High performance simplex solvers for linear programming problems

Julian Hall[1] Qi Huangfu[2] Miles Lubin[3]

[1]School of Mathematics, University of Edinburgh

[2]FICO

[3]MIT

Google, Paris

11 September 2015

High performance simplex solvers: Overview

Talk

- A little mathematics
- Some algorithms
- Mainly numerical linear algebra

Content

- Background
- Exploiting hyper-sparsity
- Exploiting parallelism
- Conclusions

Linear programming (LP)

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

Background

- Fundamental model in optimal decision-making
- Solution techniques
  - Simplex method (1947)
  - Interior point methods (1984)
- Large problems have
  - 10^3–$10^{\cancel{7}8}$ variables
  - 10^3–$10^{\cancel{7}8}$ constraints
- Matrix A is (usually) sparse

Example

STAIR: 356 rows, 467 columns and 3856 nonzeros

Solving LP problems

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

Partitioned LP

- Let $\mathcal{B} \cup \mathcal{N}$ be a **partition** of the variable set
  - Let A be partitioned as $\begin{bmatrix} B & N \end{bmatrix}$ with nonsingular **basis matrix** B
  - Let $\boldsymbol{c}$ be partitioned as $\begin{bmatrix} \boldsymbol{c}_B \\ \boldsymbol{c}_N \end{bmatrix}$
- Partitioned LP is

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

Solving LP problems

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

Reduced LP

- Equations yield

$$\boldsymbol{x}_B = \widehat{\boldsymbol{b}} - \widehat{N}\boldsymbol{x}_N \quad \text{where} \quad \widehat{N} = B^{-1}N \quad \text{and} \quad \widehat{\boldsymbol{b}} = B^{-1}\boldsymbol{b}$$

- Eliminate $\boldsymbol{x}_B$ from the objective to yield the **reduced LP**

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

where

$$\widehat{f} = \boldsymbol{c}_B^T \widehat{\boldsymbol{b}} \quad \text{and} \quad \widehat{\boldsymbol{c}}^T = \boldsymbol{c}_N^T - \boldsymbol{c}_B^T B^{-1} N$$

Solving LP problems

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

Sufficient optimality conditions

For $\boldsymbol{x}_N = \boldsymbol{0}$ sufficient optimality conditions are

- Primal feasibility $\widehat{\boldsymbol{b}} \geq \boldsymbol{0}$
- Dual feasibility $\widehat{\boldsymbol{c}} \geq \boldsymbol{0}$
  f cannot be reduced by increasing any component of $\boldsymbol{x}_N$ from zero

Simplex algorithm: concept

- Represent the reduced LP in a tableau
- Find a primal and dual feasible partition $\mathcal{B} \cup \mathcal{N}$

	$\mathcal{N}$	RHS
$\mathcal{B}$	$\widehat{N}$	$\widehat{\boldsymbol{b}}$
	$\widehat{\boldsymbol{c}}^T$	

Simplex algorithm: Primal or dual?

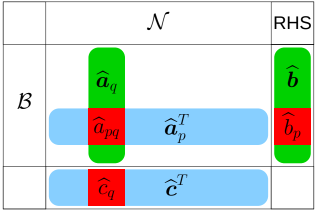
Primal simplex algorithm

- Traditional variant
  - Assume primal feasibility $\widehat{\boldsymbol{b}} \geq \mathbf{0}$
  - Seek dual feasibility $\widehat{\boldsymbol{c}} \geq \mathbf{0}$
- Solution generally not primal feasible when (primal) LP is tightened

Dual simplex algorithm

- Preferred variant
  - Assume dual feasibility $\widehat{\boldsymbol{c}} \geq \mathbf{0}$
  - Seek primal feasibility $\widehat{\boldsymbol{b}} \geq \mathbf{0}$
- Easier to get dual feasibility
- More progress in many iterations
- Solution dual feasible when LP is tightened

