SYNPLEX
A task-parallel scheme for the revised simplex method

Julian Hall
School of Mathematics
University of Edinburgh
June 23rd 2005
Overview

- The (standard and revised) simplex method for linear programming
Overview

• The (standard and revised) simplex method for linear programming
• Approaches to parallelising the simplex method
Overview

- The (standard and revised) simplex method for linear programming
- Approaches to parallelising the simplex method
- SYNPLEX
Overview

- The (standard and revised) simplex method for linear programming
- Approaches to parallelising the simplex method
- SYNPLEX
- Results and conclusions
Solving LP problems

\[
\begin{align*}
\text{minimize} \quad & f = c^T x \\
\text{subject to} \quad & Ax = b \\
& x \geq 0 \\
\text{where} \quad & x \in \mathbb{R}^n \quad \text{and} \quad b \in \mathbb{R}^m
\end{align*}
\]
Solving LP problems

\[
\begin{align*}
\text{minimize} & \quad f = c^T x \\
\text{subject to} & \quad Ax = b \\
& \quad x \geq 0 \\
\text{where} & \quad x \in \mathbb{R}^n \quad \text{and} \quad b \in \mathbb{R}^m
\end{align*}
\]

- At any vertex the variables may be partitioned into index sets
  - $B$ of $m$ basic variables $x_B \geq 0$
  - $N$ of $n - m$ nonbasic variables $x_N = 0$
Solving LP problems

\[
\begin{align*}
\text{minimize} & \quad f = c^T x \\
\text{subject to} & \quad Ax = b \\
& \quad x \geq 0 \\
\text{where} & \quad x \in \mathbb{R}^n \quad \text{and} \quad b \in \mathbb{R}^m
\end{align*}
\]

- At any vertex the variables may be partitioned into index sets
  - \( B \) of \( m \) basic variables \( x_B \geq 0 \)
  - \( N \) of \( n - m \) nonbasic variables \( x_N = 0 \)
- Components of \( c \) and columns of \( A \) are
  - the basic costs \( c_B \) and basis matrix \( B \)
  - the non-basic costs \( c_N \) and matrix \( N \)
Reduced LP problem

At any vertex the original problem is

\[
\begin{align*}
\text{minimize} & \quad f = c^T_N x_N + c^T_B x_B \\
\text{subject to} & \quad N x_N + B x_B = b \\
& \quad x_N \geq 0 \quad x_B \geq 0
\end{align*}
\]
Reduced LP problem

At any vertex the original problem is

\[
\begin{aligned}
\text{minimize} \quad f &= \mathbf{c}_N^T \mathbf{x}_N + \mathbf{c}_B^T \mathbf{x}_B \\
\text{subject to} \quad \mathbf{N} \mathbf{x}_N + \mathbf{B} \mathbf{x}_B &= \mathbf{b} \\
&\quad \mathbf{x}_N \geq 0 \quad \mathbf{x}_B \geq 0
\end{aligned}
\]

Eliminate \( \mathbf{x}_B \) from the objective to give

\[
\begin{aligned}
\text{minimize} \quad f &= \hat{\mathbf{c}}_N^T \mathbf{x}_N + \hat{f} \\
\text{subject to} \quad \hat{\mathbf{N}} \mathbf{x}_N + \mathbf{I} \mathbf{x}_B &= \hat{\mathbf{b}} \\
&\quad \mathbf{x}_N \geq 0 \quad \mathbf{x}_B \geq 0
\end{aligned}
\]
Reduced LP problem

At any vertex the original problem is

$$\begin{align*}
\text{minimize} \quad f &= c_N^T x_N + c_B^T x_B \\
\text{subject to} \quad N x_N + B x_B &= b \\
\quad x_N &\geq 0 \quad x_B \geq 0
\end{align*}$$

