# Parallelisation of the revised simplex method for general large scale LP problems

Julian Hall

School of Mathematics

University of Edinburgh

18th October 2004

http://www.maths.ed.ac.uk/hall/MSO2004/

Work started at CERFACS: Autumn 2003

# Overview

- The simplex method for linear programming
  - The standard simplex method
  - The revised simplex method
- Approaches to parallelising the simplex method
- SYNPLEX
- Implementation
- Results and conclusions

# Solving LP problems

$$
\begin{array}{rl}
\text{minimize} & f = \boldsymbol{c}^T \boldsymbol{x} \\
\text{subject to} & A\boldsymbol{x} = \boldsymbol{b} \\
 & \boldsymbol{x} \geq \boldsymbol{0} \\
\text{where} & \boldsymbol{x} \in \mathbb{R}^n \quad \text{and} \quad \boldsymbol{b} \in \mathbb{R}^m.
\end{array}
$$

- At any vertex the variables may be partitioned into index sets
  - $\mathcal{B}$ of $m$ basic variables $\boldsymbol{x}_B \geq \boldsymbol{0}$:
  - $\mathcal{N}$ of $n - m$ nonbasic variables $\boldsymbol{x}_N = \boldsymbol{0}$.
- Components of $\boldsymbol{c}$ and columns of $A$ are
  - the basic costs $\boldsymbol{c}_B$ and basis matrix $B$;
  - the non-basic costs $\boldsymbol{c}_N$ and matrix $N$.

# Reduced LP problem

At any vertex the original problem is

$$\begin{array}{lrcrcl} \text{minimize} & f & = & \boldsymbol{c}_N^T\boldsymbol{x}_N & + & \boldsymbol{c}_B^T\boldsymbol{x}_B \\ \text{subject to} & & & N\,\boldsymbol{x}_N & + & B\,\boldsymbol{x}_B \; = \; \boldsymbol{b} \\ & & & & & \boldsymbol{x}_N \ge \mathbf{0} \quad \boldsymbol{x}_B \ge \mathbf{0}. \end{array}$$

Eliminate $\boldsymbol{x}_B$ from the objective to give

$$\begin{array}{lrcrcl} \text{minimize} & f & = & \hat{\boldsymbol{c}}_N^T\boldsymbol{x}_N & + & \hat{f} \\ \text{subject to} & & & \hat{N}\,\boldsymbol{x}_N & + & I\,\boldsymbol{x}_B \; = \; \hat{\boldsymbol{b}} \\ & & & & & \boldsymbol{x}_N \ge \mathbf{0} \quad \boldsymbol{x}_B \ge \mathbf{0}, \end{array}$$

where $\hat{\boldsymbol{b}} = B^{-1}\boldsymbol{b}$, $\hat{N} = B^{-1}N$ and $\hat{\boldsymbol{c}}_N$ is the vector of **reduced costs** given by

$$\hat{\boldsymbol{c}}_N^T = \boldsymbol{c}_N^T - \boldsymbol{c}_B^T\hat{N}.$$

# The standard simplex method

| | $\mathcal{N}$ | $\mathcal{B}$ | RHS |
|---|---|---|---|
| $1$ $\vdots$ $m$ | $\hat{N}$ | $I$ | $\hat{\boldsymbol{b}}$ |
| $0$ | $\hat{\boldsymbol{c}}_N^T$ | $\mathbf{0}^T$ | $-\hat{f}$ |

**In each iteration:**

- Select the column $q'$ of a nonbasic variable $q \in \mathcal{N}$ to be increased from zero.
- Find the 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**.
  - Scale row $p$ by $\hat{N}_{pq'}$.
  - Eliminate $\hat{N}_{iq'}$, $i = 0, \ldots, m, i \neq p$.

# Revised simplex method

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

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

`FTRAN:` Form the pivotal column $\hat{\boldsymbol{a}}_q = B^{-1}\boldsymbol{a}_q$, where $\boldsymbol{a}_q$ 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 $\hat{\boldsymbol{b}} := \hat{\boldsymbol{b}} - \alpha\hat{\boldsymbol{a}}_q$.

`BTRAN:` Form $\boldsymbol{\pi}_p^T = \boldsymbol{e}_p^T B^{-1}$.