Simplex algorithm: Each iteration



Dual algorithm: Assume $\widehat{\boldsymbol{c}} \geq \mathbf{0}$ Seek $\widehat{\boldsymbol{b}} \geq \mathbf{0}$

Scan $\widehat{b}_i$, $i \in \mathcal{B}$, for a good candidate p to leave $\mathcal{B}$ CHUZR

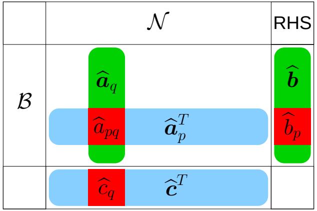
Scan $\widehat{c}_j/\widehat{a}_{pj}$, $j \in \mathcal{N}$, for a good candidate q to leave $\mathcal{N}$ CHUZC

Update: Exchange p and q between $\mathcal{B}$ and $\mathcal{N}$

Update $\widehat{\boldsymbol{b}} := \widehat{\boldsymbol{b}} - \theta_p \widehat{\boldsymbol{a}}_q$ $\theta_p = \widehat{b}_p/\widehat{a}_{pq}$ UPDATE-PRIMAL

Update $\widehat{\boldsymbol{c}}_N^T := \widehat{\boldsymbol{c}}_N^T - \theta_d \widehat{\boldsymbol{a}}_p^T$ $\theta_d = \widehat{c}_q/\widehat{a}_{pq}$ UPDATE-DUAL

Standard simplex method (SSM): Computation



Major computational component

Update of tableau:

$$\widehat{N} := \widehat{N} - \frac{1}{\widehat{a}_{pq}} \widehat{\boldsymbol{a}}_q \widehat{\boldsymbol{a}}_p^T$$

where $\widehat{N} = B^{-1}N$

- Hopelessly inefficient for sparse LP problems
- Prohibitively expensive for large LP problems

Revised simplex method (RSM): Computation

	$\mathcal{N}$	RHS
$\mathcal{B}$	$\widehat{\boldsymbol{a}}_q$ $\widehat{a}_{pq}$ $\widehat{\boldsymbol{a}}_p^T$	$\widehat{\boldsymbol{b}}$ $\widehat{b}_p$
	$\widehat{c}_q$ $\widehat{\boldsymbol{c}}^T$	

Major computational components

$\boldsymbol{\pi}_p^T = \boldsymbol{e}_p^T B^{-1}$ BTRAN $\qquad$ $\widehat{\boldsymbol{a}}_p^T = \boldsymbol{\pi}_p^T N$ PRICE

$\widehat{\boldsymbol{a}}_q = B^{-1}\boldsymbol{a}_q$ FTRAN $\qquad$ Invert B INVERT

Don't form B^{-1}!

- If B is sparse then B^{-1} is generally dense
- INVERT: form sparsity-preserving decomposition $B = LU$ to operate with B^{-1}

Exploiting hyper-sparsity

Exploiting hyper-sparsity in the revised simplex method

Recall: major computational components

- BTRAN: Solve $B^T\boldsymbol{\pi}_p = \boldsymbol{e}_p$
- PRICE: Form $\widehat{\boldsymbol{a}}_p^T = \boldsymbol{\pi}_p^T N$
- FTRAN: Solve $B\,\widehat{\boldsymbol{a}}_q = \boldsymbol{a}_q$

Phenomenon of hyper-sparsity

- Vectors $\boldsymbol{\pi}_p$, $\widehat{\boldsymbol{a}}_p^T$ and $\widehat{\boldsymbol{a}}_q$ may be sparse
- Why?

Because B^{-1} is sparse

- So?

