

Towards a practical parallelisation of the revised simplex method

Julian Hall, Edmund Smith and Huangfu Qi

School of Mathematics

University of Edinburgh

14th September 2010

Overview

- LP problems and the simplex method
- Scope parallelism
- (Preliminary) results
- Conclusions

Linear programming (LP) and the simplex method

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

Linear programming (LP) and the simplex method

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

- **Geometry:** Feasible points form a convex polyhedron

Linear programming (LP) and the simplex method

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

- **Geometry:** Feasible points form a convex polyhedron
- **Result:** At a vertex the variable set can be partitioned as $\mathcal{B} \cup \mathcal{N}$ and constraints as

$$B\mathbf{x}_B + N\mathbf{x}_N = \mathbf{b}$$

so B is nonsingular and $\mathbf{x}_N = \mathbf{0}$

Linear programming (LP) and the simplex method

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

- **Geometry:** Feasible points form a convex polyhedron
- **Result:** At a vertex the variable set can be partitioned as $\mathcal{B} \cup \mathcal{N}$ and constraints as

$$\mathbf{B}\mathbf{x}_B + \mathbf{N}\mathbf{x}_N = \mathbf{b}$$

so $\mathbf{B}$ is nonsingular and $\mathbf{x}_N = \mathbf{0}$

- Finding an optimal partition $\mathcal{B} \cup \mathcal{N}$ underpins the **simplex method**

Linear programming (LP) and the simplex method

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

- **Geometry:** Feasible points form a convex polyhedron
- **Result:** At a vertex the variable set can be partitioned as $\mathcal{B} \cup \mathcal{N}$ and constraints as

$$\mathbf{B}\mathbf{x}_B + \mathbf{N}\mathbf{x}_N = \mathbf{b}$$

so $\mathbf{B}$ is nonsingular and $\mathbf{x}_N = \mathbf{0}$

- Finding an optimal partition $\mathcal{B} \cup \mathcal{N}$ underpins the **simplex method** which
 - **Moves along edges** of the feasible region
 - **Terminates** at an optimal vertex

Linear programming (LP) and the simplex method

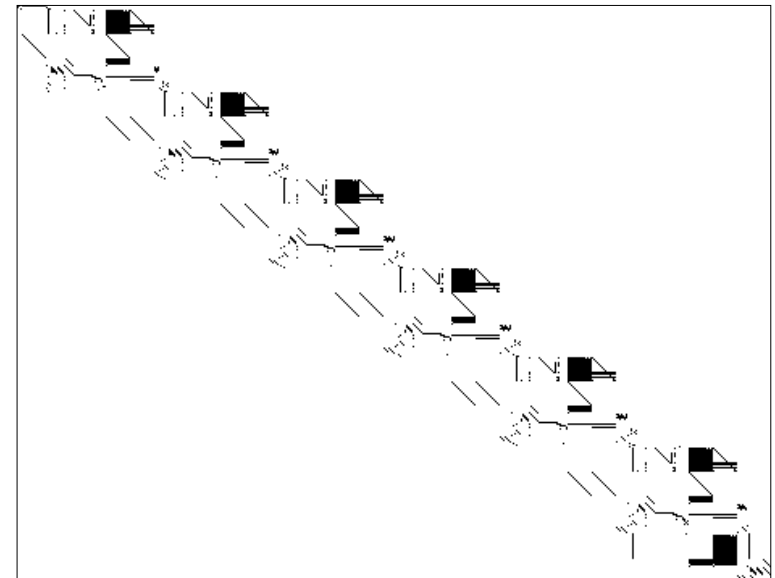
$$\begin{array}{ll}\text{minimize} & f = \mathbf{c}^T \mathbf{x} \\ \text{subject to} & \mathbf{Ax} = \mathbf{b} \quad \mathbf{x} \geq 0\end{array}$$

