A parallel revised simplex solver for large scale block angular LP problems

Julian Hall and Edmund Smith

School of Mathematics

University of Edinburgh

29th July 2010

Overview

- Block angular LP (BALP) problems
  - Structure
  - Origin
- Revised simplex method for BALP problems
  - Basis matrix and its inversion
  - Solution of linear systems
- Parallel revised simplex schemes for BALP
  - Structural data parallelism
  - Algorithmic task parallelism
- Conclusions

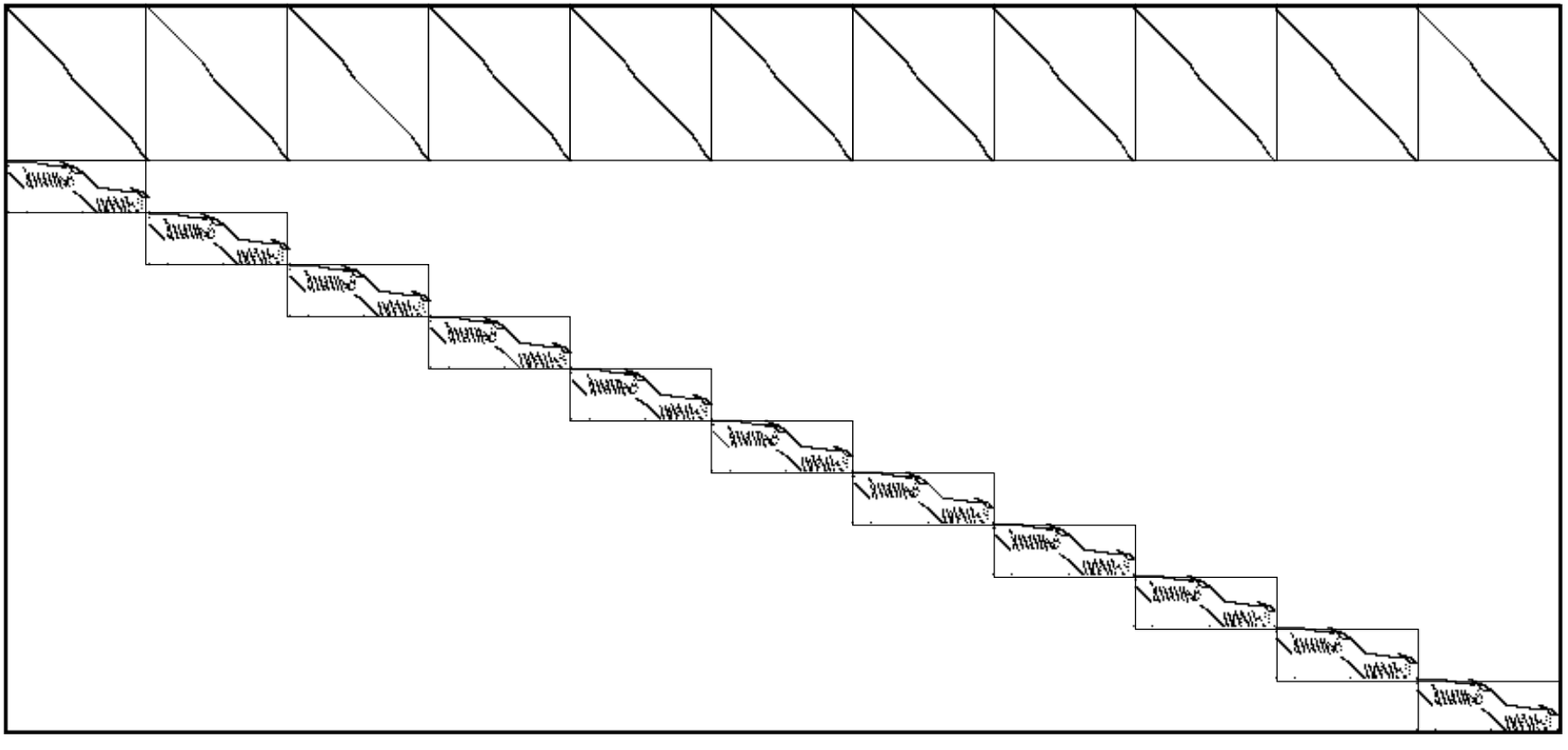
Block angular LP (BALP) problems

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

$$A = \begin{bmatrix} A_{00} & A_{01} & A_{02} & \dots & A_{0r} \\ & A_{11} & & & \\ & & A_{22} & & \\ & & & \ddots & \\ & & & & A_{rr} \end{bmatrix}$$