Eliminate $x_B$ from the objective to give

$$\begin{align*}
\text{minimize} \quad f &= \hat{c}_N^T x_N + \hat{f} \\
\text{subject to} \quad \hat{N} x_N + I x_B &= \hat{b} \\
\quad x_N &\geq 0 \quad x_B \geq 0
\end{align*}$$

where $\hat{b} = B^{-1} b$, $\hat{N} = B^{-1} N$, $\hat{f} = c_B^T \hat{b}$ and $\hat{c}_N$ is the vector of reduced costs

$$\hat{c}_N^T = c_N^T - c_B^T \hat{N}$$
The standard simplex method

<table>
<thead>
<tr>
<th></th>
<th>$\mathcal{N}$</th>
<th>$\mathcal{B}$</th>
<th>RHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 : $m$</td>
<td>$\hat{N}$</td>
<td>$I$</td>
<td>$\hat{b}$</td>
</tr>
<tr>
<td>0</td>
<td>$\hat{c}_N^T$</td>
<td>$0^T$</td>
<td>$-\hat{f}$</td>
</tr>
</tbody>
</table>
The standard simplex method

<table>
<thead>
<tr>
<th></th>
<th>$\mathcal{N}$</th>
<th>$B$</th>
<th>RHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\hat{N}$</td>
<td>$I$</td>
<td>$\hat{b}$</td>
</tr>
<tr>
<td>$m$</td>
<td>$\hat{c}_N^T$</td>
<td>$0^T$</td>
<td>$-\hat{f}$</td>
</tr>
</tbody>
</table>

In each iteration:
The standard simplex method

<table>
<thead>
<tr>
<th></th>
<th>$\mathcal{N}$</th>
<th>$\mathcal{B}$</th>
<th>RHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\hat{N}$</td>
<td>$I$</td>
<td>$\hat{b}$</td>
</tr>
<tr>
<td>$m$</td>
<td>$\hat{N}$</td>
<td>$I$</td>
<td>$\hat{b}$</td>
</tr>
<tr>
<td>0</td>
<td>$\hat{c}_N^T$</td>
<td>$0^T$</td>
<td>$-\hat{f}$</td>
</tr>
</tbody>
</table>

In each iteration:

- Select the **pivotal column** $q'$ of a nonbasic variable $q \in \mathcal{N}$ to be increased from zero
The standard simplex method

<table>
<thead>
<tr>
<th></th>
<th>( \mathcal{N} )</th>
<th>( B )</th>
<th>RHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( \hat{N} )</td>
<td></td>
<td>( \hat{b} )</td>
</tr>
<tr>
<td>m</td>
<td>( \hat{N} )</td>
<td>( I )</td>
<td>( \hat{b} )</td>
</tr>
<tr>
<td>0</td>
<td>( \hat{c}_N^T )</td>
<td>( 0^T )</td>
<td>( -\hat{f} )</td>
</tr>
</tbody>
</table>

In each iteration:

- Select the **pivotal column** \( q' \) of a nonbasic variable \( q \in \mathcal{N} \) to be increased from zero
- Find the **pivotal row** \( p \) of the first basic variable \( p' \in B \) to be zeroed
The standard simplex method

<table>
<thead>
<tr>
<th>$\mathcal{N}$</th>
<th>$\mathcal{B}$</th>
<th>RHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1$</td>
<td>$\hat{N}$</td>
<td>$\hat{b}$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$I$</td>
<td></td>
</tr>
<tr>
<td>$m$</td>
<td>$\hat{c}_N^T$</td>
<td>$0^T$</td>
</tr>
</tbody>
</table>

In each iteration:

- Select the **pivotal column** $q'$ of a nonbasic variable $q \in \mathcal{N}$ to be increased from zero
- Find the **pivotal row** $p$ of the first basic variable $p' \in \mathcal{B}$ to be zeroed
- Exchange indices $p'$ and $q$ between sets $\mathcal{B}$ and $\mathcal{N}$
- Update tableau corresponding to this **basis change**
The standard simplex method (cont.)

Advantages:

- Easy to understand
- Simple to implement
The standard simplex method (cont.)

Advantages:

• Easy to understand
• Simple to implement

Disadvantages:

• Expensive: the matrix $\hat{N}$ ‘usually’ treated as full
  • Storage requirement: $O(mn)$ memory locations
  • Computation requirement: $O(mn)$ floating point operations per iteration
• Numerically unstable
Revised simplex method

Given \( \hat{c}_N, \hat{b} \) and a representation of \( B^{-1} \), repeat

CHUZC: Scan the reduced costs \( \hat{c}_N \) for a good candidate \( q \) to enter the basis
Revised simplex method

Given $\hat{c}_N$, $\hat{b}$ and a representation of $B^{-1}$, repeat

CHUZC: Scan the reduced costs $\hat{c}_N$ for a good candidate $q$ to enter the basis

