

A high performance dual revised simplex solver

Julian Hall and Qi Huangfu

School of Mathematics
University of Edinburgh

`jajhall@ed.ac.uk`

13th September 2011

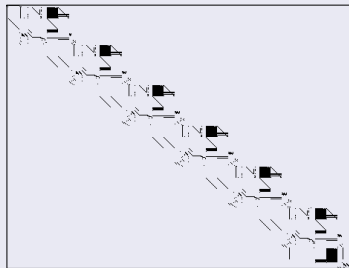
Linear programming (LP)

$$\begin{array}{ll}\text{minimize} & f = \mathbf{c}^T \mathbf{x} \\ \text{subject to} & A\mathbf{x} = \mathbf{b} \quad \mathbf{x} \geq \mathbf{0}\end{array}$$

Background

- Fundamental model in optimal decision-making
- Solution techniques
 - Simplex method (1947)
 - Interior point methods (1984)
- Large problems have
 - 10^3 – 10^7 variables
 - 10^3 – 10^7 constraints
- Matrix A is (usually) sparse

Example



STAIR: 356 rows, 467 columns and 3856 nonzeros

Solving LP problems

$$\begin{array}{ll}\text{minimize} & f = \mathbf{c}^T \mathbf{x} \\ \text{subject to} & A\mathbf{x} = \mathbf{b} \quad \mathbf{x} \geq \mathbf{0}\end{array}$$

Optimality conditions

- For a partition $\mathcal{B} \cup \mathcal{N}$ of the variable set with B nonsingular in
$$B\mathbf{x}_B + N\mathbf{x}_N = \mathbf{b}$$
- Eliminate $\mathbf{x}_B$ from the objective to give the **reduced LP**

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

where $\hat{\mathbf{b}} = B^{-1}\mathbf{b}$, $\hat{N} = B^{-1}N$, $\hat{f} = \mathbf{c}_B^T \hat{\mathbf{b}}$ and $\hat{\mathbf{c}}_N$ is given by

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

- Partition is optimal $\iff \hat{\mathbf{b}} \geq \mathbf{0}$ and $\hat{\mathbf{c}}_N \geq \mathbf{0}$

The (dual) simplex method

The algorithm

- Assuming **dual feasibility** ($\hat{\mathbf{c}}_N \geq \mathbf{0}$):
 - Choose a row $p \in \mathcal{B}$ that is attractive to reduce **primal infeasibility** ($\hat{b}_p < 0$)
 - Choose a column $q \in \mathcal{B}$ so dual feasibility is maintained
 - Exchange indices p and q between $\mathcal{B}$ and $\mathcal{N}$
 - Stop when $\hat{\mathbf{b}} \geq \mathbf{0}$
- One of the **Top 10 algorithms of the 20th Century**
 - Has not yet been parallelised
 - Modern hardware limitations mean we must try

Implementation: dual standard simplex method

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

Implementation: dual standard simplex method

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

In each iteration:

- Choose a row p with $\hat{b}_p < 0$
- Use the **pivotal row** p of $\hat{N}$ and $\hat{\mathbf{c}}_N$ to find the **pivotal column** q
- Exchange indices p and q between $\mathcal{B}$ and $\mathcal{N}$
- Update tableau corresponding to this **basis change**

$$\begin{aligned}\hat{N} &:= \hat{N} - \frac{1}{\hat{a}_{pq}} \hat{\mathbf{a}}_q \hat{\mathbf{a}}_p^T & \hat{\mathbf{b}} &:= \hat{\mathbf{b}} - \frac{\hat{b}_p}{\hat{a}_{pq}} \hat{\mathbf{a}}_q \\ \hat{\mathbf{c}}_N^T &:= \hat{\mathbf{c}}_N^T - \frac{\hat{c}_q}{\hat{a}_{pq}} \hat{\mathbf{a}}_p^T & -\hat{f} &:= -\hat{f} - \frac{\hat{c}_q}{\hat{a}_{pq}} \hat{b}_p\end{aligned}$$