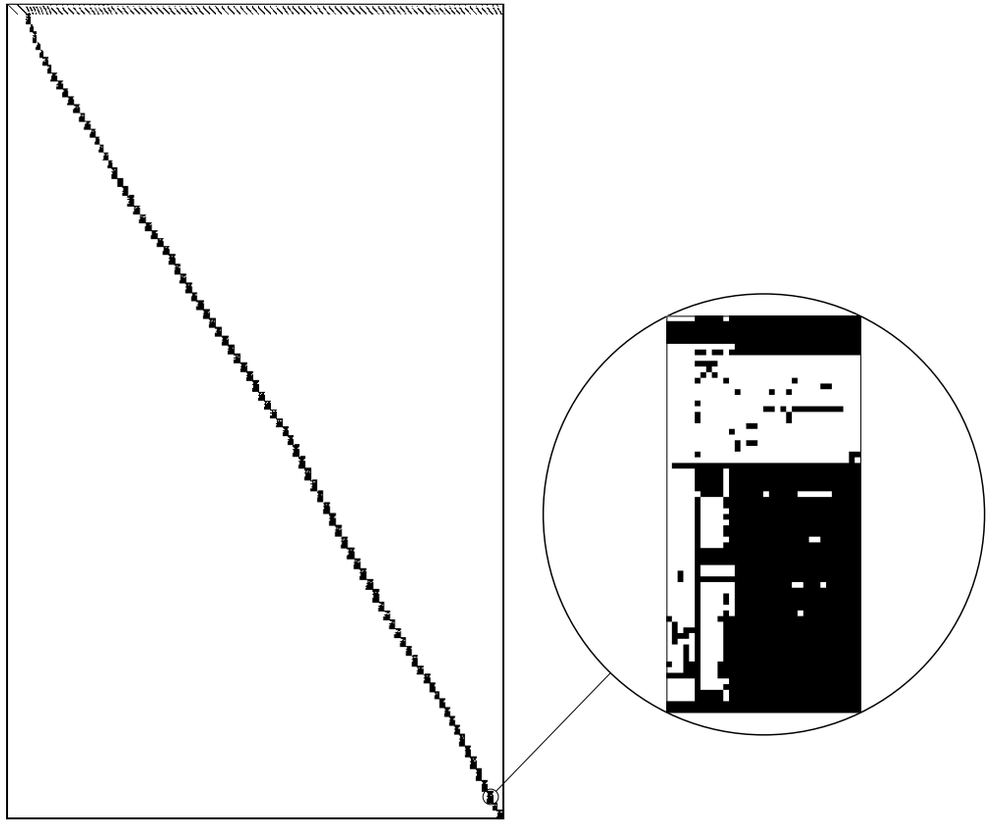
- Structure
  - The **linking rows** are $\begin{bmatrix} A_0 & A_{01} & A_{02} & \dots & A_{0r} \end{bmatrix}$
  - The **master columns** are $\begin{bmatrix} A_0 \\ 0 \end{bmatrix}$
  - The **diagonal blocks** are $A_{11}, \dots, A_{rr}$
- Origin
  - Occur naturally in (eg) decentralised planning and multicommodity flow
  - BALP structure can be identified in general sparse LPs

pds-02: 3518 rows, 7535 columns, 11 diagonal blocks



Source: Patient Distribution System, Carolan *et al.* (1990)

dcp1: 4950 rows, 3007 columns, 87 diagonal blocks



Source: Industrial, H (1997)