Exploiting hyper-sparsity: Representing B^{-1}

- **Recall:** `INVERT` forms sparsity-preserving decomposition $B = LU$
  - Can use this to solve $B\boldsymbol{x} = \boldsymbol{r}$ using column-wise foward/backward substitution
  - Many columns are trivial
- Remove the trivial columns to represent B^{-1} by the eta file $\{p_k, \mu_k, \boldsymbol{\eta}_k\}_{k=1}^{K}$
- Derived directly from the results of Gaussian elimination
  - The pivots μ_k are in rows p_k
  - $\boldsymbol{\eta}_k$ are the eta vectors
  - $K \ll 2m$ is common
- Operating with the eta file $\equiv$ Column-wise foward/backward substitution

Exploiting hyper-sparsity: When solving $Bx = \boldsymbol{r}$

Traditional technique transforms $\boldsymbol{r}$ into $\boldsymbol{x}$

```
do k = 1, K
```

$$
\begin{aligned}
r_{p_k} &:= r_{p_k}/\mu_k \\
\boldsymbol{r} &:= \boldsymbol{r} - r_{p_k}\boldsymbol{\eta}_k
\end{aligned}
$$

```
end do
```

Exploiting hyper-sparsity: When solving $Bx = \boldsymbol{r}$

When $\boldsymbol{r}$ is sparse skip $\boldsymbol{\eta}_k$ if r_{p_k} is zero

```
do k = 1, K
    if (r_{p_k} .ne. 0) then
        r_{p_k} := r_{p_k}/μ_k
        r := r − r_{p_k} η_k
    end if
end do
```

- When $\boldsymbol{x}$ is sparse, the dominant cost is the test for zero
- Requires efficient identification of vectors $\boldsymbol{\eta}_k$ to be applied

Gilbert and Peierls (1988)
H and McKinnon (1998–2005)

Exploiting hyper-sparsity: When solving $B^T \boldsymbol{x} = \boldsymbol{r}$

Traditional technique transforms $\boldsymbol{r}$ into $\boldsymbol{x}$

```
do k = K, 1
    r_{p_k} := (r_{p_k} - r^T η_k)/μ_k
end do
```

$$
\begin{array}{l}
\texttt{do } k = K, 1 \\
\quad r_{p_k} := (r_{p_k} - \boldsymbol{r}^T \boldsymbol{\eta}_k)/\mu_k \\
\texttt{end do}
\end{array}
$$

- When $\boldsymbol{x}$ is sparse most $\boldsymbol{r}^T \boldsymbol{\eta}_k$ are zero
- No way to exploit hyper-sparsity properly with “column-wise” eta file
- After `INVERT`: Form a “row-wise” copy of the eta file
- Pass row-wise eta file to hyper-sparse forward solution code

H and McKinnon (1998–2005)

Speedup in total solution time and computational components

Problem	Dimension	Solution	$B^{-1}\boldsymbol{r}_F$	$\boldsymbol{r}_B^T B^{-1}$	$\boldsymbol{r}_\pi^T N$
80bau3b	2262	3.34	5.13	3.51	6.06
fit2p	3000	1.75	1.30	12.22	13.47
stocfor3	16675	1.85	1.14	7.26	7.61
dcp2	32388	5.32	8.24	6.21	6.20
ken-11	14694	22.84	98.04	27.22	66.36
ken-13	28632	12.12	104.09	12.87	17.60
ken-18	105127	15.27	263.94	13.91	19.92
pds-06	9881	17.48	24.07	21.58	28.18
pds-10	16558	10.36	11.24	16.60	17.55
pds-20	33874	10.35	5.96	14.33	15.40

H and McKinnon (1998–2005)
[Won COAP best paper prize for 2005]

Exploiting parallelism

Parallelising the simplex method: Background

Data parallel standard simplex method

- Good parallel efficiency *was* achieved
- Only relevant for dense LP problems

Data parallel revised simplex method

- Only immediate parallelism is in forming $\pi_p^T N$
- When $n \gg m$ 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