Implementation: dual revised simplex method

- Maintains a representation of B^{-1} (rather than explicit $\hat{N}$)
- Rows and columns of $\hat{N}$ obtained as required

Implementation: dual revised simplex method

- Maintains a representation of B^{-1} (rather than explicit $\hat{N}$)
- Rows and columns of $\hat{N}$ obtained as required
- Pivotal row is

$$\hat{\mathbf{a}}_p^T = \boldsymbol{\pi}_p^T N, \text{ where } B^T \boldsymbol{\pi}_p = \mathbf{e}_p$$

- Pivotal column is

$$B \hat{\mathbf{a}}_q = \mathbf{a}_q, \text{ where } \mathbf{a}_q \text{ is column } q \text{ of } A$$

Implementation: dual revised simplex method

- Maintains a representation of B^{-1} (rather than explicit $\hat{N}$)
- Rows and columns of $\hat{N}$ obtained as required
- Pivotal row is

$$\hat{\mathbf{a}}_p^T = \boldsymbol{\pi}_p^T N, \text{ where } B^T \boldsymbol{\pi}_p = \mathbf{e}_p$$

- Pivotal column is

$$B \hat{\mathbf{a}}_q = \mathbf{a}_q, \text{ where } \mathbf{a}_q \text{ is column } q \text{ of } A$$

- Representation of B^{-1} is updated by exploiting

$$B := B + (\mathbf{a}_q - \mathbf{a}_p) \mathbf{e}_p^T$$

- Periodically a representation of B^{-1} is formed from scratch

Implementation: dual revised simplex method

- Maintains a representation of B^{-1} (rather than explicit $\hat{N}$)
- Rows and columns of $\hat{N}$ obtained as required
- Pivotal row is

$$\hat{\mathbf{a}}_p^T = \boldsymbol{\pi}_p^T N, \text{ where } B^T \boldsymbol{\pi}_p = \mathbf{e}_p$$

- Pivotal column is

$$B \hat{\mathbf{a}}_q = \mathbf{a}_q, \text{ where } \mathbf{a}_q \text{ is column } q \text{ of } A$$

- Representation of B^{-1} is updated by exploiting

$$B := B + (\mathbf{a}_q - \mathbf{a}_p) \mathbf{e}_p^T$$

- Periodically a representation of B^{-1} is formed from scratch
- Efficient solution of large sparse LP problems requires the revised simplex method

Summary: major computational components

Standard simplex method (SSM)

- Update tableau

$$\hat{N} := \hat{N} - (1/\hat{a}_{pq})\hat{\mathbf{a}}_q\hat{\mathbf{a}}_p^T$$

Revised simplex method (RSM)

- Operations

- Solve $B^T \pi_p = \mathbf{e}_p$
- Form $\hat{\mathbf{a}}_p^T = \pi_p^T B$
- Solve $B \hat{\mathbf{a}}_q = \mathbf{a}_q$

- Inversion of B

Summary: major computational components

Standard simplex method (SSM)

- Update tableau

$$\hat{N} := \hat{N} - (1/\hat{a}_{pq})\hat{\mathbf{a}}_q\hat{\mathbf{a}}_p^T$$

Revised simplex method (RSM)

- Operations

- Solve $B^T \pi_p = \mathbf{e}_p$
- Form $\hat{\mathbf{a}}_p^T = \pi_p^T B$
- Solve $B \hat{\mathbf{a}}_q = \mathbf{a}_q$

- Inversion of B

Distinctive features

- Vectors $\mathbf{e}_p$, $\mathbf{a}_q$ are **always sparse**
- B may be **highly reducible**
- B^{-1} **may be sparse**
- Vectors π_p , $\hat{\mathbf{a}}_p$ and $\hat{\mathbf{a}}_q$ **may be sparse**
- Efficient implementations must exploit these features
H and McKinnon (1998–2005)

Standard simplex: immediate scope for parallelism

Computational component

Algebraically	Computationally	Ref
$\hat{N} := \hat{N} - (1/\hat{a}_{pq})\hat{\mathbf{a}}_q\hat{\mathbf{a}}_p^T$	Dense outer product	U

