

Promoting hyper-sparsity in the revised simplex method

Julian Hall and Qi Huangfu

School of Mathematics
University of Edinburgh

`jajhall@ed.ac.uk`

28th March 2012

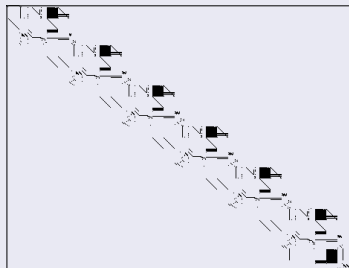
Linear programming (LP)

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

Background

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

Example



STAIR: 356 rows, 467 columns and 3856 nonzeros

Solving LP problems

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

$$\begin{array}{ll}\text{maximize} & f_D = \mathbf{b}^T \mathbf{y} \\ \text{subject to} & A^T \mathbf{y} + \mathbf{s} = \mathbf{c} \quad (D) \\ & \mathbf{s} \geq \mathbf{0}\end{array}$$

Optimality conditions

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

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

with $\mathbf{x}_N = \mathbf{0}$ and $\mathbf{s}_B = \mathbf{0}$

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

Dual simplex method: algorithm

Each iteration

Scan $\hat{b}_i$ for a good candidate p to leave $\mathcal{B}$	CHUZR
Scan $\hat{c}_j/\hat{a}_{pj}$ for a good candidate q to enter $\mathcal{B}$	CHUZYC
Update $\hat{\mathbf{c}}_N := \hat{\mathbf{c}}_N - \theta_d \hat{\mathbf{a}}_p \quad \theta_d = \hat{c}_q/\hat{a}_{pq}$	UPDATE-DUAL
Update $\hat{\mathbf{b}} := \hat{\mathbf{b}} - \theta_p \hat{\mathbf{a}}_q \quad \theta_p = \hat{b}_p/\hat{a}_{pq}$	UPDATE-PRIMAL

CHUZYC and degeneracy

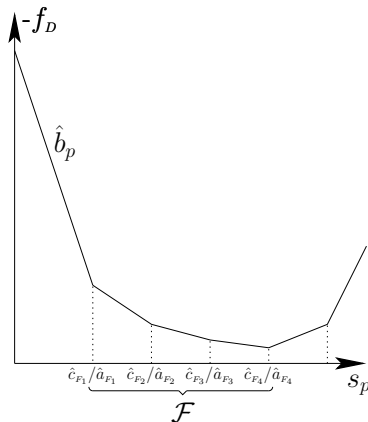
- Dual step aims to increase f_D by increasing s_p from zero
- Limit identified in CHUZYC by considering $\mathbf{s}_N = \hat{\mathbf{c}}_N - s_p \hat{\mathbf{a}}_p$
- Unless $\hat{\mathbf{c}}_N > \mathbf{0}$, limiting value θ_d of s_p may be **zero**
 - Dual step is **degenerate**
 - Dual objective is not increased
 - Can lead to stalling or cycling

Dual simplex method: bound-flipping ratio test (BFRT)

BFRT: Fourer (1994)

minimize $f_P = \mathbf{c}^T \mathbf{x}$
subject to $A\mathbf{x} = \mathbf{b} \quad \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}$

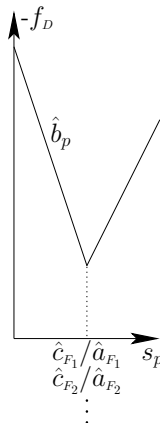
- Primal variable x_j for $j \in \mathcal{N}$ now has value l_j or u_j
- If attractive,
 - Allow dual variables s_j for $j \in \mathcal{F}$ to change sign
 - Flip the bound of x_j for $j \in \mathcal{F}$ to keep dual feasibility
- Allows multiple iterations for a single basis change



Dual simplex method: bound-flipping ratio test (BFRT)

BFRT and degeneracy

- Nodes coalesce
- Restricts/eliminates bound-flipping (by default)



Dual simplex method: cost perturbation

Cost perturbation: Clp, Koberstein (2008)

- Perturb coefficients of $\mathbf{c}$ by random ϵ *a priori*
- Ensures $\hat{\mathbf{c}}_N > \mathbf{0}$
- Ensures $\theta_d > 0$ and “spreads” degenerate node(s) in BFRT
- When optimal
 - Remove perturbation
 - Possibly perform primal simplex “clean-up”