FTRAN: Form the pivotal column $\hat{a}_q = B^{-1}a_q$, where $a_q$ is column $q$ of $A$
Revised simplex method

Given $\hat{c}_N$, $\hat{b}$ and a representation of $B^{-1}$, repeat

CHUZC: Scan the reduced costs $\hat{c}_N$ for a good candidate $q$ to enter the basis
FTRAN: Form the pivotal column $\hat{a}_q = B^{-1}a_q$, where $a_q$ is column $q$ of $A$
CHUZR: Scan the ratios $\hat{b}_i/\hat{a}_{iq}$ for the row $p$ of a good candidate to leave the basis
Revised simplex method

Given $\hat{c}_N$, $\hat{b}$ and a representation of $B^{-1}$, repeat

CHUZC: Scan the reduced costs $\hat{c}_N$ for a good candidate $q$ to enter the basis
FTRAN: Form the pivotal column $\hat{a}_q = B^{-1}a_q$, where $a_q$ is column $q$ of $A$
CHUZR: Scan the ratios $\hat{b}_i/\hat{a}_{iq}$ for the row $p$ of a good candidate to leave the basis
   Let $\alpha = \hat{b}_p/\hat{a}_{pq}$
   Update $\hat{b} := \hat{b} - \alpha \hat{a}_q$
Revised simplex method

Given \( \hat{c}_N, \hat{b} \) and a representation of \( B^{-1} \), repeat

**CHUZC:** Scan the reduced costs \( \hat{c}_N \) for a good candidate \( q \) to enter the basis

**FTRAN:** Form the pivotal column \( \hat{a}_q = B^{-1}a_q \), where \( a_q \) is column \( q \) of \( A \)

**CHUZR:** Scan the ratios \( \hat{b}_i/\hat{a}_{iq} \) for the row \( p \) of a good candidate to leave the basis
   
   Let \( \alpha = \hat{b}_p/\hat{a}_{pq} \)
   Update \( \hat{b} := \hat{b} - \alpha \hat{a}_q \)

**BTRAN:** Form \( \pi^T = e_p^T B^{-1} \)
Revised simplex method

Given $\hat{c}_N$, $\hat{b}$ and a representation of $B^{-1}$, repeat

CHUZC: Scan the reduced costs $\hat{c}_N$ for a good candidate $q$ to enter the basis
FTRAN: Form the pivotal column $\hat{a}_q = B^{-1}a_q$, where $a_q$ is column $q$ of $A$
CHUZR: Scan the ratios $\hat{b}_i/\hat{a}_{iq}$ for the row $p$ of a good candidate to leave the basis
  Let $\alpha = \hat{b}_p/\hat{a}_{pq}$
  Update $\hat{b} := \hat{b} - \alpha \hat{a}_q$
BTRAN: Form $\pi^T = e_p^T B^{-1}$
PRICE: Form the pivotal row $\hat{a}_p^T = \pi^T N$
Revised simplex method

Given $\hat{c}_N$, $\hat{b}$ and a representation of $B^{-1}$, repeat

CHUZC: Scan the reduced costs $\hat{c}_N$ for a good candidate $q$ to enter the basis

FTRAN: Form the pivotal column $\hat{a}_q = B^{-1}a_q$, where $a_q$ is column $q$ of $A$

CHUZR: Scan the ratios $\hat{b}_i/\hat{a}_{iq}$ for the row $p$ of a good candidate to leave the basis
    Let $\alpha = \hat{b}_p/\hat{a}_{pq}$
    Update $\hat{b} := \hat{b} - \alpha \hat{a}_q$

BTRAN: Form $\pi^T = e_p^T B^{-1}$

PRICE: Form the pivotal row $\hat{a}_p^T = \pi^T N$
    Update reduced costs $\hat{c}_N^T := \hat{c}_N^T - \hat{c}_q \hat{a}_p^T$
Revised simplex method

