Exploiting ultra-sparsity in the revised simplex method

Julian Hall

School of Mathematics
University of Edinburgh

jajhall@ed.ac.uk

Sparse Days, CERFACS, Toulouse

28 September 2018

THE UNIVERSITY *of* EDINBURGH

LP and high performance simplex solvers

$$\text{minimize} \quad f = \boldsymbol{c}^T\boldsymbol{x} \qquad \text{subject to} \quad A\boldsymbol{x} = \boldsymbol{b} \qquad \boldsymbol{x} \geq \boldsymbol{0}$$

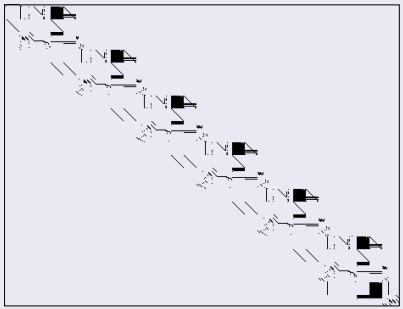
LP and high performance simplex solvers

$$\text{minimize} \quad f = \boldsymbol{c}^T\boldsymbol{x} \qquad \text{subject to} \quad A\boldsymbol{x} = \boldsymbol{b} \qquad \boldsymbol{x} \geq \boldsymbol{0}$$

Background

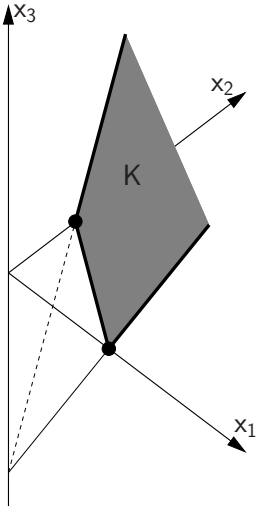
- Fundamental model in optimal decision-making
- Simplex method preferred when solving related problems
- High performance requires
  - Algorithmic tricks
  - Computational tricks

Example



STAIR: 356 rows, 467 columns and 3856 nonzeros

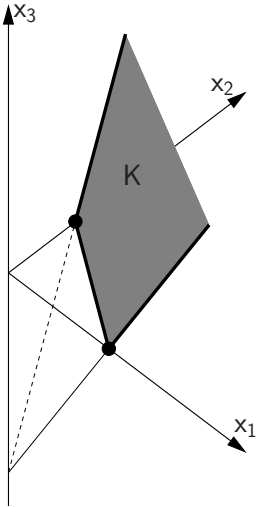
Solving LP problems: Characterizing the feasible region



minimize $f = \boldsymbol{c}^T\boldsymbol{x}$ subject to $A\boldsymbol{x} = \boldsymbol{b}$ $\boldsymbol{x} \geq \boldsymbol{0}$

- Optimal solution at a vertex of feasible region K

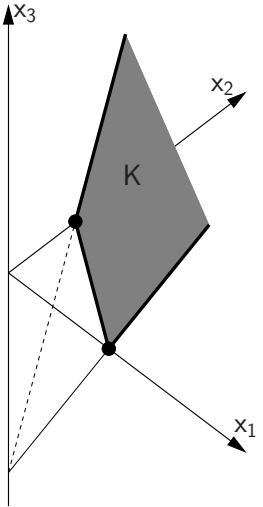
Solving LP problems: Characterizing the feasible region



minimize $f = \boldsymbol{c}^T\boldsymbol{x}$ subject to $A\boldsymbol{x} = \boldsymbol{b}$ $\boldsymbol{x} \geq \boldsymbol{0}$

- Optimal solution at a vertex of feasible region K
- Vertex characterised by partition $\mathcal{B} \cup \mathcal{N}$ of index set