BALP form of general LP problems

A general LP problem may be partitioned into BALP form

Ferris and Horn (1998)

- Apply a graph partitioning algorithm to the matrix A
- Rows and columns are removed until remaining partitions are disjoint
- Order A according to partition with removed rows and columns in a border

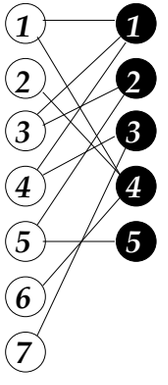
$$\begin{bmatrix} A_{00} & A_{01} & A_{02} & \dots & A_{0r} \\ A_{10} & A_{11} & & & \\ A_{20} & & A_{22} & & \\ \vdots & & & \ddots & \\ A_{r0} & & & & A_{rr} \end{bmatrix}$$

- Remove linking columns by splitting

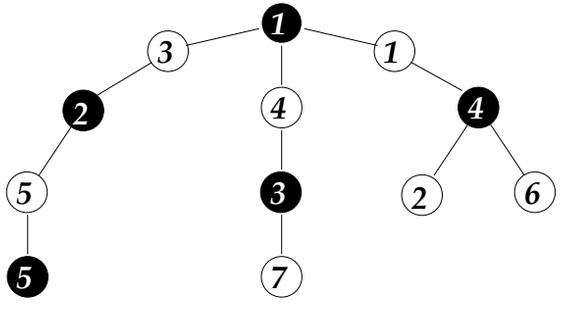
Example

- Matrix... with bipartite graph

	1	2	3	4	5	6	7
1	⋆		⋆	⋆			
2			⋆		⋆		
3				⋆			⋆
4	⋆	⋆				⋆	
5					⋆		



- Partitioned graph... yields BALP form



	3	5	4	7	1	2	6
1	⋆		⋆		⋆		
2	⋆	⋆					
5		⋆					
3			⋆	⋆			
4					⋆	⋆	⋆

Revised simplex method

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

- Major computational components of the revised simplex method (RSM) are
  - Invert B
  - Solve $B\boldsymbol{x} = \boldsymbol{r}$
  - Solve $B^T\boldsymbol{y} = \boldsymbol{d}$
  - Form $\boldsymbol{z} = N^T\boldsymbol{y}$

  where $[B : N]$ is a partition of A
- Data parallel RSM for general LP problems

  (eg) Forrest and Tomlin (1990), Shu (1995), Wunderling (1996), H and McKinnon (1996, 1998), Bixby and Martin (2000), H (2010)
  - Inversion and solving the systems are "hard" to parallelise
  - Forming $\boldsymbol{z} = N^T\boldsymbol{y}$ is "easy" to parallelise
  - Other computational components scan vectors and are "easy" to parallelise

Revised simplex method for BALP problems

- Matrices B and N in the revised simplex method inherit structure of A

$$B = \begin{bmatrix} B_{00} & B_{01} & \dots & B_{0r} \\ & B_{11} & & \\ & & \ddots & \\ & & & B_{rr} \end{bmatrix} \qquad N = \begin{bmatrix} N_{00} & N_{01} & \dots & N_{0r} \\ & N_{11} & & \\ & & \ddots & \\ & & & N_{rr} \end{bmatrix}$$

- Operations with B and N must exploit its structure
  (eg) Lasdon (1970)

RSM for BALP problems: inverting B

$$B = \begin{bmatrix} B_{00} & B_{01} & \dots & B_{0r} \\ & B_{11} & & \\ & & \ddots & \\ & & & B_{rr} \end{bmatrix}$$

- Matrices B_{ii}, $i = 1, \dots, r$ are rectangular
  - Partition of B_{ii} as $[\, R_i \quad T_i \,]$ with T_i nonsingular is guaranteed
  - Yields structure

$$\left[\begin{array}{cccc|ccc} S_0 & S_1 & \dots & S_r & C_1 & \dots & C_r \\ \hline & R_1 & & & T_1 & & \\ & & \ddots & & & \ddots & \\ & & & R_r & & & T_r \end{array}\right]$$