Given $\hat{c}_N$, $\hat{b}$ and a representation of $B^{-1}$, repeat

CHUZC: Scan the reduced costs $\hat{c}_N$ for a good candidate $q$ to enter the basis
FTRAN: Form the pivotal column $\hat{a}_q = B^{-1}a_q$, where $a_q$ is column $q$ of $A$
CHUZR: Scan the ratios $\hat{b}_i/\hat{a}_{iq}$ for the row $p$ of a good candidate to leave the basis
   Let $\alpha = \hat{b}_p/\hat{a}_{pq}$
   Update $\hat{b} := \hat{b} - \alpha \hat{a}_q$
BTRAN: Form $\pi^T = e_p^T B^{-1}$
PRICE: Form the pivotal row $\hat{a}_p^T = \pi^T N$
   Update reduced costs $\hat{c}_N^T := \hat{c}_N^T - \hat{c}_q \hat{a}_p^T$
If (growth in factors) then
   INVERT: Form a representation of $B^{-1}$
else
   UPDATE: Update the representation of $B^{-1}$ corresponding to the basis change
end if
Factored representation of $B^{-1}$

- Each iteration, $a_q$ replaces column $p$ of $B$
Factored representation of $B^{-1}$

- Each iteration, $a_q$ replaces column $p$ of $B$

$$B := B + (a_q - a_p)e_p^T$$
Factored representation of $B^{-1}$

- Each iteration, $a_q$ replaces column $p$ of $B$

$$B := B + (a_q - a_p)e_p^T \quad \Rightarrow \quad B^{-1} := \left( I - \frac{(\hat{a}_q - e_p)e_p^T}{\hat{a}_{pq}} \right) B^{-1}$$
Factored representation of $B^{-1}$

- Each iteration, $a_q$ replaces column $p$ of $B$

$$B := B + (a_q - a_p)e_p^T \quad \Rightarrow \quad B^{-1} := \left( I - \frac{(\hat{a}_q - e_p)e_p^T}{\hat{a}_{pq}} \right) B^{-1}$$

- When using the product form update $B^{-1} = E_U^{-1} B_0^{-1}$
Factored representation of $B^{-1}$

- Each iteration, $a_q$ replaces column $p$ of $B$

\[ B := B + (a_q - a_p)e_p^T \quad \Rightarrow \quad B^{-1} := \left( I - \frac{(\hat{a}_q - e_p)e_p^T}{\hat{a}_{pq}} \right) B^{-1} \]

- When using the product form update $B^{-1} = E_U^{-1}B_0^{-1}$
  - $B_0^{-1}$ is represented by the INVERT etas
  - $E_U^{-1}$ is represented by the UPDATE etas
Factored representation of $B^{-1}$

- Each iteration, $a_q$ replaces column $p$ of $B$

$$B := B + (a_q - a_p)e_p^T \quad \Rightarrow \quad B^{-1} := \left(I - \frac{(\hat{a}_q - e_p)e_p^T}{\hat{a}_{pq}}\right)B^{-1}$$

- When using the product form update $B^{-1} = E_U^{-1}B_0^{-1}$
  - $B_0^{-1}$ is represented by the INVERT etas
  - $E_U^{-1}$ is represented by the UPDATE etas
  - FTRAN ($\hat{a}_q = B^{-1}a_q$) is performed as

$$\tilde{a}_q = B_0^{-1}a_q \quad \text{and} \quad \hat{a}_q = E_U^{-1}\tilde{a}_q$$
Factored representation of $B^{-1}$

- Each iteration, $a_q$ replaces column $p$ of $B$

\[
B := B + (a_q - a_p)e_p^T \Rightarrow B^{-1} := \left( I - \frac{(\hat{a}_q - e_p)e_p^T}{\hat{a}_{pq}} \right)B^{-1}
\]

- When using the product form update $B^{-1} = E^{-1}_U B^{-1}_0$
  - $B^{-1}_0$ is represented by the INVERT etas
  - $E^{-1}_U$ is represented by the UPDATE etas
  - FTRAN ($\hat{a}_q = B^{-1}_0 a_q$) is performed as
    \[
    \tilde{a}_q = B^{-1}_0 a_q \quad \text{and} \quad \hat{a}_q = E^{-1}_U \tilde{a}_q
    \]
  - BTRAN ($\pi^T = e_p^T B^{-1}$) is performed as
    \[
    \tilde{\pi}^T = e_p^T E^{-1}_U \quad \text{and} \quad \pi^T = \tilde{\pi}^T B^{-1}_0
    \]
Revised simplex method with multiple pricing