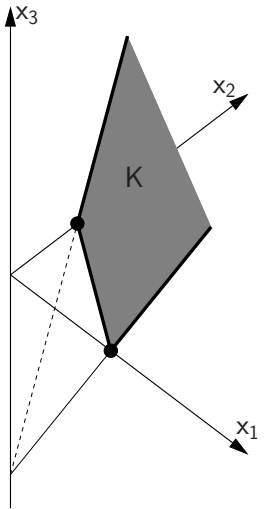
Solving LP problems: Characterizing the feasible region



minimize $f = \boldsymbol{c}^T\boldsymbol{x}$ subject to $A\boldsymbol{x} = \boldsymbol{b}$ $\boldsymbol{x} \geq \boldsymbol{0}$

- Optimal solution at a vertex of feasible region K
- Vertex characterised by partition $\mathcal{B} \cup \mathcal{N}$ of index set with
  - Nonsingular **basis matrix** B

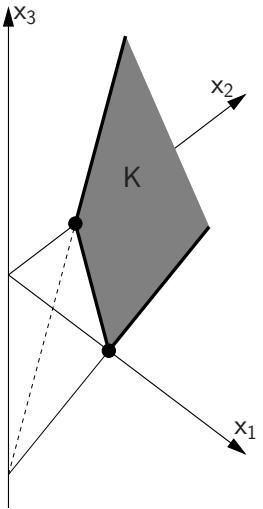
Solving LP problems: Characterizing the feasible region



minimize $f = \boldsymbol{c}^T\boldsymbol{x}$ subject to $A\boldsymbol{x} = \boldsymbol{b}$ $\boldsymbol{x} \geq \boldsymbol{0}$

- Optimal solution at a vertex of feasible region K
- Vertex characterised by partition $\mathcal{B} \cup \mathcal{N}$ of index set with
  - Nonsingular **basis matrix** B
  - Equations partitioned as $B\boldsymbol{x}_B + N\boldsymbol{x}_N = \boldsymbol{b}$

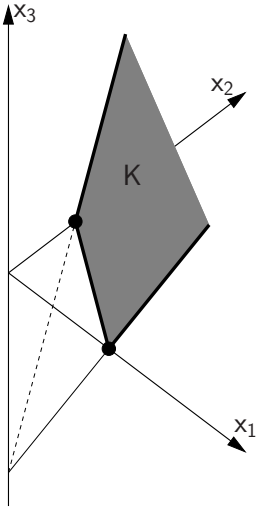
Solving LP problems: Characterizing the feasible region



minimize $f = \boldsymbol{c}^T\boldsymbol{x}$ subject to $A\boldsymbol{x} = \boldsymbol{b}$ $\boldsymbol{x} \geq \boldsymbol{0}$

- Optimal solution at a vertex of feasible region K
- Vertex characterised by partition $\mathcal{B} \cup \mathcal{N}$ of index set with
  - Nonsingular **basis matrix** B
  - Equations partitioned as $B\boldsymbol{x}_B + N\boldsymbol{x}_N = \boldsymbol{b}$
  - Vertex given by $\boldsymbol{x}_N = \boldsymbol{0}$ and $\boldsymbol{x}_B = B^{-1}\boldsymbol{b} = \widehat{\boldsymbol{b}}$

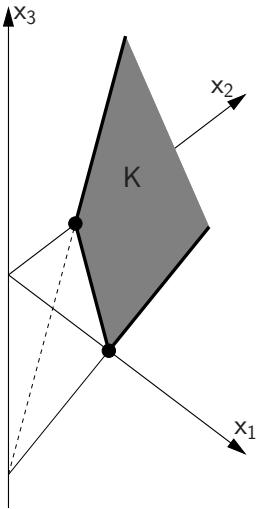
Solving LP problems: Characterizing the feasible region



minimize $f = \boldsymbol{c}^T\boldsymbol{x}$ subject to $A\boldsymbol{x} = \boldsymbol{b}$ $\boldsymbol{x} \geq \boldsymbol{0}$

