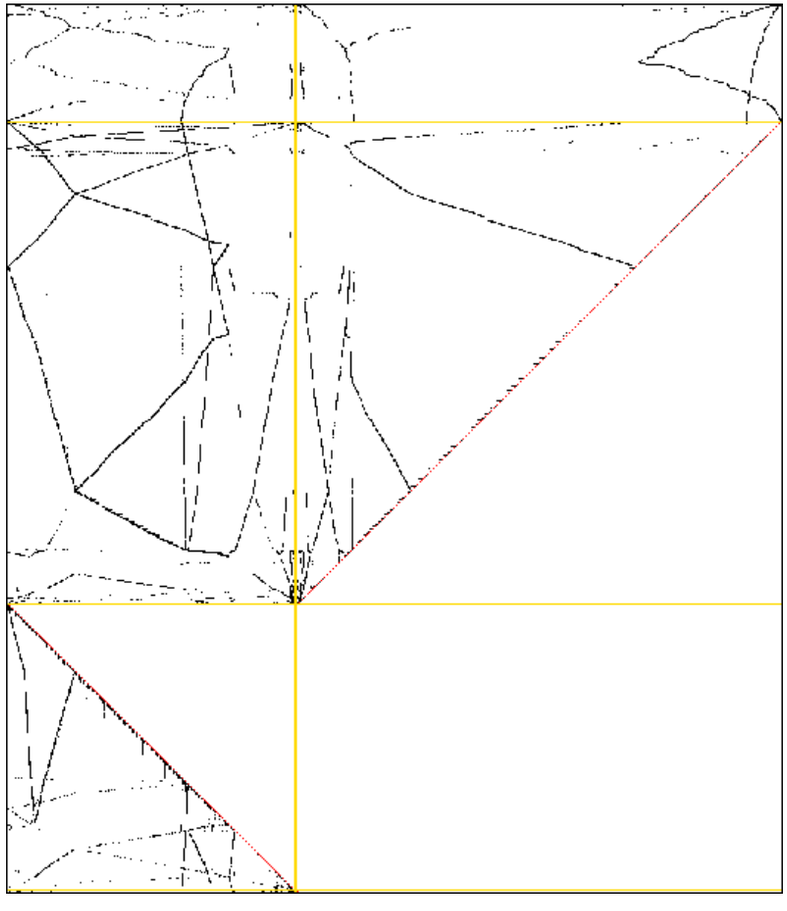
Basis matrix is of dimension is 821; bump is of dimension 411

Eta file (`stair`)



Basis matrix is of dimension is 356; bump is of dimension 324

Eta file (sgpf5y6)



Basis matrix is of dimension is 246077; bump is of dimension 32

Using the eta file

- The solution of $B\boldsymbol{x} = \boldsymbol{r}$ is formed by transforming $\boldsymbol{r}$ into $\boldsymbol{x}$ as follows.

$$r_{p_k} := \mu_k r_{p_k}, \quad \boldsymbol{r} := \boldsymbol{r} - r_{p_k}\boldsymbol{\eta}_k \qquad k = 1, \ldots, K_I.$$

Using the eta file

- The solution of $B\boldsymbol{x} = \boldsymbol{r}$ is formed by transforming $\boldsymbol{r}$ into $\boldsymbol{x}$ as follows.

$$r_{p_k} := \mu_k r_{p_k}, \quad \boldsymbol{r} := \boldsymbol{r} - r_{p_k}\boldsymbol{\eta}_k \qquad k = 1, \ldots, K_I.$$

  This requires a sparse eta-vector to be added to $\boldsymbol{r}$ (up to) K_I times.

Using the eta file

- The solution of $B\boldsymbol{x} = \boldsymbol{r}$ is formed by transforming $\boldsymbol{r}$ into $\boldsymbol{x}$ as follows.

$$r_{p_k} := \mu_k r_{p_k}, \quad \boldsymbol{r} := \boldsymbol{r} - r_{p_k}\boldsymbol{\eta}_k \qquad k = 1, \ldots, K_I.$$

  This requires a sparse eta-vector to be added to $\boldsymbol{r}$ (up to) K_I times.