Dual simplex method: computation

Major computational components

$$\pi_p^T = \mathbf{e}_p^T B^{-1} \quad \text{BTRAN}$$

$$\hat{\mathbf{a}}_q = B^{-1} \mathbf{a}_q \quad \text{FTRAN}$$

$$\hat{\mathbf{a}}_p^T = \pi_p^T N \quad \text{PRICE}$$

$$\text{Invert } B \quad \text{INVERT}$$

Hyper-sparsity

- Vectors $\mathbf{e}_p$ and $\mathbf{a}_q$ are **always sparse**
- B may be **highly reducible**; B^{-1} may be **sparse**
- Vectors π_p , $\hat{\mathbf{a}}_p$ and $\hat{\mathbf{a}}_q$ **may be sparse**
- Efficient implementations must exploit these features
H and McKinnon (1998–2005), Bixby (1999)
Clp, Koberstein and Suhl (2005–2008)

Promoting hyper-sparsity

- Hyper-sparse LPs have small proportions of dense tableau rows and columns
- Cannot always choose sparse attractive row
 - Density not known *a priori*
 - Other rows may be much more attractive
- Hyper-sparsity properties of optimal solution are fixed
- Is there a path to the optimal solution via hyper-sparse vertices?

Testing cost perturbation

Name	Degenerate iterations		Improvement factors		
	Without	With	Iterations	Speed	Time
pds-100	88%	0%	2.4	5.2	12.5
neos1	61%	0%	18.1	1.4	25.8
stormG2-1k	78%	0%	1.0	1.0	1.0

- Cost perturbation eliminates degeneracy
- Some dramatic reduction in iteration count
- Some dramatic improvement in iteration speed

[For ParISS: parallel simplex solver with Clp-like performance]

Increased hyper-sparsity

Percentage sparse results: without and with cost perturbation

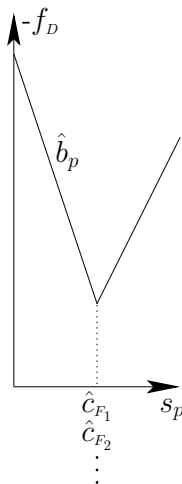
Name	Without			With		
	BTRAN	PRICE	FTRAN	BTRAN	PRICE	FTRAN
pds-100	38	73	100	97	100	100
neos1	100	70	22	100	96	42
stormG2-1k	100	100	100	100	100	100

- A result is sparse if its density ≤ 0.1
- Proportions of sparse results increase significantly
- Dense pivotal rows and columns are avoided
- Why?

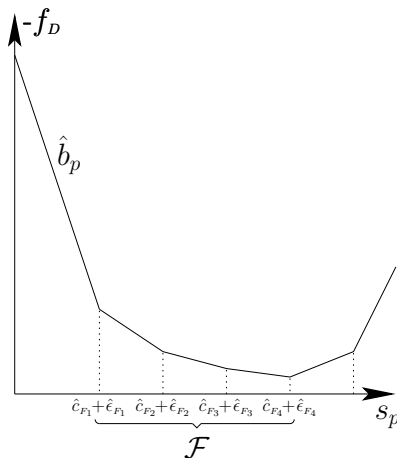
- Use network LP problems (NETGEN)
 - Varying degrees of degeneracy
 - Should be hyper-sparse
- Appealing algebraic simplicity
 - Nonzeros in pivotal rows/columns are $+1$ and -1
 - $\hat{b}_i$ is sum of b_i and (signed) values from $\mathbf{l}$ and $\mathbf{u}$
 - $\hat{c}_j$ is sum of c_j and (signed) values from $\mathbf{c}$
- Interpretation of simplex algebra via networks