CHUZC: Scan $\hat{c}_N$ for a set $Q$ of good candidates to enter the basis
Revised simplex method with multiple pricing

CHUZC: Scan $\hat{c}_N$ for a set $Q$ of good candidates to enter the basis
FTRAN: Form $\hat{a}_j = B^{-1}a_j, \forall j \in Q$, where $a_j$ is column $j$ of $A$
Revised simplex method with multiple pricing

CHUZC: Scan \( \hat{c}_N \) for a set \( Q \) of good candidates to enter the basis

FTRAN: Form \( \hat{a}_j = B^{-1}a_j, \forall j \in Q \), where \( a_j \) is column \( j \) of \( A \)

Loop \{minor iterations\}

CHUZC_MI: Scan \( \hat{c}_Q \) for a good candidate \( q \) to enter the basis

CHUZR: Scan the ratios \( \hat{b}_i/\hat{a}_{iq} \) for the row \( p \) of a good candidate to leave the basis

UPDATE_MI: Update \( Q := Q\{q\}; \hat{b} := \hat{b} - \alpha \hat{a}_q; \hat{a}_j \) and \( \hat{c}_j, \forall j \in Q \)

End loop \{minor iterations\}
Revised simplex method with multiple pricing

CHUZC: Scan $\hat{c}_N$ for a set $Q$ of good candidates to enter the basis

FTRAN: Form $\hat{a}_j = B^{-1}a_j$, $\forall j \in Q$, where $a_j$ is column $j$ of $A$

Loop \{minor iterations\}

CHUZC_MI: Scan $\hat{c}_Q$ for a good candidate $q$ to enter the basis

CHUZR: Scan the ratios $\hat{b}_i/\hat{a}_{iq}$ for the row $p$ of a good candidate to leave the basis

UPDATE_MI: Update $Q := Q\{q\}$; $\hat{b} := \hat{b} - \alpha\hat{a}_q$; $\hat{a}_j$ and $\hat{c}_j$, $\forall j \in Q$

End loop \{minor iterations\}

For \{each basis change\} do

BTRAN: Form $\pi^T = e_p^T B^{-1}$

PRICE: Form pivotal row $\hat{a}_p^T = \pi^T N$ and update $\hat{c}_N := \hat{c}_N - \hat{c}_q \hat{a}_p^T$

If \{growth in factors\} then

INVERT: Form a new representation of $B^{-1}$

else

UPDATE: Update the representation of $B^{-1}$ corresponding to the basis change

end if

End do

SYNPLEX, a task-parallel scheme for the revised simplex method
Revised simplex method with multiple pricing

Disadvantages:

- Column selected in second and subsequent minor iteration is not the best
  Number of iterations required to solve the LP may increase
Revised simplex method with multiple pricing

Disadvantages:

- Column selected in second and subsequent minor iteration is not the best
  Number of iterations required to solve the LP may increase
- Some columns in $Q$ may become unattractive during minor iterations
  Work of some FTRANs may be wasted
Revised simplex method with multiple pricing

Disadvantages:

- Column selected in second and subsequent minor iteration is not the best
  Number of iterations required to solve the LP may increase
- Some columns in $\mathbf{Q}$ may become unattractive during minor iterations
  Work of some FTRANs may be wasted

Advantages:

- Offers scope for task parallelism
Parallelising the simplex method

Why?
Parallelising the simplex method

Why?

- Never been done
Parallelising the simplex method

Why?

- Never been done
- Simplex method (still) very widely used
Parallelising the simplex method

Why?

- Never been done
- Simplex method (still) very widely used
- Enables significantly larger problems to be solved
Parallelising the simplex method

Why?

- Never been done
- Simplex method (still) very widely used
- Enables significantly larger problems to be solved

How?
Parallelising the simplex method

Why?

- Never been done
- Simplex method (still) very widely used
- Enables significantly larger problems to be solved

How?

- Exploit data parallelism
  Use several processors simultaneously to perform a single operation
Parallelising the simplex method

Why?

- Never been done
- Simplex method (still) very widely used
- Enables significantly larger problems to be solved

How?