Parallelising the dual revised simplex method: Overview

Single iteration parallelism for general LP

- Pure dual revised simplex
- **Data parallelism:** Form $\boldsymbol{\pi}_p^T N$
- **Task parallelism:** Identify serial computation which can be overlapped

Multiple iteration parallelism for general LP

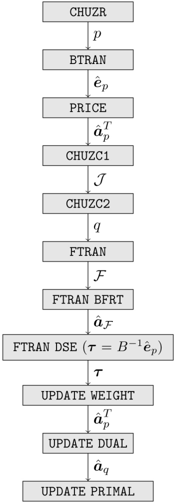
- Dual revised simplex with minor iterations of dual standard simplex
- **Data parallelism:** Form $\boldsymbol{\pi}_p^T N$ and update (slice of) dual standard simplex tableau
- **Task parallelism:** Identify serial computation which can be overlapped

Data parallelism for stochastic LP

- Pure dual revised simplex for column-linked block angular LP problems
- **Data parallelism:** Solve $B^T\boldsymbol{\pi} = \boldsymbol{e}_p$, $B\widehat{\boldsymbol{a}}_q = \boldsymbol{a}_q$ and form $\boldsymbol{\pi}_p^T N$

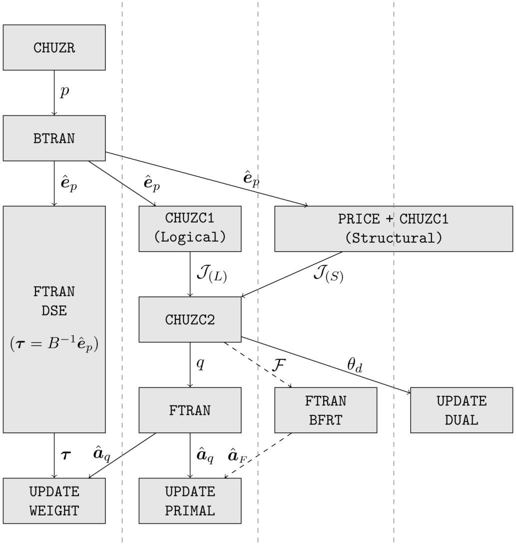
Single iteration parallelism

Single iteration parallelism: Dual revised simplex method



- Computational components appear sequential
- Each has highly-tuned sparsity-exploiting serial implementation
- Exploit “slack” in data dependencies

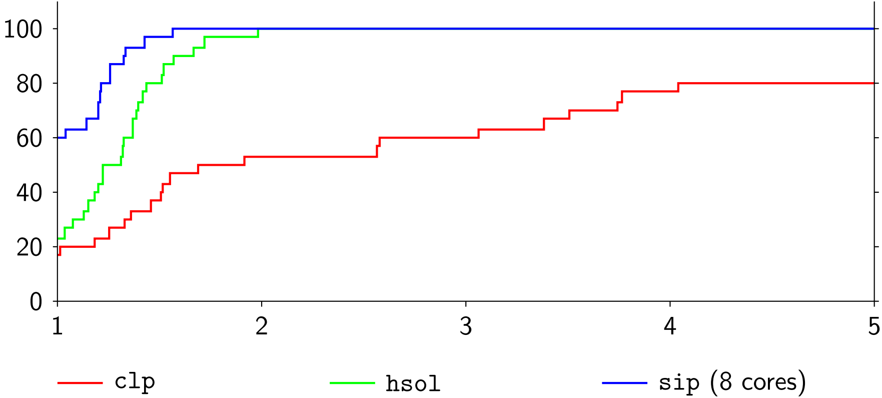
Single iteration parallelism: Computational scheme



- Parallel PRICE to form $\hat{\boldsymbol{a}}_p^T = \boldsymbol{\pi}_p^T N$
- Other computational components serial
- Overlap any independent calculations
- Only four worthwhile threads unless $n \gg m$ so PRICE dominates
- More than Bixby and Martin (2000)
- Better than Forrest (2012)