- The solution of $B^T\boldsymbol{x} = \boldsymbol{r}$ is $(\boldsymbol{r}^T B^{-1})^T$ and is formed by transforming $\boldsymbol{r}$ into $\boldsymbol{x}$ as follows.

$$r_{p_k} := \mu_k(r_{p_k} - \boldsymbol{r}^T\boldsymbol{\eta}_k) \qquad k = K_I, \ldots, 1$$

Using the eta file

- The solution of $B\boldsymbol{x} = \boldsymbol{r}$ is formed by transforming $\boldsymbol{r}$ into $\boldsymbol{x}$ as follows.

$$r_{p_k} := \mu_k r_{p_k}, \quad \boldsymbol{r} := \boldsymbol{r} - r_{p_k}\boldsymbol{\eta}_k \qquad k = 1, \ldots, K_I.$$

  This requires a sparse eta-vector to be added to $\boldsymbol{r}$ (up to) K_I times.

- The solution of $B^T\boldsymbol{x} = \boldsymbol{r}$ is $(\boldsymbol{r}^T B^{-1})^T$ and is formed by transforming $\boldsymbol{r}$ into $\boldsymbol{x}$ as follows.

$$r_{p_k} := \mu_k(r_{p_k} - \boldsymbol{r}^T\boldsymbol{\eta}_k) \qquad k = K_I, \ldots, 1$$

  This requires the inner product between a sparse eta-vector and $\boldsymbol{r}$ (up to) K_I times.

UPDATE

Update of B^{-1} defined by

$$B^{-1} \quad := \quad \left(I - \frac{(\hat{\boldsymbol{a}}_q - \boldsymbol{e}_p)\boldsymbol{e}_p^T}{\hat{a}_{pq}} \right) B^{-1}$$

UPDATE

Update of B^{-1} defined by

$$
\begin{aligned}
B^{-1} \quad &:= \quad \left(I - \frac{(\hat{\boldsymbol{a}}_q - \boldsymbol{e}_p)\boldsymbol{e}_p^T}{\hat{a}_{pq}} \right) B^{-1} \\
&= \quad \begin{bmatrix} 1 & & & -\hat{a}_{1q} & & \\ & \ddots & & \vdots & & \\ & & 1 & -\hat{a}_{p-1\,q} & & \\ & & & 1 & & \\ & & & -\hat{a}_{p+1\,q} & 1 & \\ & & & \vdots & & \ddots \\ & & & -\hat{a}_{mq} & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & & 1/\hat{a}_{pq} & & \\ & & & & 1 & \\ & & & & & \ddots \\ & & & & & 1 \end{bmatrix} B^{-1}
\end{aligned}
$$

UPDATE

Update of B^{-1} defined by

$$
\begin{aligned}
B^{-1} &:= \left(I - \frac{(\hat{\boldsymbol{a}}_q - \boldsymbol{e}_p)\boldsymbol{e}_p^T}{\hat{a}_{pq}} \right) B^{-1} \\
&= \begin{bmatrix} 1 & & & -\hat{a}_{1q} & & \\ & \ddots & & \vdots & & \\ & & 1 & -\hat{a}_{p-1\,q} & & \\ & & & 1 & & \\ & & & -\hat{a}_{p+1\,q} & 1 & \\ & & & \vdots & & \ddots \\ & & & -\hat{a}_{mq} & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & & 1/\hat{a}_{pq} & & \\ & & & & 1 & \\ & & & & & \ddots \\ & & & & & 1 \end{bmatrix} B^{-1} \\
&= E_k^{-1} B^{-1}
\end{aligned}
$$

for $p_k = p$, $\mu_k = 1/\hat{a}_{pq}$ and $\boldsymbol{\eta}_k = \hat{\boldsymbol{a}}_q - \hat{a}_{pq}\boldsymbol{e}_p$.

UPDATE (cont.)

- In general, K_U UPDATEs after INVERT

$$B^{-1} = \prod_{k=K_I+K_U}^{1} E_k^{-1}$$

UPDATE (cont.)

- In general, K_U UPDATEs after INVERT

$$B^{-1} = \prod_{k=K_I+K_U}^{1} E_k^{-1}$$

- Periodically it will more efficient, or necessary for numerical stability, to re-INVERT.
- Note that $K_U \ll K_I = O(m)$

FTRAN and BTRAN

Using the representation

$$B^{-1} = \prod_{k=K_I+K_U}^{1} E_k^{-1}$$

FTRAN and BTRAN

Using the representation

$$B^{-1} = \prod_{k=K_I+K_U}^{1} E_k^{-1}$$

- The pivotal column $\hat{\boldsymbol{a}}_q = B^{-1}\boldsymbol{a}_q$ is formed by transforming $\boldsymbol{r} = \boldsymbol{a}_q$ into $\hat{\boldsymbol{a}}_q$ by the **FTRAN** operation

$$r_{p_k} := \mu_k r_{p_k}, \quad \boldsymbol{r} := \boldsymbol{r} - r_{p_k}\boldsymbol{\eta}_k \qquad k = 1, \ldots, K_I + K_U.$$

