Parallelizing the dual revised simplex method

Qi Huangfu[1] Julian Hall[2]

[1]FICO

[2]School of Mathematics, University of Edinburgh

Birmingham

9 September 2016

Overview

- Background
- Two parallel schemes
  - Single iteration parallelism
  - Multiple iteration parallelism
- An open source parallel simplex solver

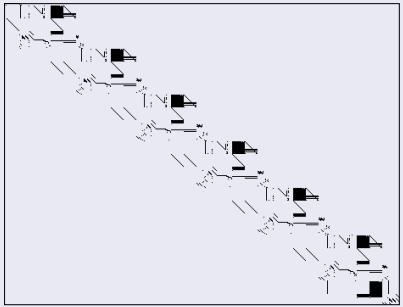
Linear programming (LP)

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

Background

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

Example



STAIR: 356 rows, 467 columns and 3856 nonzeros

Solving LP problems

$$\begin{array}{lll} \text{minimize} & f_P = \boldsymbol{c}^T\boldsymbol{x} & \\ \text{subject to} & A\boldsymbol{x} = \boldsymbol{b} \quad \boldsymbol{x} \geq \boldsymbol{0} & (P) \end{array} \qquad \begin{array}{lll} \text{maximize} & f_D = \boldsymbol{b}^T\boldsymbol{y} & \\ \text{subject to} & A^T\boldsymbol{y} + \boldsymbol{s} = \boldsymbol{c} \quad \boldsymbol{s} \geq \boldsymbol{0} & (D) \end{array}$$

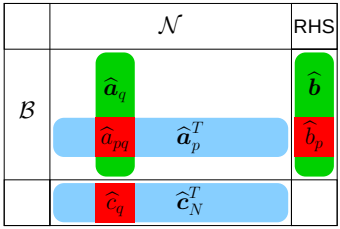
Optimality conditions

- For a partition $\mathcal{B} \cup \mathcal{N}$ of the variable set with nonsingular **basis matrix** B in

$$B\boldsymbol{x}_B + N\boldsymbol{x}_N = \boldsymbol{b} \text{ for (P)} \qquad \text{and} \qquad \begin{bmatrix} B^T \\ N^T \end{bmatrix} \boldsymbol{y} + \begin{bmatrix} \boldsymbol{s}_B \\ \boldsymbol{s}_N \end{bmatrix} = \begin{bmatrix} \boldsymbol{c}_B \\ \boldsymbol{c}_N \end{bmatrix} \text{ for (D)}$$

  with $\boldsymbol{x}_N = \boldsymbol{0}$ and $\boldsymbol{s}_B = \boldsymbol{0}$
  - Primal basic variables $\boldsymbol{x}_B$ given by $\widehat{\boldsymbol{b}} = B^{-1}\boldsymbol{b}$
  - Dual non-basic variables $\boldsymbol{s}_N$ given by $\widehat{\boldsymbol{c}}_N^T = \boldsymbol{c}_N^T - \boldsymbol{c}_B^T B^{-1} N$
- Partition is optimal if there is
  - Primal feasibility $\widehat{\boldsymbol{b}} \geq \boldsymbol{0}$
  - Dual feasibility $\widehat{\boldsymbol{c}}_N \geq \boldsymbol{0}$

Simplex algorithm: Each iteration



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

Simplex method: Computation

Standard simplex method: 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: Major computational components

$\boldsymbol{\pi}_p^T = \boldsymbol{e}_p^T B^{-1}$	BTRAN	$\widehat{\boldsymbol{a}}_p^T = \boldsymbol{\pi}_p^T N$	PRICE
$\widehat{\boldsymbol{a}}_q = B^{-1}\boldsymbol{a}_q$	FTRAN	Invert B	INVERT

Simplex method: Algorithmic enhancements

Row selection: Dual steepest edge (DSE)

- Weight $\widehat{b}_i$ by w_i: measure of $\|B^{-T}\boldsymbol{e}_i\|_2$
- Requires additional `FTRAN` but can reduce iteration count significantly

Column selection: Bound-flipping ratio test (BFRT)

- Minimizes the dual objective whilst remaining dual feasible
  - Dual variables may change sign if corresponding primal variables can flip bounds
- Requires additional `FTRAN` but can reduce iteration count significantly

Exploiting parallelism: Background

Data parallel standard simplex method

- Good parallel efficiency of $\widehat{N} := \widehat{N} - \frac{1}{\widehat{a}_{pq}} \widehat{\boldsymbol{a}}_q \widehat{\boldsymbol{a}}_p^T$ *was* achieved
- Only relevant for dense LP problems