- Optimal solution at a vertex of feasible region K
- Vertex characterised by partition $\mathcal{B} \cup \mathcal{N}$ of index set with
  - Nonsingular **basis matrix** B
  - Equations partitioned as $B\boldsymbol{x}_B + N\boldsymbol{x}_N = \boldsymbol{b}$
  - Vertex given by $\boldsymbol{x}_N = \boldsymbol{0}$ and $\boldsymbol{x}_B = B^{-1}\boldsymbol{b} = \widehat{\boldsymbol{b}}$
  - K given by $\boldsymbol{x}_B = \widehat{\boldsymbol{b}} - B^{-1}N\boldsymbol{x}_N$ for some $\boldsymbol{x}_N \geq \boldsymbol{0}$

Solving LP problems: Optimality conditions



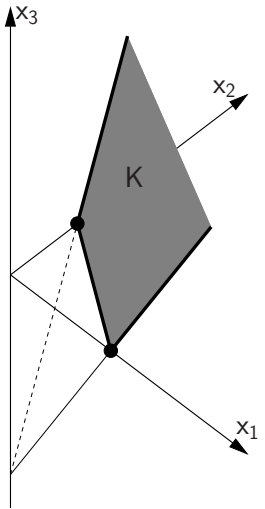
minimize $f = \boldsymbol{c}^T\boldsymbol{x}$ subject to $A\boldsymbol{x} = \boldsymbol{b}$ $\boldsymbol{x} \geq \boldsymbol{0}$

- Objective partitioned according to $\mathcal{B} \cup \mathcal{N}$ as

$$f = \boldsymbol{c}_B^T\boldsymbol{x}_B + \boldsymbol{c}_N^T\boldsymbol{x}_N = \widehat{f} + \widehat{\boldsymbol{c}}_N^T\boldsymbol{x}_N$$

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

Solving LP problems: Optimality conditions



minimize $f = \boldsymbol{c}^T\boldsymbol{x}$ subject to $A\boldsymbol{x} = \boldsymbol{b}$ $\boldsymbol{x} \geq \boldsymbol{0}$

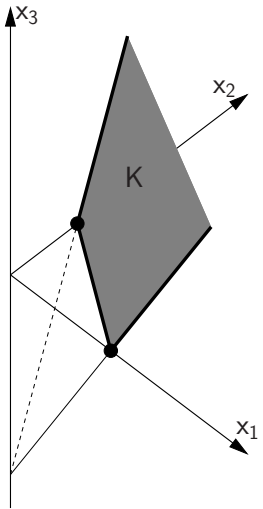
- Objective partitioned according to $\mathcal{B} \cup \mathcal{N}$ as

$$f = \boldsymbol{c}_B^T\boldsymbol{x}_B + \boldsymbol{c}_N^T\boldsymbol{x}_N = \widehat{f} + \widehat{\boldsymbol{c}}_N^T\boldsymbol{x}_N$$

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

- Partition yields an optimal solution if there is
  - Primal feasibility $\widehat{\boldsymbol{b}} \geq \boldsymbol{0}$
  - Dual feasibility $\widehat{\boldsymbol{c}}_N \geq \boldsymbol{0}$

Solving LP problems: Optimality conditions



minimize $f = \boldsymbol{c}^T\boldsymbol{x}$ subject to $A\boldsymbol{x} = \boldsymbol{b}$ $\boldsymbol{x} \geq \boldsymbol{0}$

- Objective partitioned according to $\mathcal{B} \cup \mathcal{N}$ as

$$f = \boldsymbol{c}_B^T\boldsymbol{x}_B + \boldsymbol{c}_N^T\boldsymbol{x}_N = \widehat{f} + \widehat{\boldsymbol{c}}_N^T\boldsymbol{x}_N$$

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

- Partition yields an optimal solution if there is
  - Primal feasibility $\widehat{\boldsymbol{b}} \geq \boldsymbol{0}$
  - Dual feasibility $\widehat{\boldsymbol{c}}_N \geq \boldsymbol{0}$