FTRAN and BTRAN

Using the representation

$$B^{-1} = \prod_{k=K_I+K_U}^{1} E_k^{-1}$$

- The pivotal column $\hat{\boldsymbol{a}}_q = B^{-1}\boldsymbol{a}_q$ is formed by transforming $\boldsymbol{r} = \boldsymbol{a}_q$ into $\hat{\boldsymbol{a}}_q$ by the **FTRAN** operation

$$r_{p_k} := \mu_k r_{p_k}, \quad \boldsymbol{r} := \boldsymbol{r} - r_{p_k}\boldsymbol{\eta}_k \qquad k = 1, \ldots, K_I + K_U.$$

- The vector $\boldsymbol{\pi}^T = \boldsymbol{e}_p^T B^{-1}$ required to find the pivotal row is formed by transforming $\boldsymbol{r} = \boldsymbol{e}_p$ into $\boldsymbol{\pi}$ by the **BTRAN** operation

$$r_{p_k} := \mu_k(r_{p_k} - \boldsymbol{r}^T\boldsymbol{\eta}_k) \qquad k = K_I + K_U, \ldots, 1.$$

Strategies for CHUZC

'Most negative reduced cost' rule is not the best.

Example:

$$\begin{array}{lrcrcrcl} \text{minimize} & f & = & -2x_1 & - & x_2 & & \\ \text{subject to} & & & 100x_1 & + & x_2 & \leq & 1 \\ & & & x_1 \geq 0 & \text{and} & x_2 \geq 0. & & \end{array}$$

Strategies for CHUZC

'Most negative reduced cost' rule is not the best.

Example:

$$\begin{array}{lrcrcrcl} \text{minimize} & f & = & -2x_1 & - & x_2 & & \\ \text{subject to} & & & 100x_1 & + & x_2 & \leq & 1 \\ & & & x_1 \geq 0 & \text{and} & x_2 \geq 0. & & \end{array}$$

Slack variable y_1 puts problem in standard form

$$\begin{array}{lrcrcrcrcl} \text{minimize} & f & = & -2x_1 & - & x_2 & & & \\ \text{subject to} & & & 100x_1 & + & x_2 & + & y_1 & = & 1 \\ & & & x_1 \geq 0, & & x_2 \geq 0 & \text{and} & y_1 \geq 0. & & \end{array}$$

Example (cont.)

$$\begin{array}{lrcl} \text{minimize} & f & = & -2x_1 - x_2 \\ \text{subject to} & & & 100x_1 + x_2 + y_1 = 1 \\ & & & x_1 \geq 0, \quad x_2 \geq 0 \quad \text{and} \quad y_1 \geq 0. \end{array}$$

- Starting from logical basis $y_1 = 1$:
- Reduced costs are

$$\begin{array}{ll} -2 & \text{for} \quad x_1 \\ -1 & \text{for} \quad x_2 \end{array}$$