- More simply, write

$$B = \begin{bmatrix} S & C \\ R & T \end{bmatrix}$$

RSM for BALP problems: inverting B

- For $B = \begin{bmatrix} S & C \\ R & T \end{bmatrix}$
  - S is (hopefully!) small and square
  - C is unstructured and rectangular
  - R is block-rectangular
  - T is block-diagonal
- Consider decomposition

$$\begin{bmatrix} S & C \\ R & T \end{bmatrix} = \begin{bmatrix} I & C \\ & T \end{bmatrix} \begin{bmatrix} W & \\ \hat{R} & I \end{bmatrix}$$

  - $\hat{R} = T^{-1}R$ has the same structure as R
  - $W = S - CT^{-1}R = S - C\hat{R}$ is the **Schur complement** of T
- To solve systems involving B requires operations with
  - matrices C and $\hat{R}$
  - invertible representations of T and W

RSM for BALP problems: solving $Bx = r$

- To solve $\begin{bmatrix} S & C \\ R & T \end{bmatrix} \begin{bmatrix} x_0 \\ x_\bullet \end{bmatrix} = \begin{bmatrix} r_0 \\ r_\bullet \end{bmatrix}$

Solve	Tz	$=$	$r_\bullet$
Form	w	$=$	$r_0 - Cz$
Solve	Wx_0	$=$	w
Form	$x_\bullet$	$=$	$z - \hat{R}x_0$

- In detail, partitioning $r_\bullet$ and $x_\bullet$ according to blocks,

Solve	$T_i z_i$	$=$	r_i	$i = 1, \ldots, r$
Form	w	$=$	$r_0 - \sum_{i=1}^{r} C_i z_i$	
Solve	Wx_0	$=$	w	
Form	x_i	$=$	$z_i - \hat{R}_i x_0$	$i = 1, \ldots, r$

RSM for BALP problems: exploiting RHS of $Bx = r$

When $Bx = r$ is solved to form a column of the tableau, r is a column of A

- Let r be column q in block i, then $r = \begin{bmatrix} [A_{0i}]_q \\ 0 \\ [A_{ii}]_q \\ 0 \end{bmatrix}$
- Exploiting this structure, $Bx = r$ is solved as

Solve	$T_i z_i$	$=$	r_i	
Form	w	$=$	$r_0 - C_i z_i$	
Solve	$W x_0$	$=$	w	
Form	x_i	$=$	$z_i - \hat{R}_i x_0$	$i = 1, \ldots, r$

- Insufficient scope for data parallelism
- Hyper-sparsity exploited structurally

RSM for BALP problems: solving $B^T \boldsymbol{y} = \boldsymbol{d}$

- To solve $\begin{bmatrix} S^T & R^T \\ C^T & T^T \end{bmatrix} \begin{bmatrix} \boldsymbol{y}_0 \\ \boldsymbol{y}_\bullet \end{bmatrix} = \begin{bmatrix} \boldsymbol{d}_0 \\ \boldsymbol{d}_\bullet \end{bmatrix}$

Form	$\boldsymbol{w}$	$=$	$\boldsymbol{d}_0 - \hat{R}^T \boldsymbol{d}_\bullet$
Solve	$W^T \boldsymbol{y}_0$	$=$	$\boldsymbol{w}$
Form	$\boldsymbol{z}$	$=$	$\boldsymbol{d}_\bullet - C^T \boldsymbol{y}_0$
Solve	$T^T \boldsymbol{y}_\bullet$	$=$	$\boldsymbol{z}$

- In detail, partitioning $\boldsymbol{d}_\bullet$ and $\boldsymbol{y}_\bullet$ according to blocks,

Form	$\boldsymbol{w}$	$=$	$\boldsymbol{d}_0 - \sum_{i=1}^{r} \hat{R}_i^T \boldsymbol{d}_i$	
Solve	$W^T \boldsymbol{y}_0$	$=$	$\boldsymbol{w}$	
Form	$\boldsymbol{z}_i$	$=$	$\boldsymbol{d}_i - C_i^T \boldsymbol{y}_0$	$i = 1, \ldots, r$
Solve	$T_i^T \boldsymbol{y}_i$	$=$	$\boldsymbol{z}_i$	$i = 1, \ldots, r$