- **Geometry:** Feasible points form a convex polyhedron
- **Result:** At a vertex the variable set can be partitioned as $\mathcal{B} \cup \mathcal{N}$ and constraints as

$$\mathbf{B}\mathbf{x}_B + \mathbf{N}\mathbf{x}_N = \mathbf{b}$$

so $\mathbf{B}$ is nonsingular and $\mathbf{x}_N = 0$

- Finding an optimal partition $\mathcal{B} \cup \mathcal{N}$ underpins the **simplex method** which
 - **Moves along edges** of the feasible region
 - **Terminates** at an optimal vertex
 - Has transformed decision making since 1947



$\mathbf{A}$ is usually sparse

The reduced LP problem

At a vertex, for a partition $\mathcal{B} \cup \mathcal{N}$ with B nonsingular, the original problem is

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

The reduced LP problem

At a vertex, for a partition $\mathcal{B} \cup \mathcal{N}$ with B nonsingular, the original problem is

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

Eliminate $\mathbf{x}_B$ from the objective to give the **reduced LP problem**

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

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

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

The reduced LP problem

At a vertex, for a partition $\mathcal{B} \cup \mathcal{N}$ with B nonsingular, the original problem is

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

Eliminate $\mathbf{x}_B$ from the objective to give the **reduced LP problem**

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

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

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

Vertex is optimal $\iff \mathbf{s}_N \geq \mathbf{0}$

Implementation: standard simplex method

	$\mathcal{N}$	$\mathcal{B}$	RHS
1 : m	$\hat{N}$	I	$\hat{b}$
0	s_N^T	0^T	$-\hat{f}$

In each iteration:

- Choose variable x_q with $s_q < 0$ to increase from zero
- Use the **pivotal column** q of $\hat{N}$ and $\hat{b}$ to find the **pivotal row** p with $\alpha = \hat{b}_p / \hat{a}_{pq}$
- Exchange indices p and q between $\mathcal{B}$ and $\mathcal{N}$
- Update tableau corresponding to this **basis change**

$$\begin{aligned}
 \hat{N} &:= \hat{N} - (1/\hat{a}_{pq})\hat{a}_q\hat{a}_p^T & \hat{b} &:= \hat{b} - \alpha\hat{a}_q \\
 s_N^T &:= s_N^T - (s_q/\hat{a}_{pq})\hat{a}_q\hat{a}_p^T & -\hat{f} &:= -\hat{f} - \alpha s_q
 \end{aligned}$$

Implementation: revised simplex method

- Maintains a representation of B^{-1} (rather than $\hat{N}$)
- Pivotal column is

$$\hat{\mathbf{a}}_q = B^{-1} \mathbf{a}_q, \text{ where } \mathbf{a}_q \text{ is column } q \text{ of } A$$

Implementation: revised simplex method

- Maintains a representation of B^{-1} (rather than $\hat{N}$)
- Pivotal column is

$$\hat{\mathbf{a}}_q = B^{-1} \mathbf{a}_q, \text{ where } \mathbf{a}_q \text{ is column } q \text{ of } A$$

- Pivotal row is

$$\hat{\mathbf{a}}_p^T = \boldsymbol{\pi}_p^T N, \text{ where } \boldsymbol{\pi}_p^T = \mathbf{e}_p^T B^{-1}$$

Implementation: revised simplex method

- Maintains a representation of B^{-1} (rather than $\hat{N}$)
- Pivotal column is

$$\hat{\mathbf{a}}_q = B^{-1} \mathbf{a}_q, \text{ where } \mathbf{a}_q \text{ is column } q \text{ of } A$$

- Pivotal row is

$$\hat{\mathbf{a}}_p^T = \boldsymbol{\pi}_p^T N, \text{ where } \boldsymbol{\pi}_p^T = \mathbf{e}_p^T B^{-1}$$