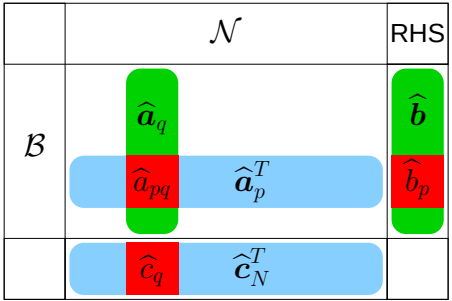
- Simplex algorithm adjusts $\mathcal{B} \cup \mathcal{N}$ until optimal

Primal simplex algorithm

Assume $\widehat{\boldsymbol{b}} \geq \mathbf{0}$ Seek $\widehat{\boldsymbol{c}}_N \geq \mathbf{0}$

Scan $\widehat{c}_j < 0$ for q to leave $\mathcal{N}$

Scan $\widehat{b}_i / \widehat{a}_{iq} > 0$ for p to leave $\mathcal{B}$



Primal simplex algorithm

Assume $\widehat{\boldsymbol{b}} \geq \mathbf{0}$ Seek $\widehat{\boldsymbol{c}}_N \geq \mathbf{0}$

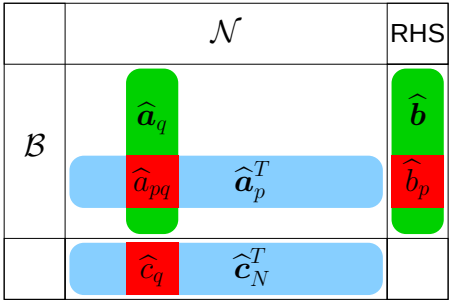
Scan $\widehat{c}_j < 0$ for q to leave $\mathcal{N}$

Scan $\widehat{b}_i/\widehat{a}_{iq} > 0$ for p to leave $\mathcal{B}$

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

Update $\widehat{\boldsymbol{b}} := \widehat{\boldsymbol{b}} - \alpha_P \widehat{\boldsymbol{a}}_q \qquad \alpha_P = \widehat{b}_p/\widehat{a}_{pq}$

Update $\widehat{\boldsymbol{c}}_N^T := \widehat{\boldsymbol{c}}_N^T + \alpha_D \widehat{\boldsymbol{a}}_p^T \qquad \alpha_D = -\widehat{c}_q/\widehat{a}_{pq}$



Primal simplex algorithm

Assume $\widehat{\boldsymbol{b}} \geq \mathbf{0}$ Seek $\widehat{\boldsymbol{c}}_N \geq \mathbf{0}$

Scan $\widehat{c}_j < 0$ for q to leave $\mathcal{N}$

Scan $\widehat{b}_i/\widehat{a}_{iq} > 0$ for p to leave $\mathcal{B}$

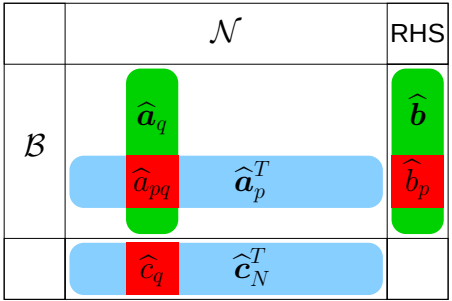
Update: Exchange p and q between $\mathcal{B}$ and $\mathcal{N}$

Update $\widehat{\boldsymbol{b}} := \widehat{\boldsymbol{b}} - \alpha_P \widehat{\boldsymbol{a}}_q \qquad \alpha_P = \widehat{b}_p/\widehat{a}_{pq}$

Update $\widehat{\boldsymbol{c}}_N^T := \widehat{\boldsymbol{c}}_N^T + \alpha_D \widehat{\boldsymbol{a}}_p^T \qquad \alpha_D = -\widehat{c}_q/\widehat{a}_{pq}$