Huangfu and H (2014)

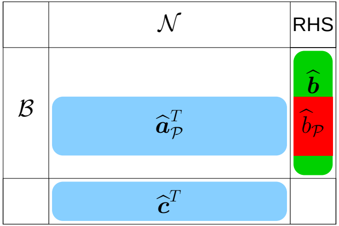
Single iteration parallelism: clp vs hsol vs sip



Multiple iteration parallelism

Multiple iteration parallelism

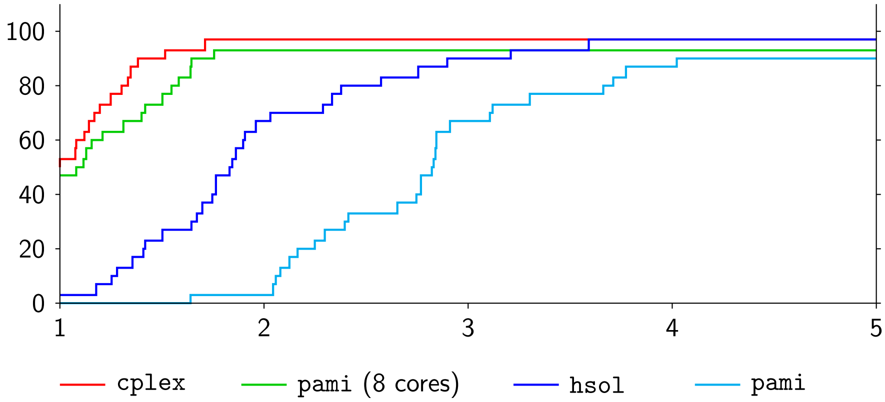
- `sip` has too little work to be performed in parallel to get good speedup
- Perform standard dual simplex minor iterations for rows in set $\mathcal{P}$ ($|\mathcal{P}| \ll m$)
- Suggested by Rosander (1975) but never implemented efficiently *in serial*



- Task-parallel multiple `BTRAN` to form $\boldsymbol{\pi}_{\mathcal{P}} = B^{-1}\boldsymbol{e}_{\mathcal{P}}$
- Data-parallel `PRICE` to form $\widehat{\boldsymbol{a}}_p^T$ (as required)
- Task-parallel multiple `FTRAN` for primal, dual and weight updates

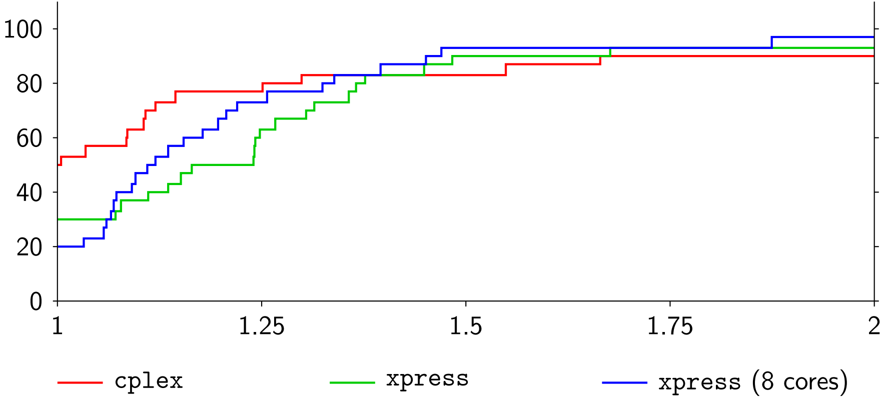
Huangfu and H (2011–2014)

Multiple iteration parallelism: cplex vs pami vs hsol



- pami is less efficient than hsol in serial
- pami speedup more than compensates
- pami performance approaching cplex

Multiple iteration parallelism: cplex vs xpress



- pami ideas incorporated in FICO Xpress (Huangfu 2014)