Standard simplex: immediate scope for parallelism

Computational component

Algebraically	Computationally	Ref
$\hat{N} := \hat{N} - (1/\hat{a}_{pq})\hat{\mathbf{a}}_q\hat{\mathbf{a}}_p^T$	Dense outer product	U

Serial



Standard simplex: immediate scope for parallelism

Computational component

Algebraically	Computationally	Ref
$\hat{N} := \hat{N} - (1/\hat{a}_{pq})\hat{\mathbf{a}}_q\hat{\mathbf{a}}_p^T$	Dense outer product	U

Serial

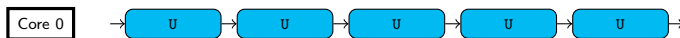


Standard simplex: immediate scope for parallelism

Computational component

Algebraically	Computationally	Ref
$\hat{N} := \hat{N} - (1/\hat{a}_{pq})\hat{\mathbf{a}}_q\hat{\mathbf{a}}_p^T$	Dense outer product	U

Serial



Standard simplex: immediate scope for parallelism

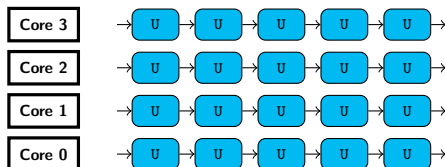
Computational component

Algebraically	Computationally	Ref
$\hat{N} := \hat{N} - (1/\hat{a}_{pq})\hat{\mathbf{a}}_q\hat{\mathbf{a}}_p^T$	Dense outer product	U

Serial



Parallel



Revised simplex: immediate scope for parallelism

Computational components

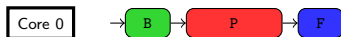
Algebraically	Computationally	Ref
$B^T \pi_p = \mathbf{e}_p$	Sparse linear system	B
$\hat{\mathbf{a}}_p^T = \pi_p^T N$	Sparse matrix-vector product	P
$B \hat{\mathbf{a}}_q = \mathbf{a}_q$	Sparse linear system	F
Represent B^{-1}	Sparse matrix inversion	I

Revised simplex: immediate scope for parallelism

Computational components

Algebraically	Computationally	Ref
$B^T \pi_p = \mathbf{e}_p$	Sparse linear system	B
$\hat{\mathbf{a}}_p^T = \pi_p^T N$	Sparse matrix-vector product	P
$B \hat{\mathbf{a}}_q = \mathbf{a}_q$	Sparse linear system	F
Represent B^{-1}	Sparse matrix inversion	I

Serial

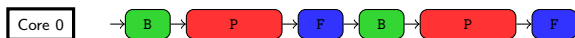


Revised simplex: immediate scope for parallelism

Computational components

Algebraically	Computationally	Ref
$B^T \pi_p = \mathbf{e}_p$	Sparse linear system	B
$\hat{\mathbf{a}}_p^T = \pi_p^T N$	Sparse matrix-vector product	P
$B \hat{\mathbf{a}}_q = \mathbf{a}_q$	Sparse linear system	F
Represent B^{-1}	Sparse matrix inversion	I

Serial

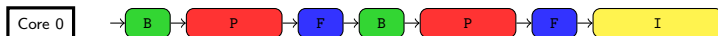


Revised simplex: immediate scope for parallelism

Computational components

Algebraically	Computationally	Ref
$B^T \pi_p = \mathbf{e}_p$	Sparse linear system	B
$\hat{\mathbf{a}}_p^T = \pi_p^T N$	Sparse matrix-vector product	P
$B \hat{\mathbf{a}}_q = \mathbf{a}_q$	Sparse linear system	F
Represent B^{-1}	Sparse matrix inversion	I

Serial



Revised simplex: immediate scope for parallelism

Computational components

Algebraically	Computationally	Ref
$B^T \pi_p = \mathbf{e}_p$	Sparse linear system	B
$\hat{\mathbf{a}}_p^T = \pi_p^T N$	Sparse matrix-vector product	P
$B \hat{\mathbf{a}}_q = \mathbf{a}_q$	Sparse linear system	F
Represent B^{-1}	Sparse matrix inversion	I