Data required

- Pivotal row $\widehat{\boldsymbol{a}}_p^T = \boldsymbol{e}_p^T B^{-1} N$
- Pivotal column $\widehat{\boldsymbol{a}}_q = B^{-1}\boldsymbol{a}_q$



Primal simplex algorithm

Assume $\widehat{\boldsymbol{b}} \geq \mathbf{0}$ Seek $\widehat{\boldsymbol{c}}_N \geq \mathbf{0}$

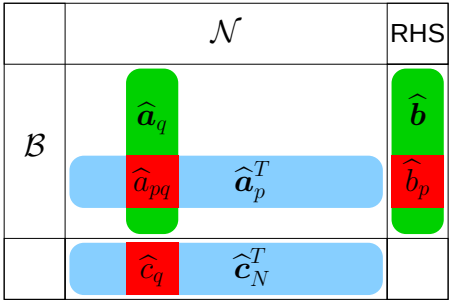
Scan $\widehat{c}_j < 0$ for q to leave $\mathcal{N}$

Scan $\widehat{b}_i/\widehat{a}_{iq} > 0$ for p to leave $\mathcal{B}$

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

Update $\widehat{\boldsymbol{b}} := \widehat{\boldsymbol{b}} - \alpha_P \widehat{\boldsymbol{a}}_q \qquad \alpha_P = \widehat{b}_p/\widehat{a}_{pq}$

Update $\widehat{\boldsymbol{c}}_N^T := \widehat{\boldsymbol{c}}_N^T + \alpha_D \widehat{\boldsymbol{a}}_p^T \qquad \alpha_D = -\widehat{c}_q/\widehat{a}_{pq}$



Computation

Pivotal row via $B^T\boldsymbol{\pi}_p = \boldsymbol{e}_p$ BTRAN and $\widehat{\boldsymbol{a}}_p^T = \boldsymbol{\pi}_p^T N$ PRICE

Pivotal column via $B\widehat{\boldsymbol{a}}_q = \boldsymbol{a}_q$ FTRAN Represent B^{-1} INVERT

Update B^{-1} exploiting $\bar{B} = B + (\boldsymbol{a}_q - B\boldsymbol{e}_p)\boldsymbol{e}_p^T$ UPDATE-BASIS

Simplex method: Hyper-sparsity

Recall: major computational components

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

Simplex method: Hyper-sparsity

Recall: major computational components

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

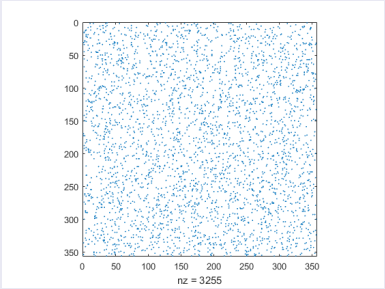
Phenomenon of hyper-sparsity

- Vectors $\boldsymbol{e}_p$ and $\boldsymbol{a}_q$ are sparse
  - In BTRAN, $\boldsymbol{\pi}_p$ is a row of B^{-1}
  - In FTRAN, $\widehat{\boldsymbol{a}}_q$ is a linear combination of a few columns of B^{-1}
- Results $\boldsymbol{\pi}_p$ and $\widehat{\boldsymbol{a}}_q$ may be sparse—because B^{-1} is sparse
  - In PRICE, $\widehat{\boldsymbol{a}}_p^T$ is a linear combination of a few rows of N

Hyper-sparsity

Inverse of a sparse matrix and solution of $B\boldsymbol{x} = \boldsymbol{r}$

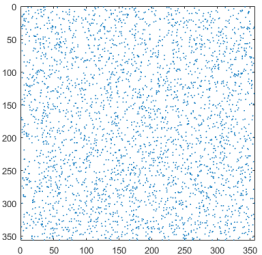
Random sparse matrix B
$m = 356$ and density 2.5%