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

`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 $B^{-1}$

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 is performed as

$$\tilde{\boldsymbol{a}}_q = B_0^{-1} \boldsymbol{a}_q \quad \text{and} \quad \hat{\boldsymbol{a}}_q = E_U^{-1} \tilde{\boldsymbol{a}}_q.$$

- BTRAN is performed as

$$\tilde{\boldsymbol{\pi}}_p{}^T = \boldsymbol{e}_p^T E_U^{-1} \quad \text{and} \quad \boldsymbol{\pi}_p^T = \tilde{\boldsymbol{\pi}}_p{}^T B_0^{-1}.$$

# Parallelising the simplex method

- Why try to parallelise the simplex method?
  - Want to solve general large sparse LP problems fast
  - Serial revised simplex solvers are very efficient
    CPLEX, OSL, Xpress, SOPLEX, EMSOL
  - Parallel solver must offer (hope of) further performance improvement
- How to parallelise the simplex method
  - Exploit **data parallelism**
    Use several processors simultaneously to perform a single operation
  - Exploit **task parallelism**
    Perform more than one operation simultaneously using several processors
- Architectures
  - Distributed memory machines
    Communication between processors using message-passing routines (PVM/MPI)
  - Shared memory machines
    Coordination of processes and memory read/write using message-passing routines or compiler directives (PVM/MPI/OpenMP)

# The standard simplex method

| | $\mathcal{N}$ | $\mathcal{B}$ | RHS |
|---|---|---|---|
| $1$<br>$\vdots$<br>$m$ | $\hat{N}$ | $I$ | $\hat{\boldsymbol{b}}$ |
| $0$ | $\hat{\boldsymbol{c}}_N^T$ | $\mathbf{0}^T$ | $-\hat{f}$ |

**In each iteration:**

- Select the column $q'$ of a nonbasic variable $q \in \mathcal{N}$ to be increased from zero.
- Find the 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**.
  - Scale row $p$ by $\hat{N}_{pq'}$.
  - Eliminate $\hat{N}_{iq'}$, $i = 0, \ldots, m, i \neq p$.

# Revised simplex method

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

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

`FTRAN:` Form the pivotal column $\hat{\boldsymbol{a}}_q = B^{-1}\boldsymbol{a}_q$, where $\boldsymbol{a}_q$ 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 $\hat{\boldsymbol{b}} := \hat{\boldsymbol{b}} - \alpha\hat{\boldsymbol{a}}_q$.

`BTRAN:` Form $\boldsymbol{\pi}_p^T = \boldsymbol{e}_p^T B^{-1}$.

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

`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`

# Revised simplex method with multiple pricing

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

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

`Loop` {minor iterations}

> `CHUZC_MI:` Scan $\hat{\boldsymbol{c}}_Q$ 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, where $\hat{\boldsymbol{b}} = B^{-1}\boldsymbol{b}$. Let $\alpha = \hat{b}_p/\hat{a}_{pq}$.
>
> `UPDATE_MI:` Update $\mathcal{Q} := \mathcal{Q}\backslash\{q\}$, $\hat{\boldsymbol{b}} := \hat{\boldsymbol{b}} - \alpha\hat{\boldsymbol{a}}_q$, $\hat{\boldsymbol{a}}_Q$ and reduced costs $\hat{\boldsymbol{c}}_Q$.