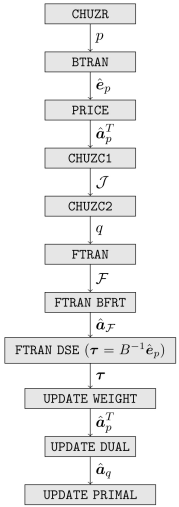
Data parallel revised simplex method

- Only immediate parallelism is in forming $\boldsymbol{\pi}_p^T N$
- When $n \gg m$ significant speed-up *was* achieved Bixby and Martin (2000)

Task parallel revised simplex method

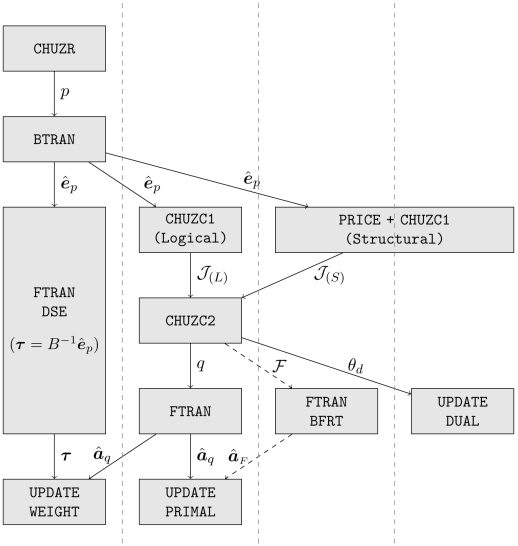
- Overlap computational components for different iterations
  Wunderling (1996), H and McKinnon (1995-2005)
- Modest speed-up *was* achieved on general sparse LP problems

Single iteration parallelism: Dual revised simplex method



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

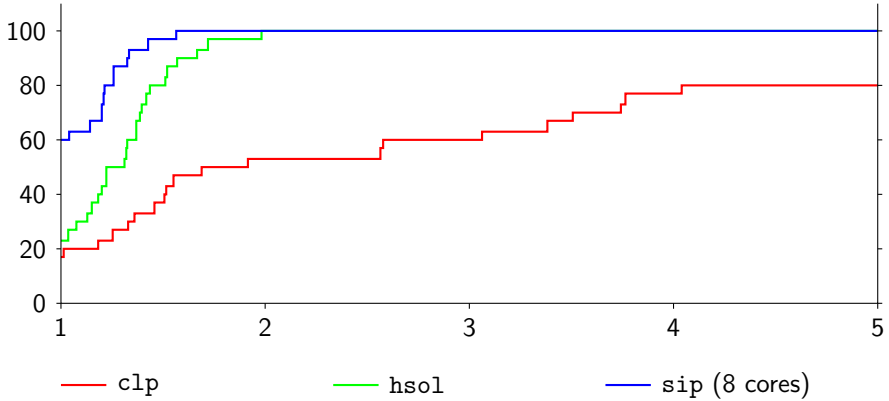
Single iteration parallelism: Computational scheme



- 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

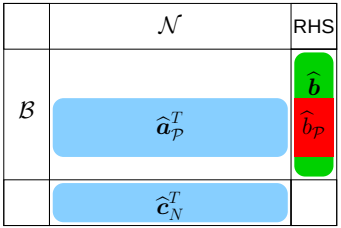


Performance on spectrum of 30 significant LP test problems

- sip on 8 cores is 1.15 times faster than hsol
- hsol (sip on 8 cores) is 2.29 (2.64) times faster than clp

Multiple iteration parallelism

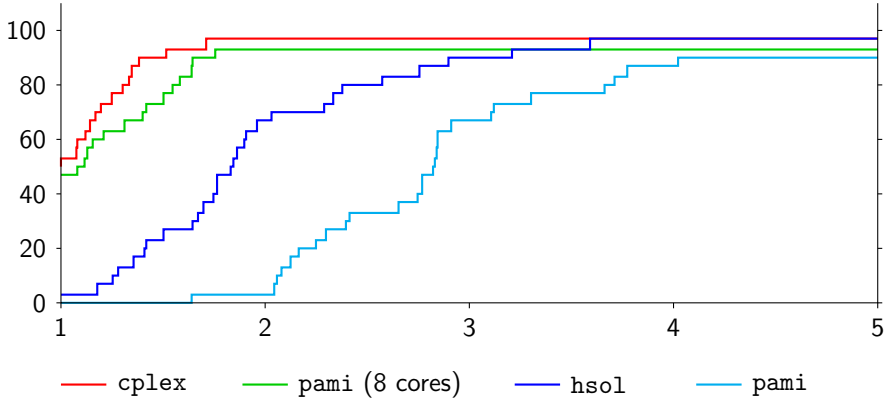
- `sip` has too little work to be performed in parallel to get good speedup
- Perform standard dual simplex minor iterations for rows in set $\mathcal{P}$ ($|\mathcal{P}| \ll m$)
- Suggested by Rosander (1975) but never implemented efficiently *in serial*