Serial



Revised simplex: immediate scope for parallelism

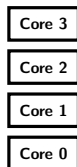
Computational components

Algebraically	Computationally	Ref
$B^T \pi_p = \mathbf{e}_p$	Sparse linear system	B
$\hat{\mathbf{a}}_p^T = \pi_p^T N$	Sparse matrix-vector product	P
$B \hat{\mathbf{a}}_q = \mathbf{a}_q$	Sparse linear system	F
Represent B^{-1}	Sparse matrix inversion	I

Serial



Parallel



Revised simplex: immediate scope for parallelism

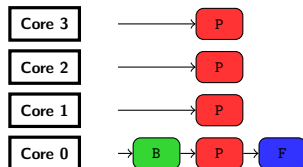
Computational components

Algebraically	Computationally	Ref
$B^T \pi_p = \mathbf{e}_p$	Sparse linear system	B
$\hat{\mathbf{a}}_p^T = \pi_p^T N$	Sparse matrix-vector product	P
$B \hat{\mathbf{a}}_q = \mathbf{a}_q$	Sparse linear system	F
Represent B^{-1}	Sparse matrix inversion	I

Serial



Parallel



Revised simplex: immediate scope for parallelism

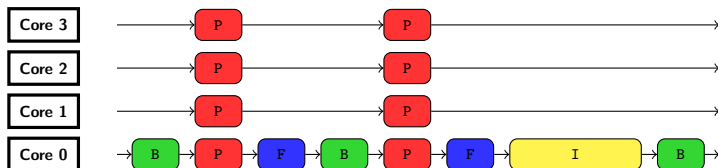
Computational components

Algebraically	Computationally	Ref
$B^T \pi_p = \mathbf{e}_p$	Sparse linear system	B
$\hat{\mathbf{a}}_p^T = \pi_p^T N$	Sparse matrix-vector product	P
$B \hat{\mathbf{a}}_q = \mathbf{a}_q$	Sparse linear system	F
Represent B^{-1}	Sparse matrix inversion	I

Serial



Parallel



Parallel simplex: past work

Data parallel standard simplex method

- Good parallel efficiency *was* achieved

Data parallel revised simplex method

- When $n \gg m$, cost of $\pi_p^T N$ dominates: significant speed-up *was* achieved
Bixby and Martin (2000)

Task parallel revised simplex method

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

Review: H (2010)

Dual revised simplex method with suboptimization

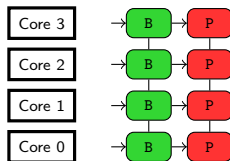
Algorithm

- Choose attractive row set $\mathcal{P}$
- Form rows $\mathcal{P}$ of B^{-1} : $B^T \hat{\pi}_{\mathcal{P}} = \mathbf{e}_{\mathcal{P}}$
- Form rows $\mathcal{P}$ of tableau: $\hat{\mathbf{a}}_{\mathcal{P}}^T = \pi_{\mathcal{P}}^T N$
- Perform dual standard simplex iterations on $\hat{\mathbf{a}}_{\mathcal{P}}^T$ to identify column set $\mathcal{Q}$
- Form pivotal columns for $q \in \mathcal{Q}$: $B \hat{\mathbf{a}}_{\mathcal{Q}} = \mathbf{a}_{\mathcal{Q}}$
- Use $\hat{\mathbf{a}}_{\mathcal{Q}}$ to update $\hat{\mathbf{b}}$

Dual revised simplex method with suboptimization

Algorithm

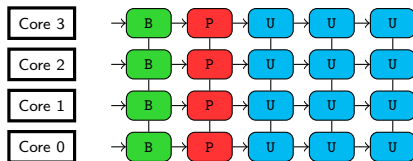
- Choose attractive row set $\mathcal{P}$
- Form rows $\mathcal{P}$ of B^{-1} : $B^T \hat{\pi}_{\mathcal{P}} = \mathbf{e}_{\mathcal{P}}$
- Form rows $\mathcal{P}$ of tableau: $\hat{\mathbf{a}}_{\mathcal{P}}^T = \pi_{\mathcal{P}}^T N$
- Perform dual standard simplex iterations on $\hat{\mathbf{a}}_{\mathcal{P}}^T$ to identify column set $\mathcal{Q}$
- Form pivotal columns for $q \in \mathcal{Q}$: $B \hat{\mathbf{a}}_{\mathcal{Q}} = \mathbf{a}_{\mathcal{Q}}$
- Use $\hat{\mathbf{a}}_{\mathcal{Q}}$ to update $\hat{\mathbf{b}}$