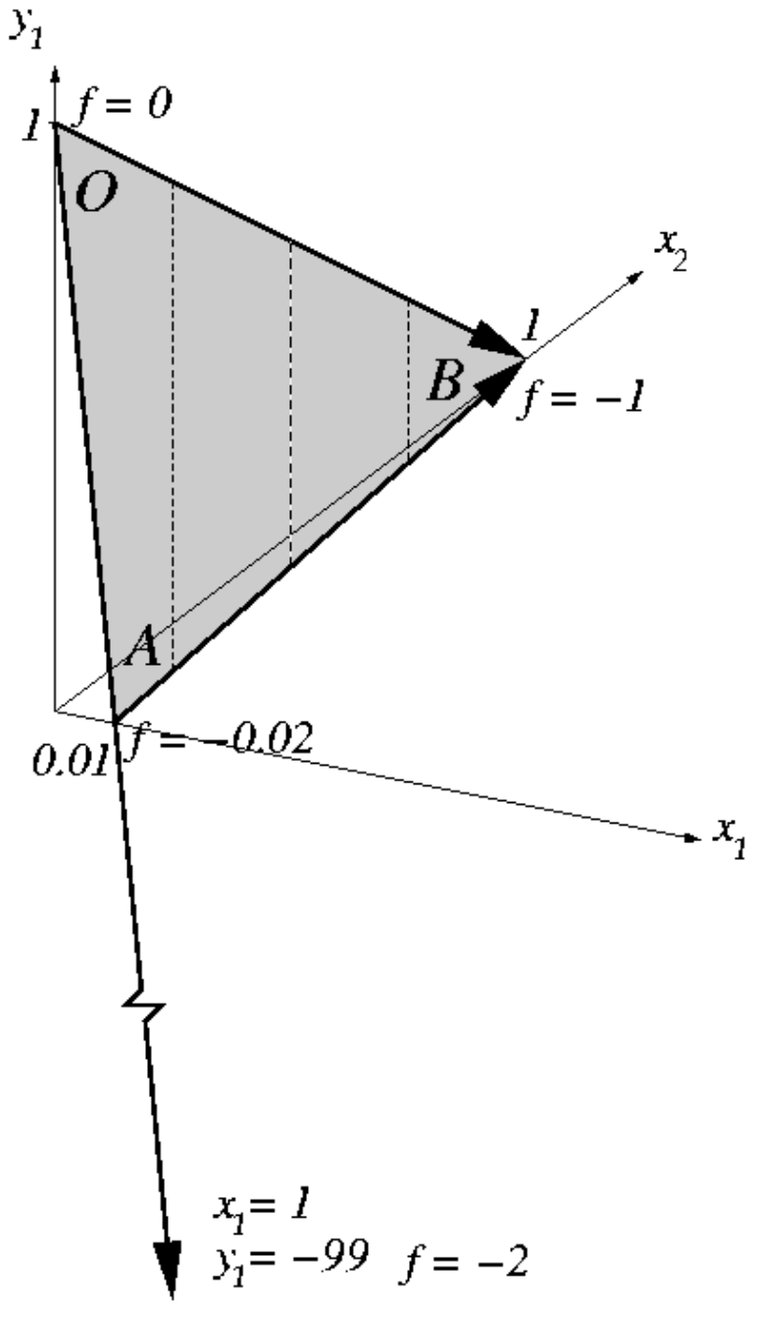
Example (cont.)

$$\begin{array}{lrcl} \text{minimize} & f & = & -2x_1 - x_2 \\ \text{subject to} & & & 100x_1 + x_2 + y_1 = 1 \\ & & & x_1 \geq 0, \quad x_2 \geq 0 \quad \text{and} \quad y_1 \geq 0. \end{array}$$

- Starting from logical basis $y_1 = 1$:
- Reduced costs are

$$\begin{array}{ll} -2 & \text{for} \quad x_1 \\ -1 & \text{for} \quad x_2 \end{array}$$

- x_1 is best under Dantzig rule but a unit change in x_1 requires a change of -100 in y_1.
- Problem solved in two iterations.


y_1
1
$f = 0$
O
x_2
1
B
$f = -1$
A
0.01
$f = -0.02$
x_1
$x_1 = 1$
$y_1 = -99$
$f = -2$

Example (cont.)

$$\begin{array}{lrcrcrcrcl} \text{minimize} & f & = & -2x_1 & - & x_2 & & & & \\ \text{subject to} & & & 100x_1 & + & x_2 & + & y_1 & = & 1 \\ & & & x_1 \geq 0, & & x_2 \geq 0 & \text{and} & y_1 \geq 0. & & \end{array}$$

- Starting from logical basis $y_1 = 1$:
- Rates of change of objective are

$$\begin{array}{lcll} -2/\sqrt{1+100^2} & \approx & -0.02 & \text{for} \quad x_1 \\ -1/\sqrt{1+1} & \approx & -0.71 & \text{for} \quad x_2 \end{array}$$

Example (cont.)

$$\begin{array}{lrcrcrcrcl} \text{minimize} & f & = & -2x_1 & - & x_2 & & & & \\ \text{subject to} & & & 100x_1 & + & x_2 & + & y_1 & = & 1 \\ & & & x_1 \geq 0, & & x_2 \geq 0 & \text{and} & y_1 \geq 0. & & \end{array}$$

- Starting from logical basis $y_1 = 1$:
- Rates of change of objective are

$$\begin{array}{lcll} -2/\sqrt{1+100^2} & \approx & -0.02 & \text{for} \quad x_1 \\ -1/\sqrt{1+1} & \approx & -0.71 & \text{for} \quad x_2 \end{array}$$

- x_2 is best under **steepest edge** rule.
- Problem solved in *one* iteration.

