Parallel revised simplex for primal block angular LP problems

Julian Hall and Edmund Smith

School of Mathematics
University of Edinburgh

jajhall@ed.ac.uk

28th June 2012

Overview

- Block angular LP (BALP) problems and their identification
- Computational components of the revised simplex method
- Exploiting parallelism via BALP structure
- Results
- Observations

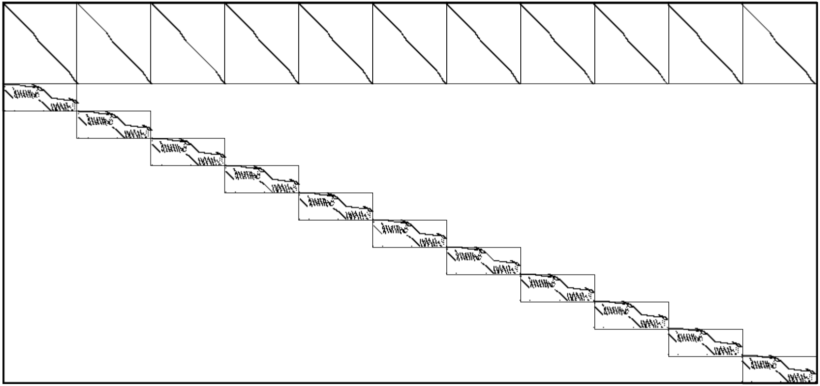
Block angular linear programming (BALP) problems)

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

$$A = \begin{bmatrix} A_{01} & A_{02} & \dots & A_{0r} \\ A_1 & & & \\ & A_2 & & \\ & & \ddots & \\ & & & A_r \end{bmatrix}$$

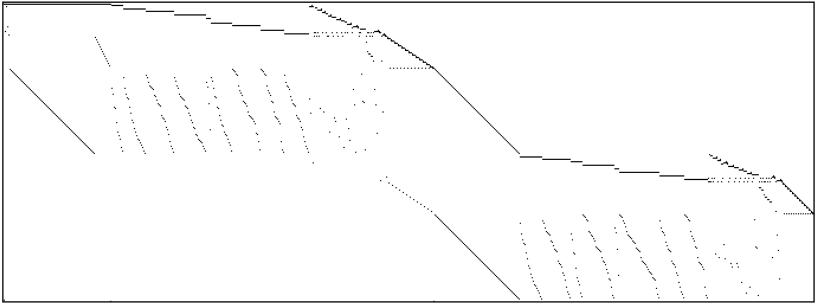
- Structure
  - The **linking rows** are $\begin{bmatrix} A_{01} & A_{02} & \dots & A_{0r} \end{bmatrix}$
  - The **diagonal blocks** are $\begin{bmatrix} A_{11} & A_{22} & \dots & A_{rr} \end{bmatrix}$
  - Diagonal blocks can be many or few; dense or sparse
- Origin
  - Occur naturally in (eg) decentralised planning and multicommodity flow
  - BALP structure can be identified in general sparse LPs

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

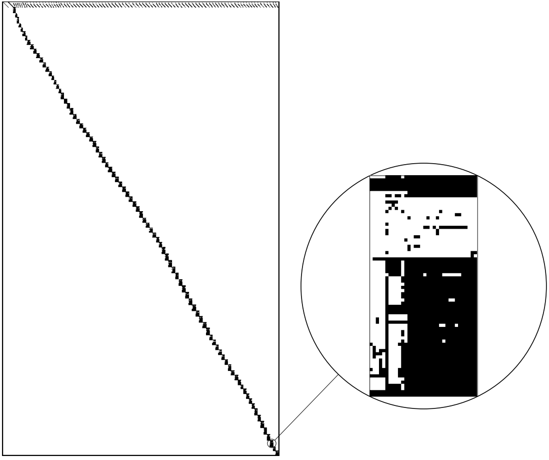


Source: Patient Distribution System, Carolan *et al.* (1990)

Diagonal block of pds-02



dcp1: 4950 rows, 3007 columns, 87 diagonal blocks



Source: Industrial, Hall (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
- Remove rows and columns until remaining partitions are disjoint
- Order A according to partition with removed rows and columns in a border

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

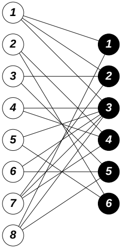
- Remove linking columns by splitting

Example

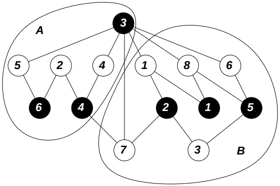
Matrix...

	1	2	3	4	5	6	7	8
1	$\star$							$\star$
2	$\star$		$\star$				$\star$	
3	$\star$			$\star$	$\star$	$\star$	$\star$	$\star$
4		$\star$		$\star$			$\star$	
5			$\star$			$\star$		$\star$
6		$\star$			$\star$			

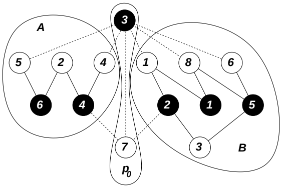
... with bipartite graph...