Data parallelism for stochastic LPs

Stochastic MIP problems: General

Two-stage stochastic LPs have column-linked block angular structure

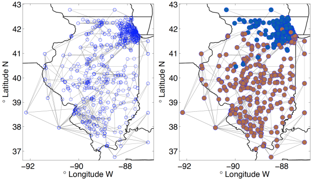
$$
\begin{array}{rcccccccccccc}
\text{minimize} & \boldsymbol{c}_0^T\boldsymbol{x}_0 & + & \boldsymbol{c}_1^T\boldsymbol{x}_1 & + & \boldsymbol{c}_2^T\boldsymbol{x}_2 & + & \dots & + & \boldsymbol{c}_N^T\boldsymbol{x}_N & & \\
\text{subject to} & A\boldsymbol{x}_0 & & & & & & & & & = & \boldsymbol{b}_0 \\
& T_1\boldsymbol{x}_0 & + & W_1\boldsymbol{x}_1 & & & & & & & = & \boldsymbol{b}_1 \\
& T_2\boldsymbol{x}_0 & & & + & W_2\boldsymbol{x}_2 & & & & & = & \boldsymbol{b}_2 \\
& \vdots & & & & & & \ddots & & & & \vdots \\
& T_N\boldsymbol{x}_0 & & & & & & & + & W_N\boldsymbol{x}_N & = & \boldsymbol{b}_N \\
& \boldsymbol{x}_0 \geq \boldsymbol{0} & & \boldsymbol{x}_1 \geq \boldsymbol{0} & & \boldsymbol{x}_2 \geq \boldsymbol{0} & & \dots & & \boldsymbol{x}_N \geq \boldsymbol{0} & &
\end{array}
$$

- Variables $\boldsymbol{x}_0 \in \mathbb{R}^{n_0}$ are **first stage** decisions
- Variables $\boldsymbol{x}_i \in \mathbb{R}^{n_i}$ for $i = 1, \dots, N$ are **second stage** decisions
  Each corresponds to a **scenario** which occurs with modelled probability
- The objective is the expected cost of the decisions
- In stochastic MIP problems, some/all decisions are discrete

Stochastic MIP problems: For Argonne

- Power systems optimization project at Argonne
- Integer second-stage decisions
- Stochasticity from wind generation
- Initial experiments carried out using model problem
- Number of scenarios increases with refinement of probability distribution sampling
- Solution via branch-and-bound
    - Solve root using parallel IPM solver `PIPS`

      Lubin, Petra *et al.* (2011)
    - Solve nodes using parallel dual simplex solver `PIPS-S`



Stochastic MIP problems: General

Convenient to permute the LP thus:

$$
\begin{array}{rcccccccccccc}
\text{minimize} & \boldsymbol{c}_1^T\boldsymbol{x}_1 & + & \boldsymbol{c}_2^T\boldsymbol{x}_2 & + & \dots & + & \boldsymbol{c}_N^T\boldsymbol{x}_N & + & \boldsymbol{c}_0^T\boldsymbol{x}_0 & & \\
\text{subject to} & W_1\boldsymbol{x}_1 & & & & & & & + & T_1\boldsymbol{x}_0 & = & \boldsymbol{b}_1 \\
 & & & W_2\boldsymbol{x}_2 & & & & & + & T_2\boldsymbol{x}_0 & = & \boldsymbol{b}_2 \\
 & & & & & \ddots & & & \vdots & & & \vdots \\
 & & & & & & & W_N\boldsymbol{x}_N & + & T_N\boldsymbol{x}_0 & = & \boldsymbol{b}_N \\
 & & & & & & & & & A\boldsymbol{x}_0 & = & \boldsymbol{b}_0 \\
 & \boldsymbol{x}_1 \geq \boldsymbol{0} & & \boldsymbol{x}_2 \geq \boldsymbol{0} & & \dots & & \boldsymbol{x}_N \geq \boldsymbol{0} & & \boldsymbol{x}_0 \geq \boldsymbol{0} & &
\end{array}
$$