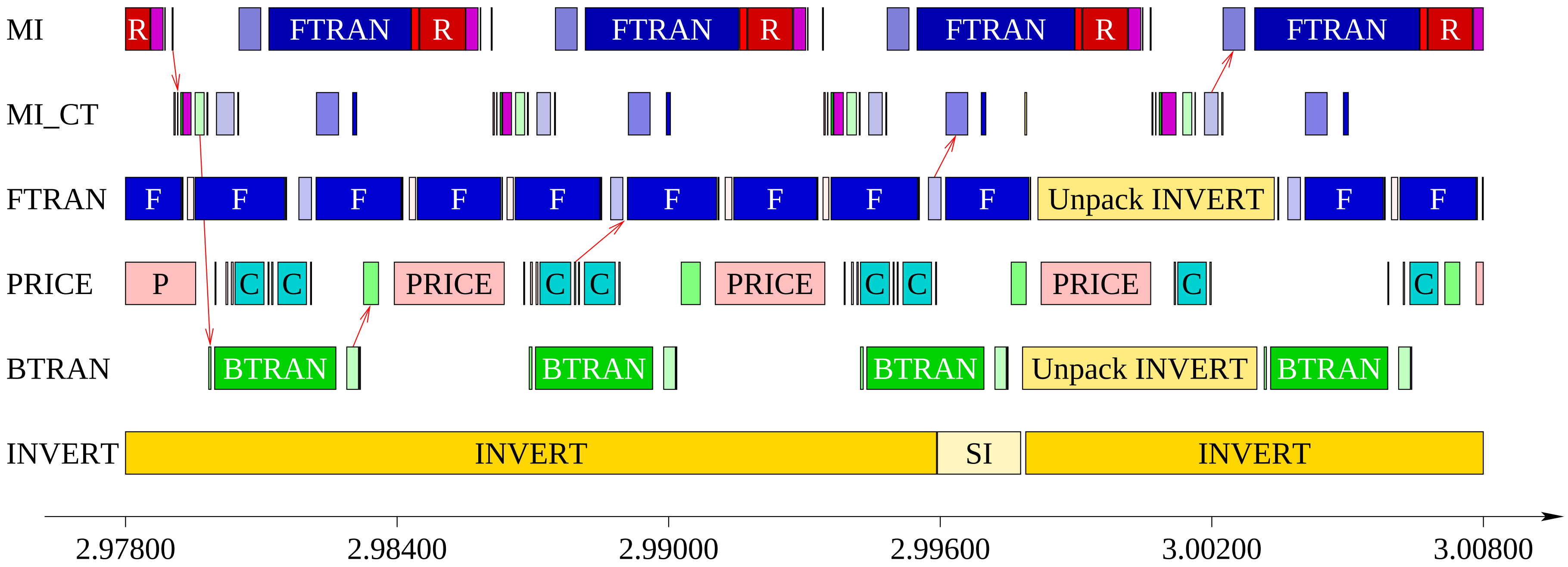
`End loop` {minor iterations}

`For` {each basis change} `do`

> `BTRAN:` Form $\boldsymbol{\pi}_p^T = \boldsymbol{e}_p^T B^{-1}$.
>
> `PRICE:` Form pivotal row $\hat{\boldsymbol{a}}_p^T = \boldsymbol{\pi}_p^T N$.
>
> Update reduced costs $\hat{\boldsymbol{c}}_N := \hat{\boldsymbol{c}}_N - \hat{c}_q\hat{\boldsymbol{a}}_p^T$.
>
> `If` {growth in factors} `then`
>
> > `INVERT:` Form a new representation of $B^{-1}$.
>
> `else`
>
> > `UPDATE:` Update representation of $B^{-1}$ corresponding to the basis change.
>
> `end if`

`End do`

# PARSMI: Hall and McKinnon (1996)




Drawbacks

- Asynchronous so very hard to implement—on Cray T3D using PVM and `shmem`
- Overlapping INVERT with basis changes leads to numerical instability
- Reduced costs always out-of-date so poorer candidate persistence
- Communication overhead still significant

# SYNPLEX

- Synchronous variant of PARSMI
- Better candidate persistency
- INVERT not overlapped with basis changes so numerically stable
- All tasks distributed over all processors
- Target platform: Sun Fire E15k (52 x 0.9GHz UltraSPARC III)
- Parallelise using OpenMP