... partitioned...



... and borderised...



Example (cont.)

... yields arrow-head form

	7	2	4	5	1	3	6	8	
3	⋆		⋆	⋆	⋆		⋆	⋆	p_0
4	⋆	⋆	⋆						A
6		⋆		⋆					A
1					⋆			⋆	B
2	⋆				⋆	⋆			B
5						⋆	⋆	⋆	B
	p_0	A	A	A	B	B	B	B	

- Split the linking column
  - Add duplicate variable x_9
  - Add row 7: $x_7 - x_9 = 0$
- Yields row-linked BALP form

	2	4	5	7	1	3	6	8	9
7				⋆					⋆
3		⋆	⋆	⋆	⋆		⋆	⋆	
4	⋆	⋆		⋆					
6	⋆		⋆						
1					⋆			⋆	
2					⋆	⋆			⋆
5						⋆	⋆	⋆	

Revised simplex method (RSM)

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

RSM: Major computational components

- Factorise B
- Solve $B\mathbf{x} = \mathbf{r}$
- Solve $B^T\mathbf{y} = \mathbf{d}$
- Form $\mathbf{z} = N^T\mathbf{y}$

where $[B : N]$ is a partition of A

(Data) parallel RSM for general LP problems

- Factorisation and solving the systems are "hard" to parallelise
- Forming $\mathbf{z} = N^T\mathbf{y}$ is "easy" to parallelise

(eg) Forrest and Tomlin (1990), Shu (1995), Wunderling (1996), H and McKinnon (1996, 1998), Bixby and Martin (2000), H (2010), H and Huangfu (2012)

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)

Factorising B

- Factorisation must exploit the structure of

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

  - Partition of B_{ii} as $\begin{bmatrix} R_i & T_i \end{bmatrix}$ 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 $B = \begin{bmatrix} S & C \\ R & T \end{bmatrix}$

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

On $\hat{R} = T^{-1}R$

- Use of $\hat{R} = T^{-1}R$ analogous to Block-LU update for general simplex

  Eldersveld and Saunders (1990)

- Explicit $\hat{R} = T^{-1}R$ can be avoided
  Replace each operation with $\hat{R}$ (possibly dense) by
  - Operation with T^{-1} (again)
  - Operation with R (sparse)

  analogous to Schur complement update for general simplex

  Bisschop and Meeraus (1977)

- But overhead of $\hat{R}$ is very small, so use it

Alternative decompositions

Alternative Block LU

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

where $\hat{C} = CT^{-1}$

Less attractive since (unlike $\hat{R}$) $\hat{C}$ is not structured

LU + Schur complement

$$\begin{bmatrix} S & C \\ R & T \end{bmatrix} = \begin{bmatrix} W & \tilde{C} \\ & L \end{bmatrix} \begin{bmatrix} I & \\ \tilde{R} & U \end{bmatrix}$$

where $T = LU$, $\tilde{C} = CU^{-1}$ and $\tilde{R} = L^{-1}R$

Possibly attractive, but

- $\tilde{C}$ and $\tilde{R}$ may have greater cumulative fill-in than $\hat{R}$
- Operations with T^{-1} are no longer a "black box"

Solving $B\mathbf{x} = \mathbf{r}$

- The system $\begin{bmatrix} S & C \\ R & T \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_\bullet \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_\bullet \end{bmatrix}$ may be solved as

$$\begin{array}{lrcl}
\text{Solve} & T\mathbf{z} & = & \mathbf{r}_\bullet \\
\text{Form} & \mathbf{w} & = & \mathbf{r}_1 - C\mathbf{z} \\
\text{Solve} & W\mathbf{x}_1 & = & \mathbf{w} \\
\text{Form} & \mathbf{x}_\bullet & = & \mathbf{z} - \hat{R}\mathbf{x}_1
\end{array}$$

- In detail

$$\begin{array}{lrcll}
\text{Solve} & T_i\mathbf{z}_i & = & \mathbf{r}_i & i = 1, \ldots, r \\
\text{Form} & \mathbf{w} & = & \mathbf{r}_1 - \displaystyle\sum_{i=1}^{r} C_i\mathbf{z}_i & \\
\text{Solve} & W\mathbf{x}_1 & = & \mathbf{w} & \\
\text{Form} & \mathbf{x}_i & = & \mathbf{z}_i - \hat{R}_i\mathbf{x}_1 & i = 1, \ldots, r
\end{array}$$

- Looks parallel

Solving $B^T\mathbf{y} = \mathbf{d}$

- The system $\begin{bmatrix} S^T & R^T \\ C^T & T^T \end{bmatrix} \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_\bullet \end{bmatrix} = \begin{bmatrix} \mathbf{d}_1 \\ \mathbf{d}_\bullet \end{bmatrix}$ may be solved as

$$\begin{array}{lrcl} \text{Form} & \mathbf{w} & = & \mathbf{d}_1 - \hat{R}^T\mathbf{d}_\bullet \\ \text{Solve} & W^T\mathbf{y}_1 & = & \mathbf{w} \\ \text{Form} & \mathbf{z} & = & \mathbf{d}_\bullet - C^T\mathbf{y}_1 \\ \text{Solve} & T^T\mathbf{y}_\bullet & = & \mathbf{z} \end{array}$$