Exploiting problem structure

- Inversion of the basis matrix B is key to revised simplex efficiency
- For column-linked BALP problems

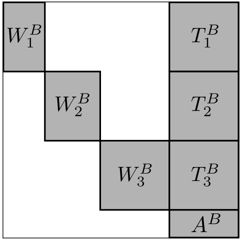
$$B = \begin{bmatrix} W_1^B & & & T_1^B \\ & \ddots & & \vdots \\ & & W_N^B & T_N^B \\ & & & A^B \end{bmatrix}$$

- W_i^B are columns corresponding to n_i^B basic variables in scenario i
- $\begin{bmatrix} T_1^B \\ \vdots \\ T_N^B \\ A^B \end{bmatrix}$ are columns corresponding to n_0^B basic first stage decisions

Exploiting problem structure

- Inversion of the basis matrix B is key to revised simplex efficiency
- For column-linked BALP problems

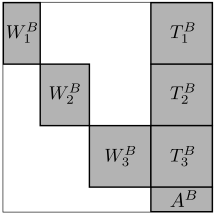
$$B = \begin{bmatrix} W_1^B & & & T_1^B \\ & \ddots & & \vdots \\ & & W_N^B & T_N^B \\ & & & A^B \end{bmatrix}$$



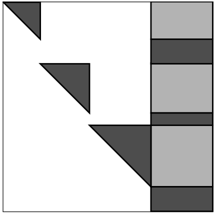
- B is nonsingular so
  - W_i^B are "tall": full column rank
  - $\begin{bmatrix} W_i^B & T_i^B \end{bmatrix}$ are "wide": full row rank
  - A^B is "wide": full row rank

- Scope for parallel inversion is immediate and well known

Exploiting problem structure

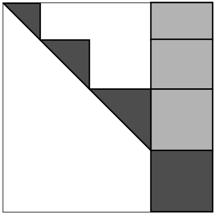


- Eliminate sub-diagonal entries in each W_i^B (independently)



- Apply elimination operations to each T_i^B (independently)

- Accumulate non-pivoted rows from the W_i^B with A^B and complete elimination



Parallel distributed-memory simplex for large-scale stochastic LP problems

Scope for parallelism

- Parallel Gaussian elimination yields **block LU** decomposition of B
- Scope for parallelism in block forward and block backward substitution
- Scope for parallelism in `PRICE`

Implementation

- Distribute problem data over processes
- Perform data-parallel `BTRAN`, `FTRAN` and `PRICE` over processes
- Used MPI

Paper: Lubin, H *et al.* (2013)

- Won COIN-OR INFORMS 2013 Cup
- Won COAP best paper prize for 2013

Results: Stochastic LP test problems

Test Problem	1st Stage n_0	m_0	2nd-Stage Scenario n_i	m_i	Nonzero Elements A	W_i	T_i
`Storm`	121	185	1,259	528	696	3,220	121
`SSN`	89	1	706	175	89	2,284	89
`UC12`	3,132	0	56,532	59,436	0	163,839	3,132
`UC24`	6,264	0	113,064	118,872	0	327,939	6,264

- `Storm` and `SSN` are publicly available
- `UC12` and `UC24` are stochastic unit commitment problems developed at Argonne
  - Aim to choose optimal on/off schedules for generators on the power grid of the state of Illinois over a 12-hour and 24-hour horizon
  - In practice each scenario corresponds to a weather simulation
    Model problem generates scenarios by normal perturbations

Zavala (2011)