Analysis: BFRT

Cost perturbation spreads degenerate node(s) in BFRT



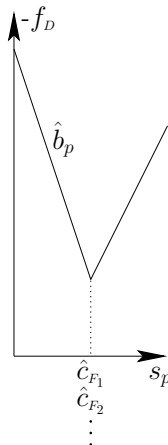
Without cost perturbation



With cost perturbation

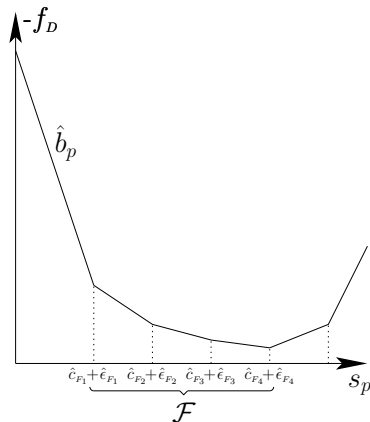
BFRT without cost perturbation

- BFRT has degenerate nodes
- Tie for column selection
- Choose on numerical grounds
Tie for network problems
- No bound flipping



BFRT with cost perturbation

- Degenerate nodes are expanded
- BFRT is well defined
 - Order of nodes is linked to sparsity
 - Denser columns tend not to be chosen
- Why?



Claim: Dense columns have larger perturbed dual values

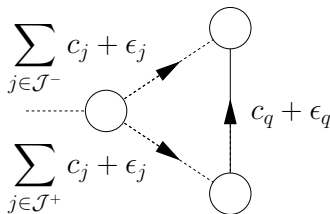
- For $q \in \mathcal{N}$ let $\mathcal{J}_q = \mathcal{J}_q^+ \cup \mathcal{J}_q^-$
index the nonzeros of $\hat{\mathbf{a}}_q$
 - $\mathcal{J}_q^+$ are indices of +1
 - $\mathcal{J}_q^-$ are indices of -1
- Dual value is

$$\hat{c}_q + \hat{\epsilon}_q = c_q + \epsilon_q + \sum_{j \in \mathcal{J}_q^+} (c_j + \epsilon_j) - \sum_{j \in \mathcal{J}_q^-} (c_j + \epsilon_j)$$

- $\hat{c}_j$ for $j \in \mathcal{F}$ are equal so are distinguished by

$$\hat{\epsilon}_q = \epsilon_q + \sum_{j \in \mathcal{J}_q^+} \epsilon_j - \sum_{j \in \mathcal{J}_q^-} \epsilon_j$$

- All values of ϵ_j are of comparable magnitude
- $\hat{\epsilon}_q$ is small only if $|\mathcal{J}_q^+| \approx |\mathcal{J}_q^-|$: unlikely if $|\mathcal{J}_q|$ is large



Cost perturbation: effect on BTRAN and PRICE

- Choosing sparser columns may lead to broader spanning trees
 - Greater sparsity in BTRAN results
- When a dense row is chosen
 - Bound flips lead to greater reduction in the primal infeasibility
 - Chosen again later rather than sooner

For NETGEN 101–150

Name	Degenerate iterations		Improvement factors		
	Without	With	Iterations	Speed	Time
Min	0%	0%	0.97	0.95	0.93
Average	32%	0%	1.14	1.59	1.80
Max	74%	1%	1.68	3.80	6.16

- Cost perturbation eliminates degeneracy
- Improvement in iterations and solution time

Results: increased hyper-sparsity

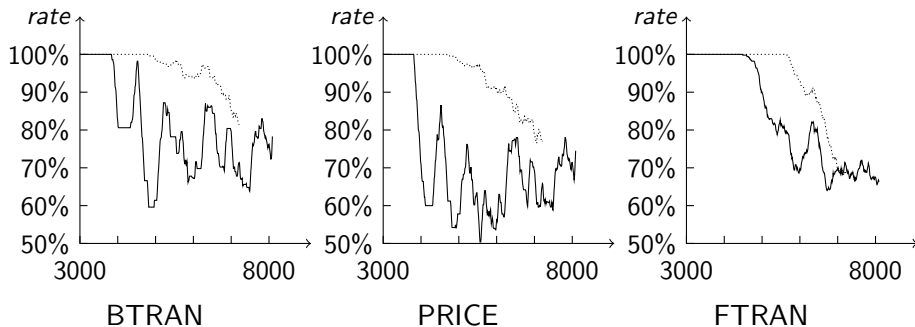
Percentage of sparse results

Name	Without cost perturbation			With cost perturbation		
	BTRAN	PRICE	FTRAN	BTRAN	PRICE	FTRAN
Min	38	33	67	81	75	77
Average	83	80	94	97	96	97
Max	99	99	99	100	99	100

What makes a result sparse?