Edge selection strategies

- 'Most negative reduced cost' rule (Dantzig)

Edge selection strategies

- 'Most negative reduced cost' rule (Dantzig)
- 'Greatest reduction' rule (1951)
  - Too expensive in practice

Edge selection strategies

- 'Most negative reduced cost' rule (Dantzig)
- 'Greatest reduction' rule (1951)
  - Too expensive in practice
- 'Steepest Edge' (1963)
  - Updates exact step lengths
  - Requires extra BTRAN and PRICE each iteration
  - Prohibitively expensive start-up cost unless $B = I$ initially

Edge selection strategies

- 'Most negative reduced cost' rule (Dantzig)
- 'Greatest reduction' rule (1951)
  - Too expensive in practice
- 'Steepest Edge' (1963)
  - Updates exact step lengths
  - Requires extra BTRAN and PRICE each iteration
  - Prohibitively expensive start-up cost unless $B = I$ initially
- 'Devex' (1965)
  - Updates approximate step lengths
  - No additional calculation
  - No start-up cost

Partial PRICE

- Some LP problems naturally have very many more columns than rows

Name	Rows (m)	Columns (n)	Nonzeros (τ)	Columns per row (n/m)	τ/m^2
`fit2d`	25	10500	129018	420	206
`nw04`	36	87482	724148	2430	559

Partial PRICE

- Some LP problems naturally have very many more columns than rows

Name	Rows (m)	Columns (n)	Nonzeros (τ)	Columns per row (n/m)	τ/m^2
`fit2d`	25	10500	129018	420	206
`nw04`	36	87482	724148	2430	559

- PRICE completely dominates solution time

Partial PRICE

- Some LP problems naturally have very many more columns than rows

Name	Rows (m)	Columns (n)	Nonzeros (τ)	Columns per row (n/m)	τ/m^2
`fit2d`	25	10500	129018	420	206
`nw04`	36	87482	724148	2430	559

- PRICE completely dominates solution time
- Simplex method only needs to choose *an* attractive column, not necessarily the best

Partial PRICE

- Some LP problems naturally have very many more columns than rows

Name	Rows (m)	Columns (n)	Nonzeros (τ)	Columns per row (n/m)	τ/m^2
`fit2d`	25	10500	129018	420	206
`nw04`	36	87482	724148	2430	559

- PRICE completely dominates solution time
- Simplex method only needs to choose *an* attractive column, not necessarily the best
- Compute only a small subset of the reduced costs

Partial PRICE

- Some LP problems naturally have very many more columns than rows

Name	Rows (m)	Columns (n)	Nonzeros (τ)	Columns per row (n/m)	τ/m^2
`fit2d`	25	10500	129018	420	206
`nw04`	36	87482	724148	2430	559

- PRICE completely dominates solution time
- Simplex method only needs to choose *an* attractive column, not necessarily the best
- Compute only a small subset of the reduced costs
- Number of iterations may increase but each iteration is vastly faster

Hyper-sparsity

- Hyper-sparsity exists when
  - the pivotal column $\hat{\boldsymbol{a}}_q = B^{-1}\boldsymbol{a}_q$ is sparse; **(FTRAN)**

Hyper-sparsity

- Hyper-sparsity exists when
  - the pivotal column $\hat{\boldsymbol{a}}_q = B^{-1}\boldsymbol{a}_q$ is sparse; **(FTRAN)**
  - the $\boldsymbol{\pi}$ vector $\boldsymbol{\pi}^T = \boldsymbol{e}_p^T B^{-1}$ is sparse; **(BTRAN)**

Hyper-sparsity

- Hyper-sparsity exists when
  - the pivotal column $\hat{\boldsymbol{a}}_q = B^{-1}\boldsymbol{a}_q$ is sparse; **(FTRAN)**
  - the $\boldsymbol{\pi}$ vector $\boldsymbol{\pi}^T = \boldsymbol{e}_p^T B^{-1}$ is sparse; **(BTRAN)**
  - the pivotal row $\hat{\boldsymbol{a}}_p^T = \boldsymbol{\pi}^T N$ is sparse. **(PRICE)**

Hyper-sparsity

- Hyper-sparsity exists when
  - the pivotal column $\hat{\boldsymbol{a}}_q = B^{-1}\boldsymbol{a}_q$ is sparse; **(FTRAN)**
  - the $\boldsymbol{\pi}$ vector $\boldsymbol{\pi}^T = \boldsymbol{e}_p^T B^{-1}$ is sparse; **(BTRAN)**
  - the pivotal row $\hat{\boldsymbol{a}}_p^T = \boldsymbol{\pi}^T N$ is sparse. **(PRICE)**
- Exploit hyper-sparsity both when forming and using these vectors.

See

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, 2005.