Dual revised simplex method with suboptimization

Algorithm

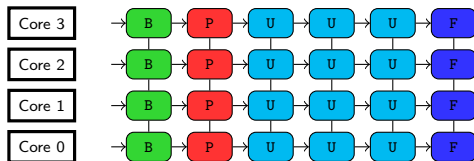
- Choose attractive row set $\mathcal{P}$
- Form rows $\mathcal{P}$ of B^{-1} : $B^T \hat{\pi}_{\mathcal{P}} = \mathbf{e}_{\mathcal{P}}$
- Form rows $\mathcal{P}$ of tableau: $\hat{\mathbf{a}}_{\mathcal{P}}^T = \pi_{\mathcal{P}}^T N$
- Perform dual standard simplex iterations on $\hat{\mathbf{a}}_{\mathcal{P}}^T$ to identify column set $\mathcal{Q}$
- Form pivotal columns for $q \in \mathcal{Q}$: $B \hat{\mathbf{a}}_{\mathcal{Q}} = \mathbf{a}_{\mathcal{Q}}$
- Use $\hat{\mathbf{a}}_{\mathcal{Q}}$ to update $\hat{\mathbf{b}}$



Dual revised simplex method with suboptimization

Algorithm

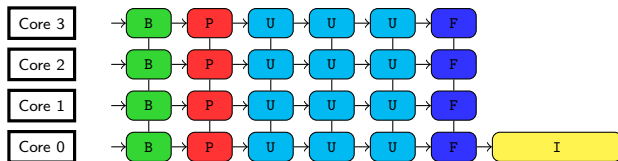
- Choose attractive row set $\mathcal{P}$
- Form rows $\mathcal{P}$ of B^{-1} : $B^T \hat{\pi}_{\mathcal{P}} = \mathbf{e}_{\mathcal{P}}$
- Form rows $\mathcal{P}$ of tableau: $\hat{\mathbf{a}}_{\mathcal{P}}^T = \pi_{\mathcal{P}}^T N$
- Perform dual standard simplex iterations on $\hat{\mathbf{a}}_{\mathcal{P}}^T$ to identify column set $\mathcal{Q}$
- Form pivotal columns for $q \in \mathcal{Q}$: $B \hat{\mathbf{a}}_{\mathcal{Q}} = \mathbf{a}_{\mathcal{Q}}$
- Use $\hat{\mathbf{a}}_{\mathcal{Q}}$ to update $\hat{\mathbf{b}}$



Dual revised simplex method with suboptimization

Algorithm

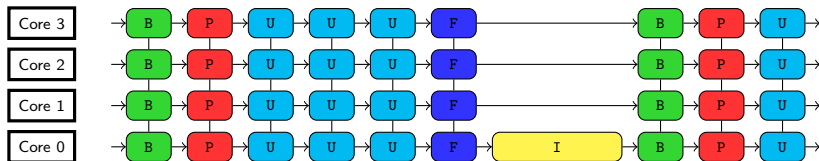
- Choose attractive row set $\mathcal{P}$
- Form rows $\mathcal{P}$ of B^{-1} : $B^T \hat{\pi}_{\mathcal{P}} = \mathbf{e}_{\mathcal{P}}$
- Form rows $\mathcal{P}$ of tableau: $\hat{\mathbf{a}}_{\mathcal{P}}^T = \pi_{\mathcal{P}}^T N$
- Perform dual standard simplex iterations on $\hat{\mathbf{a}}_{\mathcal{P}}^T$ to identify column set $\mathcal{Q}$
- Form pivotal columns for $q \in \mathcal{Q}$: $B \hat{\mathbf{a}}_{\mathcal{Q}} = \mathbf{a}_{\mathcal{Q}}$
- Use $\hat{\mathbf{a}}_{\mathcal{Q}}$ to update $\hat{\mathbf{b}}$