RSM for BALP problems: exploiting RHS of $B^T\boldsymbol{y} = \boldsymbol{d}$

When $B^T\boldsymbol{y} = \boldsymbol{d}$ is solved to form a row of B^{-1}, $\boldsymbol{d}$ is a column of I

- Let $\boldsymbol{d}$ correspond to row p in block i, then $\boldsymbol{d} = \begin{bmatrix} 0 \\ \boldsymbol{e}_p \\ 0 \end{bmatrix}$
- Exploiting this structure, $B^T\boldsymbol{y} = \boldsymbol{d}$ is solved as

Form	$\boldsymbol{w}$	$=$	$\boldsymbol{d}_0 - \hat{R}_i^T\boldsymbol{e}_p$	
Solve	$W^T\boldsymbol{y}_0$	$=$	$\boldsymbol{w}$	
Form	$\boldsymbol{z}_i$	$=$	$\boldsymbol{d}_i - C_i^T\boldsymbol{y}_0$	$i = 1, \ldots, r$
Solve	$T_i^T\boldsymbol{y}_i$	$=$	$\boldsymbol{z}_i$	$i = 1, \ldots, r$

- Still scope for **data parallelism**

RSM for BALP problems: forming $\boldsymbol{z} = N^T \boldsymbol{y}$

$$N = \begin{bmatrix} N_{00} & N_{01} & \dots & N_{0r} \\ & N_{11} & & \\ & & \ddots & \\ & & & N_{rr} \end{bmatrix}$$

- Form $\boldsymbol{z} = N^T \boldsymbol{y}$ as

  Form $\quad \boldsymbol{z}_0 \quad = \quad N_{00}^T \boldsymbol{y}_0$

  Form $\quad \boldsymbol{z}_i \quad = \quad N_{0i}^T \boldsymbol{y}_0 + N_{ii}^T \boldsymbol{y}_i \quad i = 1, \dots, r$

- Scope for **data parallelism**

A parallel revised simplex scheme for BALP

- Scope for data parallelism when exploiting BALP structure
  - When inverting B: **full**
  - When solving $B\boldsymbol{x} = \boldsymbol{r}$: **little**
  - When solving $B^T\boldsymbol{y} = \boldsymbol{d}$: **some**
  - When forming $\boldsymbol{z} = N^T\boldsymbol{y}$: **full**
- Data parallelism is insufficient (Amdahl's law)
- Exploit **task parallelism** via tableau simplex method suboptimization

Revised simplex method with suboptimization

- **Primal** simplex method: Orchard-Hays (1968)
  - Form a small subset of tableau **columns**
  - Perform minor iterations of standard primal simplex method
  - Update reduced costs and weights
- **Dual** simplex method: Rosander (1975)
  - Form a small subset of tableau **rows**
  - Perform minor iterations of standard dual simplex method
  - Update RHS and dual weights
- Parallel primal revised simplex method with suboptimization
  Wunderling (1996), H and McKinnon (1996, 1998)
- Forming tableau rows and columns and updating reduced costs/weights requires
  - Solution of linear systems with multiple RHS
  - Forming products between matrices and multiple vectors
- Updating tableau parallelises readily
  (eg) Eckstein *et al.* (1995), Lentini *et al.* (1995)

Design of a parallel primal simplex solver with suboptimization

- Aiming at desktop parallelism
- Architecture dictates algorithm
  - Assume p CPU processors, each with c cores: perhaps $p = 2$ and $c = 4$
  - Assume GPU is available
- Distribution of problem
  - Natural BALP problems: one set of blocks per core
  - Partitioned problems: one block per core
- Distribution of computational components
  - Inversion and system solution on CPUs
  - Matrix-vector product and minor iterations on GPU?