# SYNPLEX: parallel scheme

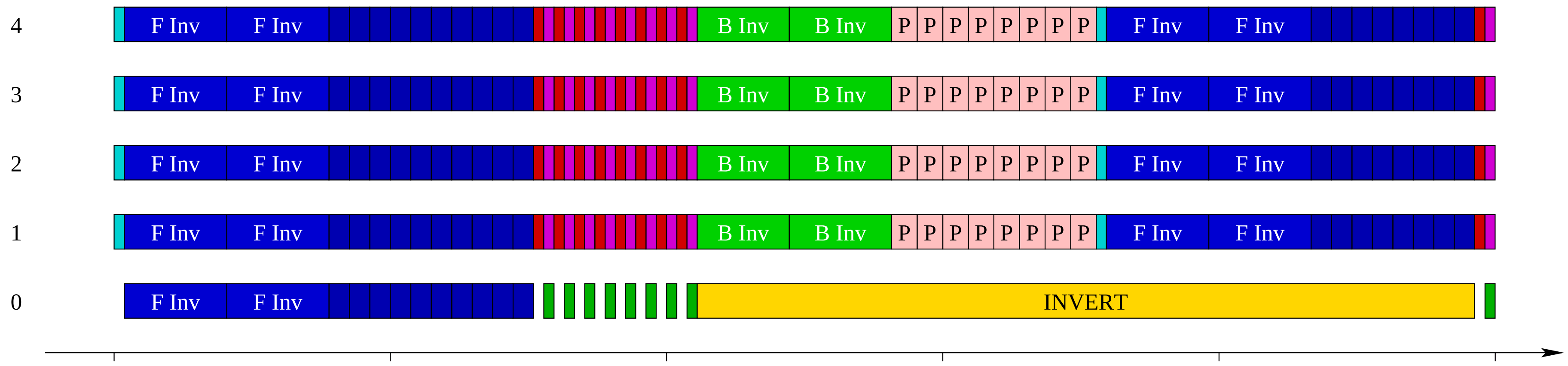
When using $1 + p$ processors

- Task parallel multiple FTRAN for INVERT etas
- Rows distributed over $p$ processors for data parallel
  - multiple FTRAN for UPDATE etas
  - CHUZR
  - UPDATE RHS
- BTRAN for UPDATE etas performed on 1 processor
  Overlapped with UPDATE RHS

# SYNPLEX: parallel scheme (cont.)

- Task parallel multiple BTRAN for INVERT etas
- Columns distributed over $p$ processors for data parallel
  - PRICE
  - CHUZC
- INVERT overlapped with
  - BTRAN for INVERT etas
  - PRICE
  - CHUZC
  - FTRAN

# SYNPLEX in theory

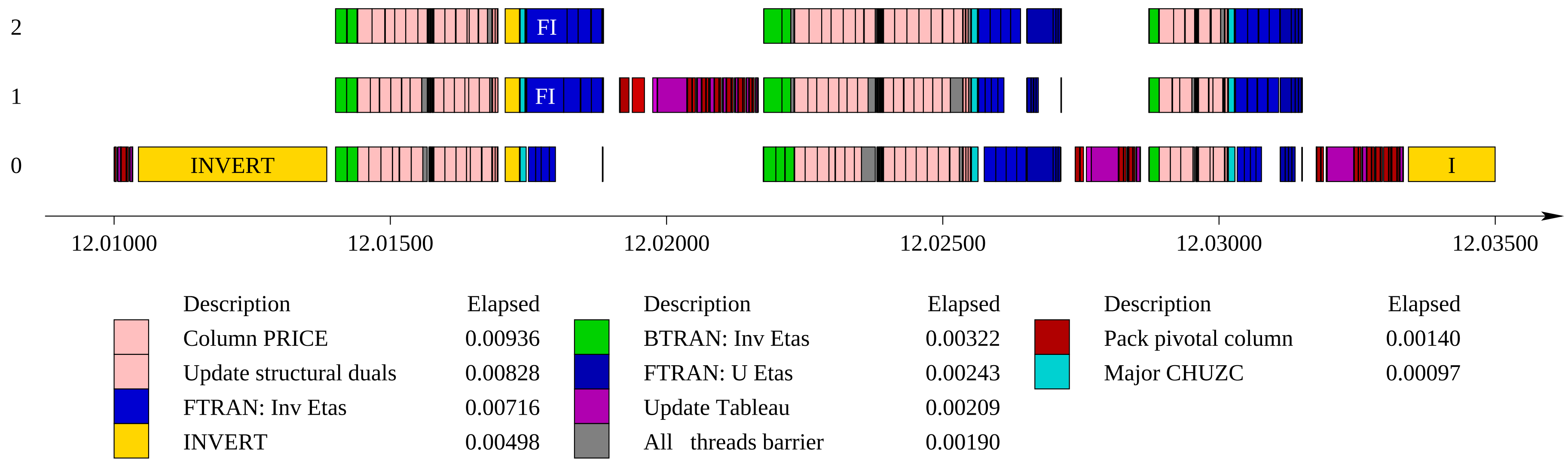


# Implementation

- SYNPLEX not yet fully implemented
  - Phase 2
  - Dantzig pricing
- SYNPLEX not yet efficient in serial
  - Column-wise PRICE
  - Doesn't exploit 'hypersparsity'
- SYNPLEX not yet fully parallelised
  - INVERT not overlapped
  - Minor iterations still sequential
  - Inefficient parallel UPDATE FTRAN
  - No overlapping of components