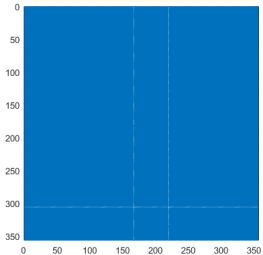
Hyper-sparsity

Inverse of a sparse matrix and solution of $B\boldsymbol{x} = \boldsymbol{r}$

Random sparse matrix B
$m = 356$ and density 2.5%



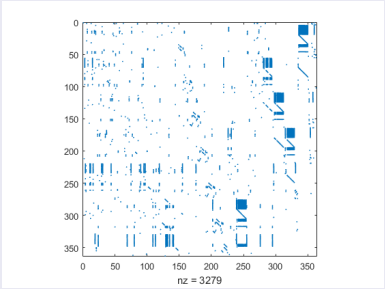
B^{-1} has density of 99%



Hyper-sparsity

Inverse of a sparse matrix and solution of $B\boldsymbol{x} = \boldsymbol{r}$

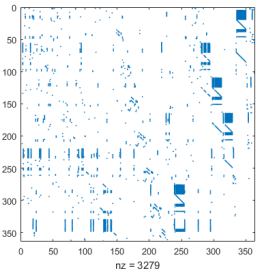
Optimal B for LP problem `stair`
$m = 356$ and density 2.5%



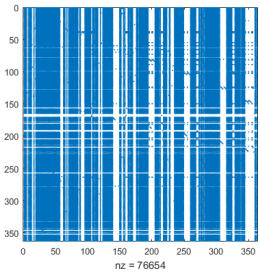
Hyper-sparsity

Inverse of a sparse matrix and solution of $B\boldsymbol{x} = \boldsymbol{r}$

Optimal B for LP problem `stair`
$m = 356$ and density 2.5%



B^{-1} has density of 58%, so $B^{-1}\boldsymbol{r}$ is typically dense

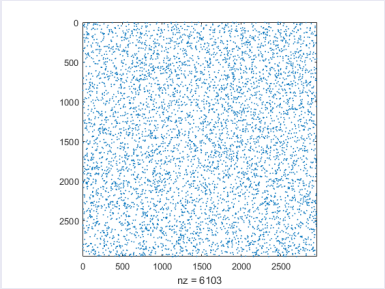


Hyper-sparsity

Inverse of a sparse matrix and solution of $B\boldsymbol{x} = \boldsymbol{r}$

Random matrix

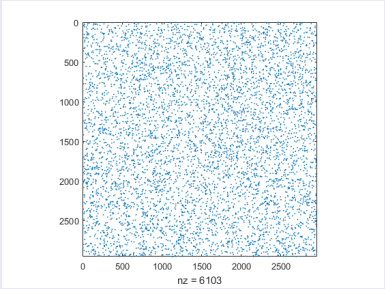
$m = 2953$ and density 0.07%



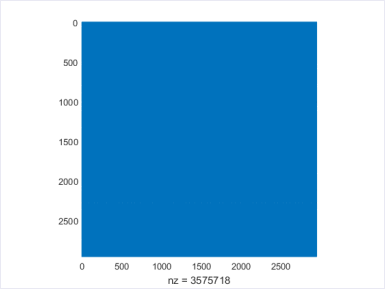
Hyper-sparsity

Inverse of a sparse matrix and solution of $B\boldsymbol{x} = \boldsymbol{r}$

Random matrix
$m = 2953$ and density 0.07%



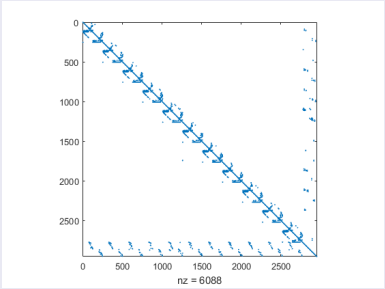
B^{-1} has density of 100%



Hyper-sparsity

Inverse of a sparse matrix and solution of $Bx = r$

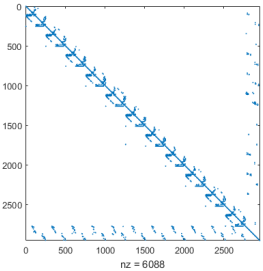
Optimal B for LP problem `pds-02`
$m = 2953$ and density 0.07%