Dual revised simplex method with suboptimization

Algorithm

- Choose attractive row set $\mathcal{P}$
- Form rows $\mathcal{P}$ of B^{-1} : $B^T \hat{\pi}_{\mathcal{P}} = \mathbf{e}_{\mathcal{P}}$
- Form rows $\mathcal{P}$ of tableau: $\hat{\mathbf{a}}_{\mathcal{P}}^T = \pi_{\mathcal{P}}^T N$
- Perform dual standard simplex iterations on $\hat{\mathbf{a}}_{\mathcal{P}}^T$ to identify column set $\mathcal{Q}$
- Form pivotal columns for $q \in \mathcal{Q}$: $B \hat{\mathbf{a}}_{\mathcal{Q}} = \mathbf{a}_{\mathcal{Q}}$
- Use $\hat{\mathbf{a}}_{\mathcal{Q}}$ to update $\hat{\mathbf{b}}$



Implementation: ParISS

Environment

- Built using (generally) highly efficient core routines
- Written in C++ with OpenMP using the Intel C++ compiler
- Tested on a dual quad-core AMD Opteron 2378 system

Test problems

- Netlib + Kennington problems
- Two of the authors' problems

Clp

- Part of COIN-OR
- Written by John Forrest (IBM)
- Best open-source simplex solver
- Serial
- H is now project manager

Problem	1 core
dcp2	1.93
pds-06	1.49
pds-10	1.43
pds-20	1.39
pilot	1.06
cre-b	1.05
d2q06c	1.01
25fv47	1.00
greenbeb	0.99
fit2p	0.96
dfl001	0.93
ken-11	0.90
truss	0.89
osa-60	0.89
greenbea	0.87
ken-13	0.87
ken-18	0.86
maros-r7	0.78
osa-30	0.78
cre-d	0.77
dcp1	0.77
osa-14	0.74
stocfor3	0.69
qap8	0.57
pilot87	0.39
Mean	0.92

Results: parallel efficiency of ParISS 2.0

Problem	2 cores	4 cores	8 cores	best
pilot87	1.09	2.01	1.39	2.01
fit2p	1.30	1.62	1.99	1.99
stocfor3	1.47	1.43	1.85	1.85
ken-18	1.10	1.34	1.57	1.57
cre-d	1.41	1.50	1.44	1.50
qap8	1.17	1.39	1.48	1.48
pilot	1.21	1.48	1.22	1.48
ken-13	1.02	1.30	1.46	1.46
ken-11	1.07	1.34	1.40	1.40
pds-06	1.06	1.40	1.20	1.40
greenbeb	1.11	0.83	1.39	1.39
pds-10	0.97	1.33	1.34	1.34
osa-30	1.17	1.18	1.20	1.20
truss	1.03	1.19	1.09	1.19
pds-20	0.92	1.19	1.08	1.19
osa-60	1.16	1.04	1.06	1.16
df1001	0.96	1.01	1.14	1.14
cre-b	1.13	1.14	0.60	1.14
osa-14	1.08	1.02	1.05	1.08
greenbea	1.03	1.06	0.97	1.06
dcp2	0.90	0.92	1.06	1.06
dcp1	0.99	0.71	0.97	0.99
25fv47	0.82	0.85	0.67	0.85
d2q06c	0.77	0.82	0.74	0.82
maros-r7	0.28	0.10	0.24	0.28
Mean				1.21

Results: ParISS 2.0 (8 cores) relative to Clp (1 core)

Problem	Relative
fit2p	1.99
stocfor3	1.85
ken-18	1.57
qap8	1.48
ken-13	1.46
cre-d	1.44
ken-11	1.40
greenbeb	1.39
pilot87	1.39
pds-10	1.34
pilot	1.22
pds-06	1.20
osa-30	1.20
df1001	1.14
truss	1.09
pds-20	1.08
osa-60	1.06
dcp2	1.06
osa-14	1.05
dcp1	0.97
greenbea	0.97
d2q06c	0.74
25fv47	0.67
cre-b	0.60
maros-r7	0.24
Mean	1.11