# Results (3 processors)

| Problem | Rows | Columns | Nonzeros | Speed-up |
|---|---|---|---|---|
| 25FV47 | 822 | 1571 | 11127 | 1.04 |
| PDS-06 | 9882 | 28655 | 82269 | 1.17 |




Why so poor?

- Implementation not fully parallel
- Poor load balance
- Longer execution time of some components

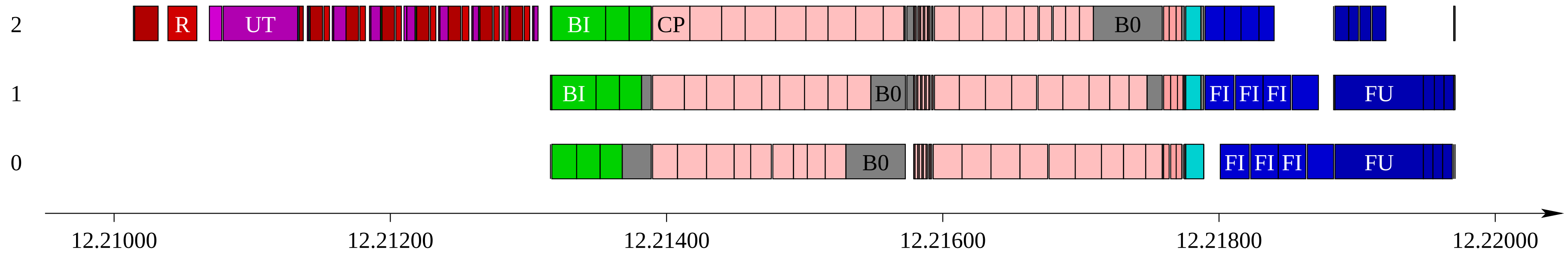
# Task manager

Implementation requires a task manager to facilitate

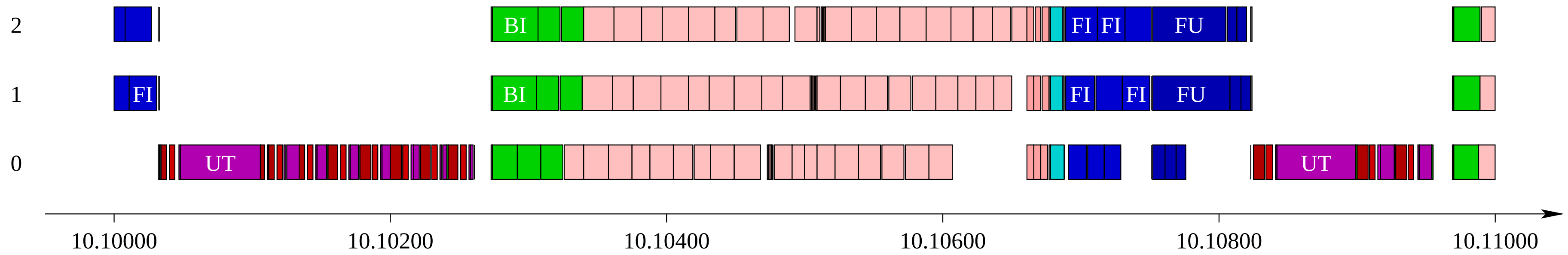
- Overlapping INVERT
- Parallel minor iterations
- Efficient parallel UPDATE FTRAN
- Efficient load balancing

# Task manager for load balancing in INVERT BTRAN and PRICE

- Without task manager




- With task manager




- Need to extend use of task manager for full load balancing

# Conclusions

- SYNPLEX addresses deficiencies of PARSMI
- Full implementation not possible with simple use of OpenMP
- Introducing task manager has already improved load balancing in PRICE
- Extension of task manager to other components is required for complete implementation