Results: Baseline serial performance for large instances

Serial performance of `PIPS-S` and `clp`

Problem	Dimensions	Solver	Iterations	Time (s)	Iter/sec
`Storm`	$n =$10,313,849	`PIPS-S`	6,353,593	385,825	16.5
8,192 scen.	$m = 4,325,561$	`clp`	6,706,401	133,047	50.4
`SSN`	$n = 5,783,651$	`PIPS-S`	1,025,279	58,425	17.5
8,192 scen.	$m = 1,433,601$	`clp`	1,175,282	12,619	93.1
`UC12`	$n = 1,812,156$	`PIPS-S`	1,968,400	236,219	8.3
32 scen.	$m = 1,901,952$	`clp`	2,474,175	39,722	62.3
`UC24`	$n = 1,815,288$	`PIPS-S`	2,142,962	543,272	3.9
16 scen.	$m = 1,901,952$	`clp`	2,441,374	41,708	58.5

Results: On Fusion cluster

Speed-up of PIPS-S relative to 1-core PIPS-S and 1-core clp

Cores	Storm	SSN	UC12	UC24
1	1.0	1.0	1.0	1.0
4	3.6	3.5	2.7	3.0
8	7.3	7.5	6.1	5.3
16	13.6	15.1	8.5	8.9
32	24.6	30.3	14.5	
clp	8.5	6.5	2.4	0.7

Results: On Fusion cluster - larger instances

	Storm	SSN	UC12	UC24
Scenarios	32,768	32,768	512	256
Variables	41,255,033	23,134,297	28,947,516	28,950,648
Constraints	17,301,689	5,734,401	30,431,232	30,431,232

Results: On Fusion cluster - larger instances, from an advanced basis

Speed-up of PIPS-S relative to 1-core PIPS-S and 1-core clp

Cores	Storm	SSN	UC12	UC24
1	1	1	1	1
8	15	19	7	6
16	52	45	14	12
32	117	103	26	22
64	152	181	44	41
128	202	289	60	64
256	285	383	70	80
clp	299	45	67	68

Results: On Blue Gene supercomputer - very large instance

- Instance of `UC12`
  - 8,192 scenarios
  - 463,113,276 variables
  - 486,899,712 constraints
- Requires 1 TB of RAM
  $\geq$ 1024 Blue Gene cores
- Runs from an advanced basis

Cores	Iterations	Time (h)	Iter/sec
1024	Exceeded execution time limit		
2048	82,638	6.14	3.74
4096	75,732	5.03	4.18
8192	86,439	4.67	5.14

High performance simplex solvers: Conclusions

- Use the dual simplex method
- Exploit hyper-sparsity
- Two parallel schemes for general LP problems
  - Meaningful performance improvement
  - Have led to publicised advances in a leading commercial solver
- One parallel scheme for stochastic LP problems
  - Demonstrated scalable parallel performance... for highly specialised problems... on highly specialised machines
  - Solved problems which would be intractable using commercial serial solvers
- Helped develop two really talented young researchers: Qi Huangfu and Miles Lubin

Slides: http://www.maths.ed.ac.uk/hall/Google15/

References

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

J. A. J. Hall and Q. Huangfu.
A high performance dual revised simplex solver.
In R. W. et al., editor, *PPAM 2011, Part I*, volume 7203 of *LNCS*, pages 143–151, Heidelberg, 2012. Springer.

J. A. J. Hall and K. I. M. McKinnon.
Hyper-sparsity in the revised simplex method and how to exploit it.
Computational Optimization and Applications, 32(3):259–283, December 2005.

Q. Huangfu and J. A. J. Hall.
Parallelizing the dual revised simplex method.
Technical Report ERGO-14-011, School of Mathematics, University of Edinburgh, 2014.
Submitted to Mathematical Programming Computation.

Q. Huangfu and J. A. J. Hall.
Novel update techniques for the revised simplex method.
Computational Optimization and Applications, 60(4):587–608, 2015.

M. Lubin, J. A. J. Hall, C. G. Petra, and M. Anitescu.
Parallel distributed-memory simplex for large-scale stochastic LP problems.
Computational Optimization and Applications, 55(3):571–596, 2013.