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

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

- Periodically B^{-1} is formed from scratch

Implementation: revised simplex method

- Maintains a representation of B^{-1} (rather than $\hat{N}$)
- Pivotal column is

$$\hat{\mathbf{a}}_q = B^{-1} \mathbf{a}_q, \text{ where } \mathbf{a}_q \text{ is column } q \text{ of } A$$

- Pivotal row is

$$\hat{\mathbf{a}}_p^T = \boldsymbol{\pi}_p^T N, \text{ where } \boldsymbol{\pi}_p^T = \mathbf{e}_p^T B^{-1}$$

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

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

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

Implementation: hyper-sparsity

- Major computational cost each iteration is forming $B^{-1}\mathbf{r}_F$, $\mathbf{r}_B^T B^{-1}$ and $\mathbf{r}_\pi^T N$
- Vectors $\mathbf{r}_F$, $\mathbf{r}_B$ are always sparse
- Vector $\mathbf{r}_\pi$ may be sparse

Implementation: hyper-sparsity

- Major computational cost each iteration is forming $B^{-1}\mathbf{r}_F$, $\mathbf{r}_B^T B^{-1}$ and $\mathbf{r}_\pi^T N$
- Vectors $\mathbf{r}_F$, $\mathbf{r}_B$ are always sparse
- Vector $\mathbf{r}_\pi$ may be sparse
- If the results of these operations are (usually) sparse then LP is said to be **hyper-sparse**