Results: ParISS 2.0 (8 cores) relative to Clp (1 core)

Problem	Relative
fit2p	1.99
stocfor3	1.85
ken-18	1.57
qap8	1.48
ken-13	1.46
cre-d	1.44
ken-11	1.40
greenbeb	1.39
pilot87	1.39
pds-10	1.34
pilot	1.22
pds-06	1.20
osa-30	1.20
df1001	1.14
truss	1.09
pds-20	1.08
osa-60	1.06
dcp2	1.06
osa-14	1.05
dcp1	0.97
greenbea	0.97
d2q06c	0.74
25fv47	0.67
cre-b	0.60
maros-r7	0.24
Mean	1.11

Limits on performance

- Memory bound
 - Few flops per memory read
 - A and B^{-1} don't fit in cache!

Results: ParISS 2.0 (8 cores) relative to Clp (1 core)

Problem	Relative
fit2p	1.99
stocfor3	1.85
ken-18	1.57
qap8	1.48
ken-13	1.46
cre-d	1.44
ken-11	1.40
greenbeb	1.39
pilot87	1.39
pds-10	1.34
pilot	1.22
pds-06	1.20
osa-30	1.20
df1001	1.14
truss	1.09
pds-20	1.08
osa-60	1.06
dcp2	1.06
osa-14	1.05
dcp1	0.97
greenbea	0.97
d2q06c	0.74
25fv47	0.67
cre-b	0.60
maros-r7	0.24
Mean	1.11

Limits on performance

- Memory bound
 - Few flops per memory read
 - A and B^{-1} don't fit in cache!
- Wasted work
 - Some rows not used

Results: ParISS 2.0 (8 cores) relative to Clp (1 core)

Problem	Relative
fit2p	1.99
stocfor3	1.85
ken-18	1.57
qap8	1.48
ken-13	1.46
cre-d	1.44
ken-11	1.40
greenbeb	1.39
pilot87	1.39
pds-10	1.34
pilot	1.22
pds-06	1.20
osa-30	1.20
df1001	1.14
truss	1.09
pds-20	1.08
osa-60	1.06
dcp2	1.06
osa-14	1.05
dcp1	0.97
greenbea	0.97
d2q06c	0.74
25fv47	0.67
cre-b	0.60
maros-r7	0.24
Mean	1.11

Limits on performance

- Memory bound
 - Few flops per memory read
 - A and B^{-1} don't fit in cache!
- Wasted work
 - Some rows not used
- Load imbalance
 - Some systems solved faster than others
 - $|Q|$ may not a multiple of number of cores

Results: ParISS 2.0 (8 cores) relative to Clp (1 core)

Problem	Relative
fit2p	1.99
stocfor3	1.85
ken-18	1.57
qap8	1.48
ken-13	1.46
cre-d	1.44
ken-11	1.40
greenbeb	1.39
pilot87	1.39
pds-10	1.34
pilot	1.22
pds-06	1.20
osa-30	1.20
df1001	1.14
truss	1.09
pds-20	1.08
osa-60	1.06
dcp2	1.06
osa-14	1.05
dcp1	0.97
greenbea	0.97
d2q06c	0.74
25fv47	0.67
cre-b	0.60
maros-r7	0.24
Mean	1.11

Limits on performance

- Memory bound
 - Few flops per memory read
 - A and B^{-1} don't fit in cache!
- Wasted work
 - Some rows not used
- Load imbalance
 - Some systems solved faster than others
 - $|Q|$ may not a multiple of number of cores
- Inversion of B
 - Serial
 - Not overlapped with other operations

Summary and conclusions

- Performance improvements for general sparse LP problems
 - Average (best) of 18% (50%) relative to 1 core
 - Average (best) of 10% (50%) relative to Clp
- There is scope for parallel simplex
- Some of the limits on performance can be addressed