Hyper-sparsity in FTRAN

- FTRAN forms $\hat{\boldsymbol{a}}_q = B^{-1}\boldsymbol{a}_q$ using $\boldsymbol{r} = \boldsymbol{a}_q$ in

```
        do 10, k = 1, K_I + K_U
           if (r_{p_k} .eq. 0) go to 10
           r_{p_k} := μ_k r_{p_k}
           r := r − r_{p_k} η_k
   10   continue
```

Hyper-sparsity in FTRAN

- FTRAN forms $\hat{\boldsymbol{a}}_q = B^{-1}\boldsymbol{a}_q$ using $\boldsymbol{r} = \boldsymbol{a}_q$ in

```
      do 10, k = 1, K_I + K_U
         if (r_{p_k} .eq. 0) go to 10
         r_{p_k} := μ_k r_{p_k}
         r := r − r_{p_k} η_k
10    continue
```

- When $\hat{\boldsymbol{a}}_q$ is sparse, most of the work of FTRAN is the test for zero!

Hyper-sparsity in FTRAN

- FTRAN forms $\hat{\boldsymbol{a}}_q = B^{-1}\boldsymbol{a}_q$ using $\boldsymbol{r} = \boldsymbol{a}_q$ in

```
        do 10, k = 1, K_I + K_U
           if (r_{p_k} .eq. 0) go to 10
           r_{p_k} := μ_k r_{p_k}
           r := r − r_{p_k} η_k
10      continue
```

- When $\hat{\boldsymbol{a}}_q$ is sparse, most of the work of FTRAN is the test for zero!

  Remedy:

  Identify the INVERT etas to be applied by

  - knowing etas associated with a nonzero in a particular row Hall and McKinnon (1999)

Hyper-sparsity in FTRAN

- FTRAN forms $\hat{\boldsymbol{a}}_q = B^{-1}\boldsymbol{a}_q$ using $\boldsymbol{r} = \boldsymbol{a}_q$ in

```
        do 10, k = 1, K_I + K_U
           if (r_{p_k} .eq. 0) go to 10
           r_{p_k} := μ_k r_{p_k}
           r := r − r_{p_k} η_k
10      continue
```

- When $\hat{\boldsymbol{a}}_q$ is sparse, most of the work of FTRAN is the test for zero!

  Remedy:

  Identify the INVERT etas to be applied by
  - knowing etas associated with a nonzero in a particular row — Hall and McKinnon (1999)
  - representing etas as a graph and perform a depth-first search — Gilbert and Peierls (1988)

Hyper-sparsity in BTRAN

- BTRAN forms $\boldsymbol{\pi}^T = \boldsymbol{e}_p^T B^{-1}$ using $\boldsymbol{r} = \boldsymbol{e}_p^T$ in

```
        do 10, k = K_I + K_U, 1
           r_{p_k} := μ_k(r_{p_k} − r^T η_k)
10      continue
```

Hyper-sparsity in BTRAN

- BTRAN forms $\boldsymbol{\pi}^T = \boldsymbol{e}_p^T B^{-1}$ using $\boldsymbol{r} = \boldsymbol{e}_p^T$ in

```
        do 10, k = K_I + K_U, 1
           r_{p_k} := μ_k(r_{p_k} − r^T η_k)
10      continue
```

- When $\boldsymbol{\pi}$ is sparse, almost all operations in BTRAN are with zeros

Hyper-sparsity in BTRAN

- BTRAN forms $\boldsymbol{\pi}^T = \boldsymbol{e}_p^T B^{-1}$ using $\boldsymbol{r} = \boldsymbol{e}_p^T$ in

```
        do 10, k = K_I + K_U, 1
           r_{p_k} := μ_k(r_{p_k} − r^T η_k)
10      continue
```

- When $\boldsymbol{\pi}$ is sparse, almost all operations in BTRAN are with zeros

  Remedy:
  - Store INVERT etas row-wise
  - Use special techniques developed for FTRAN

Hyper-sparsity in PRICE

- PRICE forms $\hat{\boldsymbol{a}}_p^T = \boldsymbol{\pi}^T N$
- When $\boldsymbol{\pi}$ is sparse, components of $\hat{\boldsymbol{a}}_p$ are inner products between two sparse vectors

Hyper-sparsity in PRICE

- PRICE forms $\hat{\boldsymbol{a}}_p^T = \boldsymbol{\pi}^T N$
- When $\boldsymbol{\pi}$ is sparse, components of $\hat{\boldsymbol{a}}_p$ are inner products between two sparse vectors
- Most operations are with zeros