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

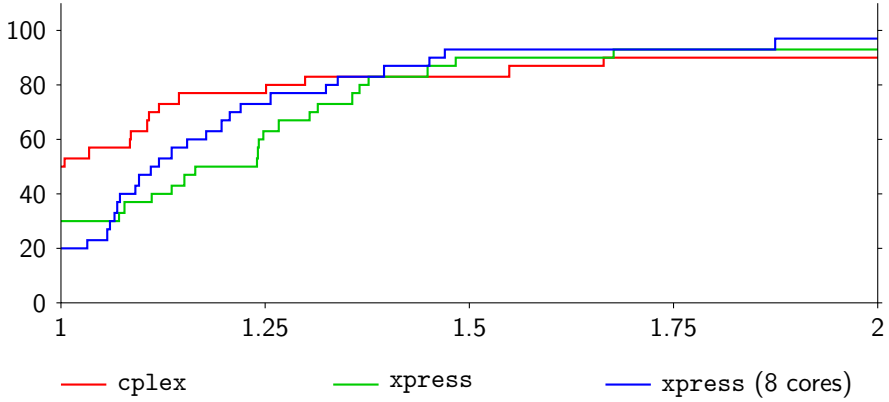
Huangfu and H (2011–2014)

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)

hsol: An open source parallel simplex solver

- hsol and its variants (sip and pami) are high performance codes
- Useful open-source resource?
  - Reliable?
  - Well-written?
  - Better than clp?
- Limitations
  - No presolve
  - No crash
  - No advanced basis start

hsol: Performance and reliability

Extended testing using 159 test problems

- 98 Netlib
- 16 Kennington
- 4 Industrial
- 41 Mittelmann

Exclude 7 which are "hard"

Reliability on 152 test problems

- hsol, sip and pami fail on cycle
- pami with 8 threads fails on stat96v1

hsol: Performance and reliability

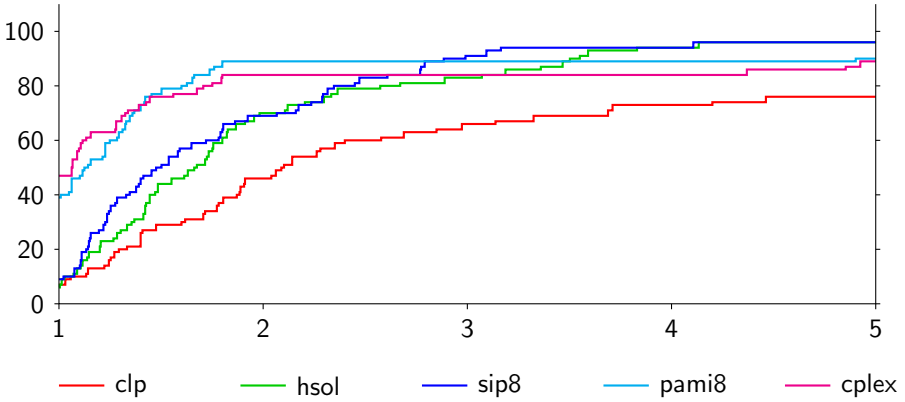
Performance

Benchmark against clp (v1.16) and cplex (v12.5)

- Dual simplex
- No presolve
- No crash

Ignore results for 82 LPs with minimum solution time below 0.1s

hsol: Performance and reliability



hsol: Addressing limitations

Presolve

Prototype implementation of a full range of presolve techniques

Galabova (2016)

Crash

- Standard crash procedures are simple to implement
- Questionable value for dual simplex:
  dual steepest edge is expensive to initialise when $B \neq I$

Advanced basis start

- Essential for hsol to be used in MIP and SLP solvers
- Only complication is dual steepest edge

Conclusions

- Two parallel schemes for general LP problems
  - Meaningful performance improvement
  - Have led to publicised advances in a leading commercial solver
- High performance open-source parallel simplex solver in preparation

Slides: http://www.maths.ed.ac.uk/hall/NLAO_16/

Paper: 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
Accepted for publication in Mathematical Programming Computation