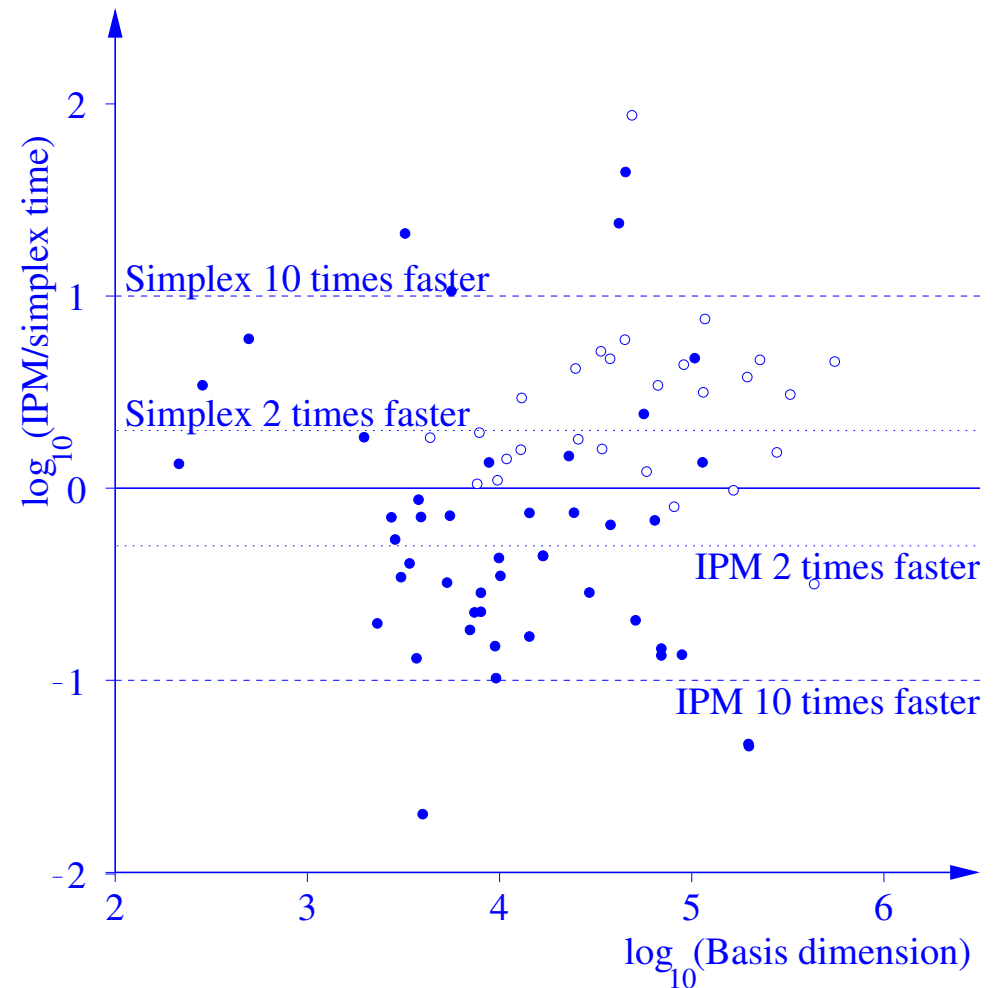
Implementation: hyper-sparsity

- Major computational cost each iteration is forming $B^{-1}\mathbf{r}_F$, $\mathbf{r}_B^T B^{-1}$ and $\mathbf{r}_\pi^T N$
- Vectors $\mathbf{r}_F$, $\mathbf{r}_B$ are always sparse
- Vector $\mathbf{r}_\pi$ may be sparse
- If the results of these operations are (usually) sparse then LP is said to be **hyper-sparse**
- LP structure means B^{-1} **is sparse**

Implementation: hyper-sparsity

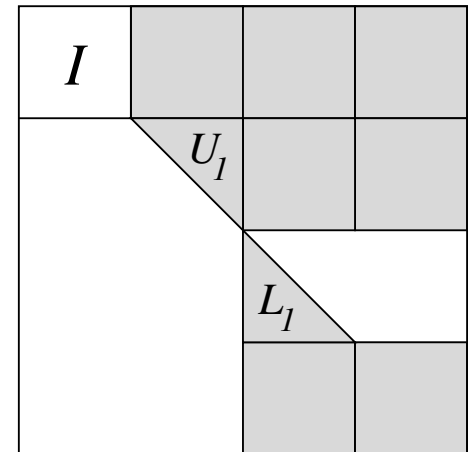
- Major computational cost each iteration is forming $B^{-1}r_F$, $r_B^T B^{-1}$ and $r_\pi^T N$
- Vectors r_F , r_B are always sparse
- Vector r_π may be sparse
- If the results of these operations are (usually) sparse then LP is said to be **hyper-sparse**
- LP structure means B^{-1} **is sparse**
- Simplex method typically better than interior point methods for hyper-sparse LP problems (○) but frequently not for other LPs (●)

H and McKinnon (1998–2005)



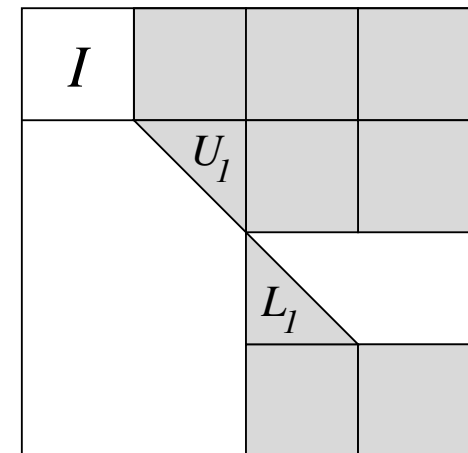
Implementation: inverting B

- Identify L_1 and U_1 : **triangularisation**
- Only bottom RH corner “bump” needs Gaussian elimination



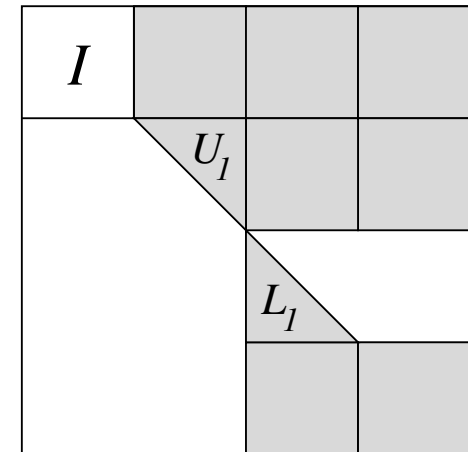
Implementation: inverting B

- Identify L_1 and U_1 : **triangularisation**
- Only bottom RH corner “bump” needs Gaussian elimination
- For many (hyper-sparse) LP problems, B is highly reducible
 - Bump is very small
 - Cost of triangularisation dominates factorization of bump