- Exploit **data parallelism**
  Use several processors simultaneously to perform a single operation
- Exploit **task parallelism**
  Perform more than one operation simultaneously using several processors
**Parallelising simplex computational components**

<table>
<thead>
<tr>
<th>Component</th>
<th>Properties</th>
<th>Scope for data parallelism</th>
</tr>
</thead>
</table>
Parallelising simplex computational components

<table>
<thead>
<tr>
<th>Component</th>
<th>Properties</th>
<th>Scope for data parallelism</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHUZC</td>
<td>Pass through a vector</td>
<td>Immediate</td>
</tr>
</tbody>
</table>
## Parallelising simplex computational components

<table>
<thead>
<tr>
<th>Component</th>
<th>Properties</th>
<th>Scope for data parallelism</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHUZC</td>
<td>Pass through a vector</td>
<td>Immediate</td>
</tr>
<tr>
<td>FTRAN</td>
<td>INVERT etas are short: some may be applied independently UPDATE etas are long(er): may be applied as a matrix vector product</td>
<td>Little Immediate</td>
</tr>
</tbody>
</table>
### Parallelising simplex computational components

<table>
<thead>
<tr>
<th>Component</th>
<th>Properties</th>
<th>Scope for data parallelism</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHUZC</td>
<td>Pass through a vector</td>
<td>Immediate</td>
</tr>
</tbody>
</table>
| FTRAN     | INVERT etas are short: some may be applied independently  
UPDATE etas are long(er): may be applied as a matrix vector product | Little  
Immediate |
| UPDATE_MI | Dense Gauss-Jordan elimination | Immediate |
## Parallelising simplex computational components

<table>
<thead>
<tr>
<th>Component</th>
<th>Properties</th>
<th>Scope for data parallelism</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHUZC</td>
<td>Pass through a vector</td>
<td>Immediate</td>
</tr>
<tr>
<td>FTRAN</td>
<td>INVERT etas are short: some may be applied independently</td>
<td>Little</td>
</tr>
<tr>
<td></td>
<td>UPDATE etas are long(er): may be applied as a matrix vector product</td>
<td>Immediate</td>
</tr>
<tr>
<td>UPDATE_MI</td>
<td>Dense Gauss-Jordan elimination</td>
<td>Immediate</td>
</tr>
<tr>
<td>CHUZR</td>
<td>Pass through a vector</td>
<td>Immediate</td>
</tr>
</tbody>
</table>
### Parallelising simplex computational components

<table>
<thead>
<tr>
<th>Component</th>
<th>Properties</th>
<th>Scope for data parallelism</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHUZC</td>
<td>Pass through a vector</td>
<td>Immediate</td>
</tr>
<tr>
<td>FTRAN</td>
<td>INVERT etas are short: some may be applied independently</td>
<td>Little</td>
</tr>
<tr>
<td></td>
<td>UPDATE etas are long(er): may be applied as a matrix vector product</td>
<td>Immediate</td>
</tr>
<tr>
<td>UPDATE_MI</td>
<td>Dense Gauss-Jordan elimination</td>
<td>Immediate</td>
</tr>
<tr>
<td>CHUZR</td>
<td>Pass through a vector</td>
<td>Immediate</td>
</tr>
<tr>
<td>BTRAN</td>
<td>UPDATE etas: negligible computation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INVERT etas: (as <strong>FTRAN</strong>)</td>
<td>Little</td>
</tr>
</tbody>
</table>
# Parallelising simplex computational components

<table>
<thead>
<tr>
<th>Component</th>
<th>Properties</th>
<th>Scope for data parallelism</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHUZC</td>
<td>Pass through a vector</td>
<td>Immediate</td>
</tr>
<tr>
<td>FTRAN</td>
<td>INVERT etas are short: some may be applied independently</td>
<td>Little</td>
</tr>
<tr>
<td></td>
<td>UPDATE etas are long(er): may be applied as a matrix vector product</td>
<td>Immediate</td>
</tr>
<tr>
<td>UPDATE_MI</td>
<td>Dense Gauss-Jordan elimination</td>
<td>Immediate</td>
</tr>
<tr>
<td>CHUZR</td>
<td>Pass through a vector</td>
<td>Immediate</td>
</tr>
<tr>
<td>BTRAN</td>
<td>UPDATE etas: negligible computation</td>
<td>Immediate</td>
</tr>
<tr>
<td></td>
<td>INVERT etas: (as FTRAN)</td>
<td>Little</td>
</tr>
<tr>
<td>PRICE</td>
<td>Matrix vector product</td>
<td>Immediate</td>
</tr>
</tbody>
</table>
**Parallelising simplex computational components**