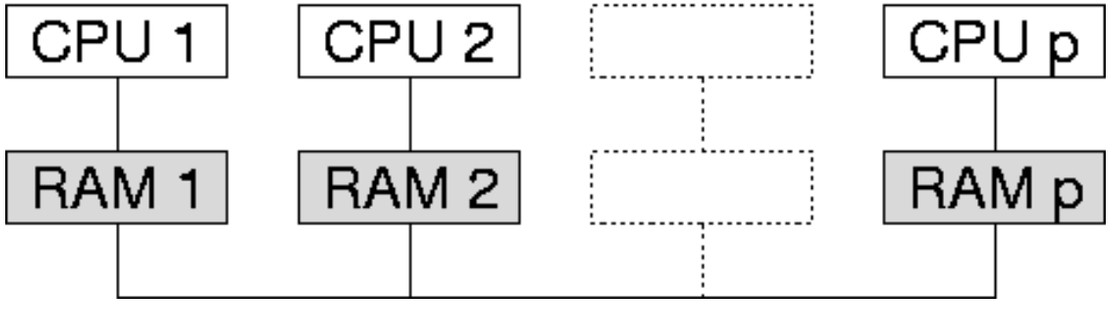
Hyper-sparsity in PRICE

- PRICE forms $\hat{\boldsymbol{a}}_p^T = \boldsymbol{\pi}^T N$
- When $\boldsymbol{\pi}$ is sparse, components of $\hat{\boldsymbol{a}}_p$ are inner products between two sparse vectors
- Most operations are with zeros

  Remedy:
  - Store N row-wise
  - Form $\hat{\boldsymbol{a}}_p^T$ by combining rows of N corresponding to nonzeros in $\boldsymbol{\pi}$.

Parallelising the simplex method

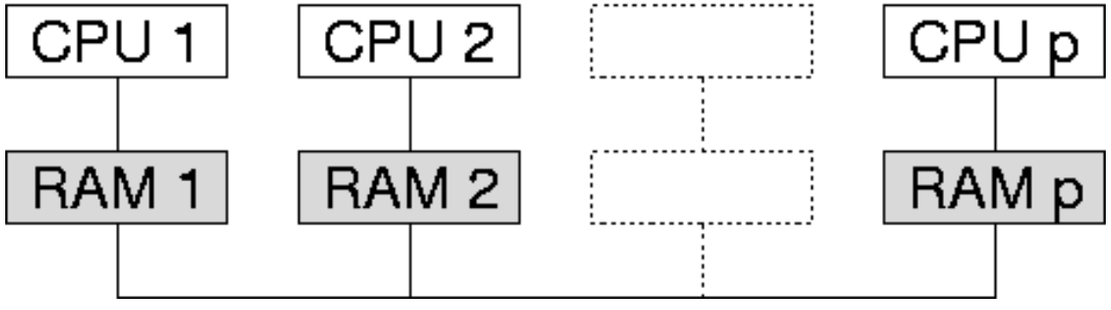
Parallel architectures



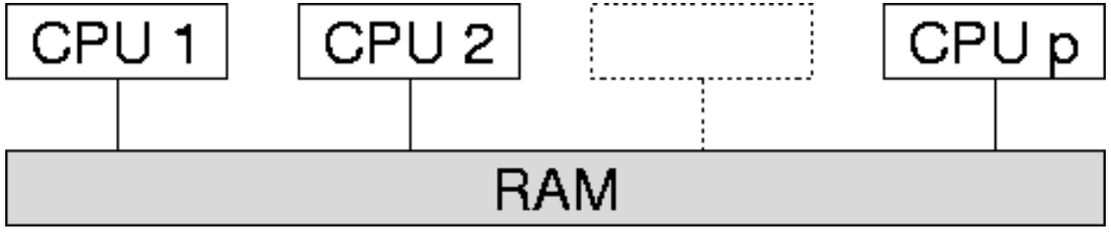
Distributed Memory

Parallelising the simplex method

Parallel architectures



Distributed Memory



Shared Memory

Progress to date

- Standard simplex method parallelises beautifully!
  - Still slower than sequential revised simplex

Progress to date

- Standard simplex method parallelises beautifully!
  - Still slower than sequential revised simplex
- Revised simplex method viewed as sequential
  - Computational components performed in sequence
  - PRICE, CHUZC and CHUZR parallelise

Progress to date

- Standard simplex method parallelises beautifully!
  - Still slower than sequential revised simplex
- Revised simplex method viewed as sequential
  - Computational components performed in sequence
  - PRICE, CHUZC and CHUZR parallelise
  - BTRAN, FTRAN and INVERT are sequential

Progress to date

- Standard simplex method parallelises beautifully!
  - Still slower than sequential revised simplex
- Revised simplex method viewed as sequential
  - Computational components performed in sequence
  - PRICE, CHUZC and CHUZR parallelise
  - BTRAN, FTRAN and INVERT are sequential
- Hall and McKinnon (1995)
  - ParLP: n-processor asynchronous (Dantzig)
  - Speed-up factor: 2.5–4.8