- For BTRAN and PRICE: density ≤ 0.1
- For FTRAN: density ≤ 0.02

Results: example hyper-sparsity trends (netgen108)



- Reduction in proportion of hyper-sparse results is inevitable
- Cost perturbation (dotted line) delays this longer

Other techniques: partial CHUZR

Background

- Similar idea to partial pricing in primal
- Consider only a small proportion of the rows
- No direct efficiency gain (unlike partial price)
- Used in Clp, otherwise not published?

Promoting hyper-sparsity

- Small proportion of dense tableau rows
 - Partial CHUZR likely to avoid them...
 - ...until they dominate the attractive rows
- Should improve performance

Results: partial CHUZR (only)

For NETGEN 101–150

Name	Degenerate iterations		Improvement factors		
	Without	With	Iterations	Speed	Time
Min	0%	0%	0.92	1.08	1.06
Average	32%	23%	1.06	1.63	1.73
Max	74%	59%	1.44	3.38	4.85

- Some reduction in degeneracy
Due to randomness of row selection?
- Improvement in iterations and solution time

New technique: over-weighted CHUZR

Steepest edge: Forrest and Goldfarb (1992)

$$\text{Weight } \hat{b}_i \text{ by } \hat{w}_i = \|\mathbf{e}_i^T B^{-1}\|_2 = \sqrt{\sum_{j \in \mathcal{J}_i} [B^{-1}]_{ij}^2}$$

where $\mathcal{J}_i$ are the indices of the nonzeros in row i of B^{-1}

For network problems

- $[B^{-1}]_{ij}^2 = 1$ for $j \in \mathcal{J}_i$ so $\|\mathbf{e}_i^T B^{-1}\|_2 = \sqrt{|\mathcal{J}_i|}$
- Consider **over-weighting** using B^{-1} **row density** $|\mathcal{J}_i|$
- Novel technique corresponds to steepest edge using **1-norm**

$$\|\mathbf{e}_i^T B^{-1}\|_1 = \|\mathbf{e}_i^T B^{-1}\|_2^2 = \hat{w}_i^2$$

For general LP problems

- Computing 1-norm weight or row density is intractable
- Consider general over-weighting using $\hat{w}_i^2$

Results: over-weighted CHUZR (only)

For NETGEN 101–150

Name	Degenerate iterations		Improvement factors		
	Without	With	Iterations	Speed	Time
Min	0%	0%	0.89	1.10	1.02
Average	32%	17%	1.03	1.77	1.83
Max	74%	50%	1.56	3.96	6.20

- Some reduction in degeneracy: why?
- Improvement in iterations and solution time

Results: using all three techniques

For NETGEN 101–150

Name	Degenerate iterations		Improvement factors		
	Without	With	Iterations	Speed	Time
Min	0%	0%	0.91	1.06	0.96
Average	32%	0%	1.05	1.87	1.97
Max	74%	0%	1.64	4.20	6.87

Percentage sparse results: without and with techniques

Name	Without			With		
	BTRAN	PRICE	FTRAN	BTRAN	PRICE	FTRAN
Min	38	33	67	97	95	87
Average	83	80	94	99	99	99
Max	99	99	99	100	100	100

Results: general LP problems

Performance: without and with techniques

Name	Degenerate iterations		Improvement factors		
	Without	With	Iterations	Speed	Time
pds-100	88%	1%	1.1	7.9	8.5
neos1	61%	0%	7.7	1.4	10.8
stormG2-1k	78%	0%	0.9	1.2	1.1

Percentage sparse results: without and with techniques

Name	Without			With		
	BTRAN	PRICE	FTRAN	BTRAN	PRICE	FTRAN
pds-100	38	73	100	100	100	100
neos1	100	70	22	100	88	36
stormG2-1k	100	100	100	100	100	100

Summary and Conclusions

- Introduced technique of over-weighting
- This and other algorithmic refinements shown to have computational consequences
 - Increased the proportions of hyper-sparse results
 - Partial answer to “why?”
 - Performance has improved significantly
- Aim to extend applicability and understanding

<http://www.maths.ed.ac.uk/hall/APMOD12>