Prototype parallel primal simplex scheme

- Choose a set of attactive columns $q \in \mathcal{Q}$
- Solve $B\hat{\boldsymbol{a}}_q = \boldsymbol{a}_q$, $q \in \mathcal{Q}$
- Perform minor iterations to identify set $\mathcal{P}$ of pivotal rows
- Solve $B^T\boldsymbol{\pi}_p = \boldsymbol{e}_p$, $p \in \mathcal{P}$
- Form $\hat{\boldsymbol{a}}_p^T = \boldsymbol{\pi}_p^T N$, $p \in \mathcal{P}$
- Update reduced costs and (Devex) weights
- Periodically reinvert B

Prototype parallel dual simplex scheme

- Choose a set of attactive rows $p \in \mathcal{P}$
- Solve $B^T \boldsymbol{\pi}_p = \boldsymbol{e}_p$, $p \in \mathcal{P}$
- Form $\hat{\boldsymbol{a}}_p^T = \boldsymbol{\pi}_p^T N$, $p \in \mathcal{P}$
- Perform minor iterations to identify set $\mathcal{Q}$ of pivotal columns
- Solve $B\hat{\boldsymbol{a}}_q = \boldsymbol{a}_q$, $q \in \mathcal{Q}$
- Update RHS and (steepest edge) weights
- Periodically reinvert B

Conclusions

- Identified BALP structure as offering significant scope for parallel revised simplex
- Defined parallel primal and dual simplex schemes for BALP problems
- Implementation is "in progress"

References

[1] R. E. Bixby and A. Martin. Parallelizing the dual simplex method. *INFORMS Journal on Computing*, 12:45–56, 2000.

[2] W. J. Carolan, J. E. Hill, J. L. Kennington, S. Niemi, and S. J. Wichmann. An empirical evaluation of the KORBX algorithms for military airlift applications. *Operations Research*, 38(2):240–248, 1990.

[3] J. Eckstein, İ. İ. Boduroğlu, L. Polymenakos, and D. Goldfarb. Data-parallel implementations of dense simplex methods on the Connection Machine CM-2. *ORSA Journal on Computing*, 7(4):402–416, 1995.

[4] M. C. Ferris and J. D. Horn. Partitioning mathematical programs for parallel solution. *Mathematical Programming*, 80:35–61, 1998.

[5] J. J. H. Forrest and J. A. Tomlin. Vector processing in the simplex and interior methods for linear programming. *Annals of Operations Research*, 22:71–100, 1990.

[6] J. A. J. Hall. Towards a practical parallelisation of the simplex method. *Computational Management Science*, 7(2):139–170, 2010.

[7] J. A. J. Hall and K. I. M. McKinnon. PARSMI, a parallel revised simplex algorithm incorporating minor iterations and Devex pricing. In J. Waśniewski, J. Dongarra, K. Madsen, and D. Olesen, editors, *Applied Parallel Computing*, volume 1184 of *Lecture Notes in Computer Science*, pages 67–76. Springer, 1996.

[8] J. A. J. Hall and K. I. M. McKinnon. ASYNPLEX, an asynchronous parallel revised simplex method algorithm. *Annals of Operations Research*, 81:27–49, 1998.

[9] L. S. Lasdon. *Optimization theory for large systems*, chapter 6. Macmillan, 1970.

[10] M. Lentini, A. Reinoza, A. Teruel, and A. Guillen. SIMPAR: a parallel sparse simplex. *Computational and Applied Mathematics*, 14(1):49–58, 1995.

[11] W. Orchard-Hays. *Advanced Linear programming computing techniques*. McGraw-Hill, New York, 1968.

[12] R. R. Rosander. Multiple pricing and suboptimization in dual linear programming algorithms. *Mathematical Programming Study*, 4:108–117, 1975.

[13] W. Shu. Parallel implementation of a sparse simplex algorithm on MIMD distributed memory computers. *Journal of Parallel and Distributed Computing*, 31(1):25–40, November 1995.

[14] R. Wunderling. Paralleler und objektorientierter simplex. Technical Report TR-96-09, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 1996.