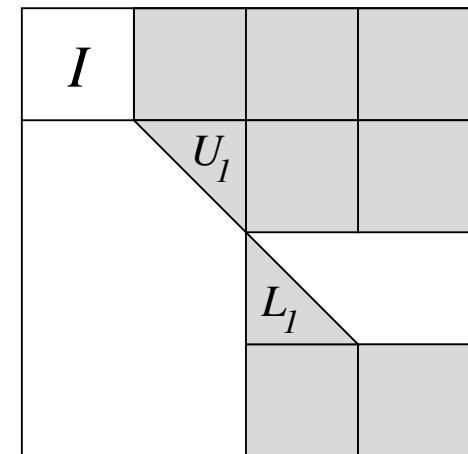
Implementation: inverting B

- Identify L_1 and U_1 : **triangularisation**
- Only bottom RH corner “bump” needs Gaussian elimination
- For many (hyper-sparse) LP problems, B is highly reducible
 - Bump is very small
 - Cost of triangularisation dominates factorization of bump
- For other LP problems, factorization dominates triangularisation



Implementation: inverting B

- Identify L_1 and U_1 : **triangularisation**
- Only bottom RH corner “bump” needs Gaussian elimination
- For many (hyper-sparse) LP problems, B is highly reducible
 - Bump is very small
 - Cost of triangularisation dominates factorization of bump
- For other LP problems, factorization dominates triangularisation
- Fill-in when representing B^{-1} is generally small
 - Typically $< 10\%$
 - Sometimes 100%
 - Rarely $> 1000\%$



Why try to exploit parallelism?

Why try to exploit parallelism?

- Moore's law drives core counts per processor, but clock speeds will stabilise

Why try to exploit parallelism?

- Moore's law drives core counts per processor, but clock speeds will stabilise
- Serial performance of simplex is spectacularly good
 - Flop count per iteration is near optimal
 - Number of iterations is near optimal

Why try to exploit parallelism?

- Moore's law drives core counts per processor, but clock speeds will stabilise
- Serial performance of simplex is spectacularly good
 - Flop count per iteration is near optimal
 - Number of iterations is near optimal
- Can't wait for faster serial processors or algorithmic improvement
- Simplex method must try to exploit parallelism

Conventional wisdom: Parallelism

- **Data parallel standard simplex method**
 - Good parallel efficiency *was* achieved
 - Totally uncompetitive with *serial* revised simplex method without prohibitive resources

Conventional wisdom: Parallelism

- **Data parallel standard simplex method**
 - Good parallel efficiency *was* achieved
 - Totally uncompetitive with *serial* revised simplex method without prohibitive resources
- **Data parallel revised simplex method**
 - Only immediate parallelism is in forming $\mathbf{r}_\pi^T \mathbf{N}$
 - When $n \gg m$, cost of $\mathbf{r}_\pi^T \mathbf{N}$ dominates: significant speed-up *was* achieved
Bixby and Martin (2000)

Conventional wisdom: Parallelism

- **Data parallel standard simplex method**
 - Good parallel efficiency *was* achieved
 - Totally uncompetitive with *serial* revised simplex method without prohibitive resources
- **Data parallel revised simplex method**
 - Only immediate parallelism is in forming $r_{\pi}^T N$
 - When $n \gg m$, cost of $r_{\pi}^T N$ 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
- All data and task parallel implementations compromised by serial inversion of B

Review: H (2010)

CPU or GPU or both?