- In detail

$$\begin{array}{lrcll} \text{Form} & \mathbf{w} & = & \mathbf{d}_1 - \sum_{i=1}^{r} \hat{R}_i^T\mathbf{d}_i & \\ \text{Solve} & W^T\mathbf{y}_1 & = & \mathbf{w} & \\ \text{Form} & \mathbf{z}_i & = & \mathbf{d}_i - C_i^T\mathbf{y}_1 & i = 1, \ldots, r \\ \text{Solve} & T_i^T\mathbf{y}_i & = & \mathbf{z}_i & i = 1, \ldots, r \end{array}$$

- Looks parallel

Forming $\mathbf{z} = N^T\mathbf{y}$

- The matrix N has structure

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

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

$$\begin{array}{lrcl} \text{Form} & \mathbf{z}_0 & = & N_{00}^T\mathbf{y}_0 \\ \text{Form} & \mathbf{z}_i & = & N_{0i}^T\mathbf{y}_0 + N_{ii}^T\mathbf{y}_i \quad i = 1, \dots, r \end{array}$$

- Looks parallel

Factorising B

- General structure

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

- For $i = 1, \dots, r$
  - Identify $\begin{bmatrix} R_i & T_i \end{bmatrix}$ from B_{ii}
  - Form $\hat{R}_i = T_i^{-1} R_i$
  - Form $W_i = S_i - C_i \hat{R}_i$
- Factorise $W = \begin{bmatrix} S_0 & W_1 & \dots & W_r \end{bmatrix}$
- Looks parallel

Updating B

- Each simplex iteration $\mathbf{a}_q$ replaces column p of B

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

- Decision
  - Leave B^{-1} as "black box" and use traditional simplex update?
  - Update structure of B and update T, $\hat{R}$, C, S, W and factorisations of T and W?
- Chose latter
- Novel serial update techniques for W

Results: Solution time speed-up

Comparison of

- Serial solver on general problem (S)
- Exploiting 16-block structure in serial (S16)
- Exploiting 16-block structure in parallel (P16)

Problem	Rows	Columns	S16 vs S	P16 vs S16	P16 vs S
80BAU3B	2263	9799	0.4	1.0	0.4
FIT2P	3001	13525	1.3	1.1	1.5
PEROLD	626	1376	0.5	1.0	0.5
STOCFOR2	2158	2031	1.3	0.8	1.0
STOCFOR3	16676	15695	2.0	1.2	2.4
CRE-B	9649	72447	0.3	1.1	0.4
CRE-C	3069	3678	0.8	1.2	1.0
KEN-13	28633	42659	1.1	1.0	1.1
KEN-18	105128	154699	1.3	0.9	1.2
OSA-30	4351	100024	1.2	1.1	1.3
OSA-60	10281	232966	0.9	1.1	1.0
PDS-20	33875	105728	1.1	0.9	0.9
PDS-40	66844	212859	0.8	1.2	1.0
STORMG2-27	14441	34114	0.7	0.8	0.5
STORMG2-125	66186	157496	0.9	0.5	0.5
NEOS1	131581	1892	2.6	1.1	2.9
Average			1.0	1.0	1.0

Using two AMD Opteron (2378) quad-core processors and 16GiB of RAM

Results: Why are they not better? (1)

Code efficiency

- Code is very efficient in serial: competitive with Clp
- Inefficient code would speed up better, but never "catch up"

RHS structure fully exploited

- For $B\mathbf{x} = \mathbf{r}$: $\mathbf{r}$ is usually nonzero only in one block so

$$T_i\mathbf{z}_i = \mathbf{r}_i,\ i = 1, \ldots, r \text{ is usually } T_q\mathbf{z}_q = \mathbf{r}_q, \text{ some } q$$

- For $B^T\mathbf{y} = \mathbf{d}$: $\mathbf{d}$ is usually nonzero only in one block so

$$\mathbf{w} = \mathbf{d}_1 - \sum_{i=1}^{r} \hat{R}_i^T\mathbf{d}_i \text{ is usually } \mathbf{w} = -\hat{R}_p^T\mathbf{d}_p, \text{ some } p$$

Results: Why are they not better? (2)

Deduced BALP structure may be poor

- $B = \begin{bmatrix} S & C \\ R & T \end{bmatrix}$ has significant for some problems
- Operations with $W = S - CT^{-1}R$ are serial

Shared memory access overhead

- Typically only one calculation for each data access
- Computation is memory-bound **in serial**
- AMD Opteron 2378 has only 2 memory channels per processor

Observations

- BALP structure is attractive in theory
- Shared memory computation is memory bound
- Distributed memory revised simplex for large BALP is more promising

  Lubin and H (2012)
- Looking to parallelise operations with W
- Hypergraph partitioning gives smaller border

  Aykanat *et al.* (2004)
- Awaiting results using

  Two Intel Sandybridge octo-core processors

  4 memory channels per processor