Progress to date

- Standard simplex method parallelises beautifully!
  - Still slower than sequential revised simplex
- Revised simplex method viewed as sequential
  - Computational components performed in sequence
  - PRICE, CHUZC and CHUZR parallelise
  - BTRAN, FTRAN and INVERT are sequential
- Hall and McKinnon (1995)
  - ParLP: n-processor asynchronous (Dantzig)
  - Speed-up factor: 2.5–4.8
- Wunderling (1996)
  - SMoPlex: 2-processor synchronous (Steepest Edge)
  - Speed-up factor: 0.75–1.4

Progress to date

- Standard simplex method parallelises beautifully!
  - Still slower than sequential revised simplex
- Revised simplex method viewed as sequential
  - Computational components performed in sequence
  - PRICE, CHUZC and CHUZR parallelise
  - BTRAN, FTRAN and INVERT are sequential
- Hall and McKinnon (1995)
  - ParLP: n-processor asynchronous (Dantzig)
  - Speed-up factor: 2.5–4.8
- Wunderling (1996)
  - SMoPlex: 2-processor synchronous (Steepest Edge)
  - Speed-up factor: 0.75–1.4
- Hall and McKinnon (1996–97)
  - PARSMI: n-processor asynchronous (Devex)
  - Speed-up factor: 1.7–1.9

Research frontiers

- **Hypersparsity**
  - Column and row selection strategies to promote hypersparsity in the revised simplex method
    - Very challenging

Research frontiers

- **Hypersparsity**
  - Column and row selection strategies to promote hypersparsity in the revised simplex method
    - Very challenging
  - Specialist variants of Gilbert-Peierls for LP

Research frontiers

- **Hypersparsity**
  - Column and row selection strategies to promote hypersparsity in the revised simplex method
    - Very challenging
  - Specialist variants of Gilbert-Peierls for LP
- **Parallel simplex**
  - SYNPLEX: stable synchronous version of PARSMI
    - Implemented but needs improvements and/or parallel INVERT

Research frontiers

- **Hypersparsity**
  - Column and row selection strategies to promote hypersparsity in the revised simplex method
    - Very challenging
  - Specialist variants of Gilbert-Peierls for LP
- **Parallel simplex**
  - SYNPLEX: stable synchronous version of PARSMI
    - Implemented but needs improvements and/or parallel INVERT
  - Parallel Tomlin INVERT
    - Serial simulation for hyper-sparse LPs gives very encouraging results.
    - Parallel implementation due Summer 2007

Research frontiers

- **Hypersparsity**
  - Column and row selection strategies to promote hypersparsity in the revised simplex method
    - Very challenging
  - Specialist variants of Gilbert-Peierls for LP
- **Parallel simplex**
  - SYNPLEX: stable synchronous version of PARSMI
    - Implemented but needs improvements and/or parallel INVERT
  - Parallel Tomlin INVERT
    - Serial simulation for hyper-sparse LPs gives very encouraging results.
    - Parallel implementation due Summer 2007
  - Parallel FTRAN and BTRAN

Research frontiers

- **Hypersparsity**
  - Column and row selection strategies to promote hypersparsity in the revised simplex method
    - Very challenging
  - Specialist variants of Gilbert-Peierls for LP
- **Parallel simplex**
  - SYNPLEX: stable synchronous version of PARSMI
    - Implemented but needs improvements and/or parallel INVERT
  - Parallel Tomlin INVERT
    - Serial simulation for hyper-sparse LPs gives very encouraging results.
    - Parallel implementation due Summer 2007
  - Parallel FTRAN and BTRAN
    - Some scope for standard LPs
    - Very challenging for hyper-sparse LPs!

Conclusions

The simplex method has been around for 60 years

Conclusions

The simplex method has been around for 60 years

- Value of method means that the computational state-of-the-art is very advanced

Conclusions

The simplex method has been around for 60 years

- Value of method means that the computational state-of-the-art is very advanced
- There is still scope for further improvements

Conclusions

The simplex method has been around for 60 years

- Value of method means that the computational state-of-the-art is very advanced
- There is still scope for further improvements
- Making advances is demanding but attracts attention!

Conclusions

The simplex method has been around for 60 years

- Value of method means that the computational state-of-the-art is very advanced
- There is still scope for further improvements
- Making advances is demanding but attracts attention!

Slides available as

`http://www.maths.ed.ac.uk/hall/RealSimplex/`