Heterogeneous desk-top architectures

CPU:

- Fewer, faster cores
- Relatively slow memory transfer
- Welcomes algorithmically complex code
- Full range of development tools

CPU or GPU or both?

Heterogeneous desk-top architectures

CPU:

- Fewer, faster cores
- Relatively slow memory transfer
- Welcomes algorithmically complex code
- Full range of development tools

GPU:

- More, slower cores
- Relatively fast memory transfer
- Global communication is expensive/difficult
- Very limited development tools

CPU or GPU or both?

Heterogeneous desk-top architectures

CPU:

- Fewer, faster cores
- Relatively slow memory transfer
- Welcomes algorithmically complex code
- Full range of development tools

GPU:

- More, slower cores
- Relatively fast memory transfer
- Global communication is expensive/difficult
- Very limited development tools

CPU and GPU:

- Answer may be to combine CPU and GPU

Computation and Data

Ideally floating point operations per value read from memory should be as large as possible

Computation and Data

Ideally floating point operations per value read from memory should be as large as possible

- Achieved by level 3 BLAS, eg $A := A + \alpha XY^T$
 - May occur naturally in an implementation
 - Can be done in parallel sparse Cholesky
(eg) Talks: Hogg (2010), Andrianov (2010)

Computation and Data

Ideally floating point operations per value read from memory should be as large as possible

- Achieved by level 3 BLAS, eg $A := A + \alpha XY^T$
 - May occur naturally in an implementation
 - Can be done in parallel sparse Cholesky
(eg) Talks: Hogg (2010), Andrianov (2010)
- Simplex tableau update is level 2: $\hat{N} := \hat{N} - (1/\hat{a}_{pq})\hat{\mathbf{a}}_q\hat{\mathbf{a}}_p^T$

Computation and Data

Ideally floating point operations per value read from memory should be as large as possible

- Achieved by level 3 BLAS, eg $A := A + \alpha XY^T$
 - May occur naturally in an implementation
 - Can be done in parallel sparse Cholesky
(eg) Talks: Hogg (2010), Andrianov (2010)
- Simplex tableau update is level 2: $\hat{N} := \hat{N} - (1/\hat{a}_{pq})\hat{\mathbf{a}}_q\hat{\mathbf{a}}_p^T$
- Sparse revised simplex method is worse: computation/data ratio is
 - One for $B^{-1}\mathbf{r}_F$, $\mathbf{r}_B^T B^{-1}$ and $\mathbf{r}_\pi^T N$
 - Small for inversion of B
 - Several sparse vectors may be read as a cache line, but only one used

Computation and Data

Ideally floating point operations per value read from memory should be as large as possible

- Achieved by level 3 BLAS, eg $A := A + \alpha XY^T$
 - May occur naturally in an implementation
 - Can be done in parallel sparse Cholesky
(eg) Talks: Hogg (2010), Andrianov (2010)
- Simplex tableau update is level 2: $\hat{N} := \hat{N} - (1/\hat{a}_{pq})\hat{\mathbf{a}}_q\hat{\mathbf{a}}_p^T$
- Sparse revised simplex method is worse: computation/data ratio is
 - One for $B^{-1}\mathbf{r}_F$, $\mathbf{r}_B^T B^{-1}$ and $\mathbf{r}_\pi^T N$
 - Small for inversion of B
 - Several sparse vectors may be read as a cache line, but only one used
- Doesn't look good

Modified revised simplex method

Aim to improve memory use and scope for parallelism

Modified revised simplex method

Aim to improve memory use and scope for parallelism

- Exploit **block angular structure**
Talk: Smith (2010)

Modified revised simplex method

Aim to improve memory use and scope for parallelism

- Exploit **block angular structure**

Talk: Smith (2010)

- Perform standard simplex **suboptimization**

Primal: Orchard-Hays (1968) Dual: Rosander (1975)