<table>
<thead>
<tr>
<th>Component</th>
<th>Properties</th>
<th>Scope for data parallelism</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHUZC</td>
<td>Pass through a vector</td>
<td>Immediate</td>
</tr>
<tr>
<td>FTRAN</td>
<td>INVERT etas are short: some may be applied independently UPDATE etas are long(er): may be applied as a matrix vector product</td>
<td>Little Immediate</td>
</tr>
<tr>
<td>UPDATE_MI</td>
<td>Dense Gauss-Jordan elimination</td>
<td>Immediate</td>
</tr>
<tr>
<td>CHUZR</td>
<td>Pass through a vector</td>
<td>Immediate</td>
</tr>
<tr>
<td>BTRAN</td>
<td>UPDATE etas: negligible computation INVERT etas: (as FTRAN)</td>
<td>Little</td>
</tr>
<tr>
<td>PRICE</td>
<td>Matrix vector product</td>
<td>Immediate</td>
</tr>
<tr>
<td>INVERT</td>
<td>Searches through $B_0$ and (half-)FTRANs</td>
<td>Little (traditionally)</td>
</tr>
</tbody>
</table>

SYNPLEX, a task-parallel scheme for the revised simplex method
Past approaches

Standard simplex method

• Good parallel efficiency achieved
Past approaches

Standard simplex method

- Good parallel efficiency achieved... many times!
Past approaches

Standard simplex method

- Good parallel efficiency achieved... many times!
- Totally uncompetitive with serial RSM without a prohibitively large number of processors
Past approaches

Standard simplex method

- Good parallel efficiency achieved... many times!
- Totally uncompetitive with serial RSM without a prohibitively large number of processors

Data parallel revised simplex method

- Only the immediate parallelism in PRICE has been exploited
Past approaches

Standard simplex method

- Good parallel efficiency achieved... many times!
- Totally uncompetitive with serial RSM without a prohibitively large number of processors

Data parallel revised simplex method

- Only the immediate parallelism in PRICE has been exploited
- Significant speed-up only obtained when $n \gg m$ so PRICE dominates
  For such problems an efficient serial solver uses partial pricing so PRICE no longer dominates
Data/task parallel revised simplex method (with multiple pricing)

Wunderling (1996)

- Parallel (except for INVERT) for only two processors
- Good results only for problems when $n \gg m$
Data/task parallel revised simplex method  
(with multiple pricing)

Wunderling (1996)

- Parallel (except for INVERT) for only two processors
- Good results only for problems when $n \gg m$

ASYNPLEX: Hall and McKinnon (1995)

- Fully task parallel (inefficient) variant of the revised simplex method
- Speed-up (on Cray T3D) of up to 5 on modest Netlib problems
Data/task parallel revised simplex method
(with multiple pricing)

Wunderling (1996)

- Parallel (except for INVERT) for only two processors
- Good results only for problems when $n \gg m$

ASYNPLEX: Hall and McKinnon (1995)

- Fully task parallel (inefficient) variant of the revised simplex method
- Speed-up (on Cray T3D) of up to 5 on modest Netlib problems

PARSMI: Hall and McKinnon (1996)

- Fully task parallel revised simplex method with multiple pricing
- Speed-up (on Cray T3D) of between 1.7 and 1.9 on modest Netlib problems

SYNPLEX, a task-parallel scheme for the revised simplex method
(Some) ASYNPLEX and PARSMI limitations

- Asynchronous—so very hard to implement
(Some) ASYNPLEX and PARSMI limitations

- Asynchronous—so very hard to implement
- Numerically unstable—due to overlapping INVERT with basis changes
(Some) ASYNPLEX and PARSMI limitations

- Asynchronous—so very hard to implement
- Numerically unstable—due to overlapping INVERT with basis changes
- Reduced costs always out-of-date—more iterations and wasted FTRANs
(Some) ASYNPLEX and PARSMI limitations

- Asynchronous—so very hard to implement
- Numerically unstable—due to overlapping INVERT with basis changes
- Reduced costs always out-of-date—more iterations and wasted FTRANs
- Significant communication overhead