Modified revised simplex method

Aim to improve memory use and scope for parallelism

- Exploit **block angular structure**

Talk: Smith (2010)

- Perform standard simplex **suboptimization**

Primal: Orchard-Hays (1968) Dual: Rosander (1975)

- Algorithmically

- ★ Identify attractive column (primal) or row (dual) slice of tableau
- ★ Identify set of basis changes

Modified revised simplex method

Aim to improve memory use and scope for parallelism

- Exploit **block angular structure**

Talk: Smith (2010)

- Perform standard simplex **suboptimization**

Primal: Orchard-Hays (1968) Dual: Rosander (1975)

- Algorithmically

- ★ Identify attractive column (primal) or row (dual) slice of tableau
- ★ Identify set of basis changes

- Computationally

- ★ Solve systems with multiple RHS
- ★ Update tableaux
- ★ Form matrix products with multiple vectors

Primal: Wunderling (1996), H and McKinnon (1995-2005)

Modified revised simplex method

Aim to improve memory use and scope for parallelism

- Exploit **block angular structure**
Talk: Smith (2010)
- Perform standard simplex **suboptimization**
Primal: Orchard-Hays (1968) Dual: Rosander (1975)
 - Algorithmically
 - ★ Identify attractive column (primal) or row (dual) slice of tableau
 - ★ Identify set of basis changes
 - Computationally
 - ★ Solve systems with multiple RHS
 - ★ Update tableaux
 - ★ Form matrix products with multiple vectorsPrimal: Wunderling (1996), H and McKinnon (1995-2005)
- Perform parallel triangularisation of B
Talk: H (2005)

Other relevant work

- Sparse matrix-vector product $A\mathbf{r}$
 - GPU out-performs comparable CPU for full $\mathbf{r}$
(eg) Vuduc *et al.* (2010)
 - Revised simplex $N^T\mathbf{r}$ has sparse $\mathbf{r}$

Other relevant work

- Sparse matrix-vector product $A\mathbf{r}$
 - GPU out-performs comparable CPU for full $\mathbf{r}$
(eg) Vuduc *et al.* (2010)
 - Revised simplex $N^T\mathbf{r}$ has sparse $\mathbf{r}$
- Solve $A^{-1}\mathbf{r}$ with given A^{-1} and full $\mathbf{r}$
 - CPU performance is memory bound
 - Speed-up factor limited to number of CPUs
(eg) Hogg *et al.* (2010)
 - Revised simplex $B^{-1}\mathbf{r}$, $\mathbf{r}^T B^{-1}$ have sparse $\mathbf{r}$

Dual revised simplex method with suboptimization

- Computational scheme
 - On CPU
 - ★ Choose attractive row set $\mathcal{P}$
 - ★ Form $\hat{\pi}_{\mathcal{P}}^T = e_{\mathcal{P}} B^{-1}$

Dual revised simplex method with suboptimization

- Computational scheme
 - On CPU
 - ★ Choose attractive row set $\mathcal{P}$
 - ★ Form $\hat{\pi}_{\mathcal{P}}^T = e_{\mathcal{P}} B^{-1}$
 - On GPU
 - ★ Form $\hat{a}_{\mathcal{P}}^T = \pi_{\mathcal{P}}^T N$
 - ★ Perform dual standard simplex iterations on $\hat{a}_{\mathcal{P}}^T$ to identify column set $\mathcal{Q}$

Dual revised simplex method with suboptimization

- Computational scheme
 - On CPU
 - ★ Choose attractive row set $\mathcal{P}$
 - ★ Form $\hat{\pi}_{\mathcal{P}}^T = e_{\mathcal{P}} B^{-1}$
 - On GPU
 - ★ Form $\hat{a}_{\mathcal{P}}^T = \pi_{\mathcal{P}}^T N$
 - ★ Perform dual standard simplex iterations on $\hat{a}_{\mathcal{P}}^T$ to identify column set $\mathcal{Q}$
 - On CPU
 - ★ Form $\hat{a}_{\mathcal{Q}} = B^{-1} a_{\mathcal{Q}}$
 - ★ Invert B

Dual revised simplex method with suboptimization

- Computational scheme
 - On CPU
 - ★ Choose attractive row set $\mathcal{P}$
 - ★ Form $\hat{\pi}_{\mathcal{P}}^T = e_{\mathcal{P}} B^{-1}$
 - On GPU
 - ★ Form $\hat{a}_{\mathcal{P}}^T = \pi_{\mathcal{P}}^T N$
 - ★ Perform dual standard simplex iterations on $\hat{a}_{\mathcal{P}}^T$ to identify column set $\mathcal{Q}$
 - On CPU
 - ★ Form $\hat{a}_{\mathcal{Q}} = B^{-1} a_{\mathcal{Q}}$
 - ★ Invert B
 - Aim to overlap CPU and GPU computation, and with CPU $\leftrightarrow$ GPU communication

Dual revised simplex method with suboptimization

- Computational scheme
 - On CPU
 - ★ Choose attractive row set $\mathcal{P}$
 - ★ Form $\hat{\pi}_{\mathcal{P}}^T = e_{\mathcal{P}} B^{-1}$
 - On GPU
 - ★ Form $\hat{a}_{\mathcal{P}}^T = \pi_{\mathcal{P}}^T N$
 - ★ Perform dual standard simplex iterations on $\hat{a}_{\mathcal{P}}^T$ to identify column set $\mathcal{Q}$
 - On CPU
 - ★ Form $\hat{a}_{\mathcal{Q}} = B^{-1} a_{\mathcal{Q}}$
 - ★ Invert B
 - Aim to overlap CPU and GPU computation, and with CPU $\leftrightarrow$ GPU communication
- Yet to be implemented
- Will make use of new ideas for serial dual revised simplex

Standard simplex method on a GPU

- Implemented by Smith as **i8**

Standard simplex method on a GPU

- Implemented by Smith as **i8**
- Experiments on machine with two quad-core AMD CPUs and GTX285 NVIDIA GPU
- For **dense** LP problems, best **in red**

Solver	Type	HPC	Time	Iterations	Speed (ms/iter)
gurobi	primal	serial	1357	16034	84.6
gurobi	dual	serial	976	14518	67.2
i7	primal	serial	6093	44384	137.3

Standard simplex method on a GPU

- Implemented by Smith as **i8**
- Experiments on machine with two quad-core AMD CPUs and GTX285 NVIDIA GPU
- For **dense** LP problems, best **in red**

Solver	Type	HPC	Time	Iterations	Speed (ms/iter)
gurobi	primal	serial	1357	16034	84.6
gurobi	dual	serial	976	14518	67.2
i7	primal	serial	6093	44384	137.3
i6	primal SSM	parallel	4039	288419	14.0
i8	primal SSM	GPU	800	221157	3.6

Standard simplex method on a GPU

- Implemented by Smith as **i8**
- Experiments on machine with two quad-core AMD CPUs and GTX285 NVIDIA GPU
- For **dense** LP problems, best **in red**

Solver	Type	HPC	Time	Iterations	Speed (ms/iter)
gurobi	primal	serial	1357	16034	84.6
gurobi	dual	serial	976	14518	67.2
i7	primal	serial	6093	44384	137.3
i6	primal SSM	parallel	4039	288419	14.0
i8	primal SSM	GPU	800	221157	3.6

- Not (really) of practical value but a good learning exercise
- No hope of beating serial solvers on sparse LP problems

Conclusions

- Identified need for simplex method to exploit parallelism
- Identified memory access as severe limitation
 - Worse with efficient simplex implementation
 - Worse with modern architectures
- Suggested approaches to improve memory use
- Shown that the standard simplex method will run fast on a GPU