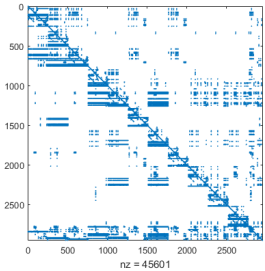
Hyper-sparsity

Inverse of a sparse matrix and solution of $B\boldsymbol{x} = \boldsymbol{r}$

Optimal B for LP problem pds-02
$m = 2953$ and density 0.07%



B^{-1} has density of 0.52%, so $B^{-1}\boldsymbol{r}$ is typically sparse—when $\boldsymbol{r}$ is sparse



Hyper-sparsity

FTRAN: Form $\widehat{\boldsymbol{a}}_q = B^{-1}\boldsymbol{a}_q$

- Triangular solves with $B = LU$
- Few columns of L and U are used to form $\widehat{\boldsymbol{a}}_q$ Gilbert and Peierls (1988)

Hyper-sparsity

FTRAN: Form $\widehat{\boldsymbol{a}}_q = B^{-1}\boldsymbol{a}_q$

- Triangular solves with $B = LU$
- Few columns of L and U are used to form $\widehat{\boldsymbol{a}}_q$ Gilbert and Peierls (1988)

BTRAN: Form $\boldsymbol{\pi}_p = B^{-T}\boldsymbol{e}_p$

- Transposed triangular solves
- Store L and U row-wise (also)
- Use FTRAN code

Hyper-sparsity

FTRAN: Form $\widehat{\boldsymbol{a}}_q = B^{-1}\boldsymbol{a}_q$

- Triangular solves with $B = LU$
- Few columns of L and U are used to form $\widehat{\boldsymbol{a}}_q$ Gilbert and Peierls (1988)

BTRAN: Form $\boldsymbol{\pi}_p = B^{-T}\boldsymbol{e}_p$

- Transposed triangular solves
- Store L and U row-wise (also)
- Use FTRAN code

PRICE: Form $\widehat{\boldsymbol{a}}_p^T = \boldsymbol{\pi}_p^T N$

- Hyper-sparsity: $\boldsymbol{\pi}_p^T$ is sparse
- Store N row-wise
- Form $\widehat{\boldsymbol{a}}_p^T$ as a combination of rows of N for nonzeros in $\boldsymbol{\pi}_p^T$

H and McKinnon (1998–2005)
COAP best paper prize (2005)

Hyper-sparsity: PRICE

- PRICE forms $\widehat{\boldsymbol{a}}_p^T = \boldsymbol{\pi}_p^T N$

Hyper-sparsity: PRICE

- PRICE forms $\widehat{\boldsymbol{a}}_p^T = \boldsymbol{\pi}_p^T N$
- Simplifying notation, this is

$$\boldsymbol{y}^T = \boldsymbol{\pi}^T A$$

Hyper-sparsity: PRICE

- PRICE forms $\widehat{\boldsymbol{a}}_p^T = \boldsymbol{\pi}_p^T N$
- Simplifying notation, this is

$$\boldsymbol{y}^T = \boldsymbol{\pi}^T A = \sum_{\pi_i \neq 0} \pi_i \times \boldsymbol{a}_i^T \quad \text{where} \quad A = \begin{bmatrix} \boldsymbol{a}_1^T \\ \boldsymbol{a}_2^T \\ \vdots \\ \boldsymbol{a}_m^T \end{bmatrix}$$

Hyper-sparsity: PRICE

- PRICE forms $\widehat{\boldsymbol{a}}_p^T = \boldsymbol{\pi}_p^T N$
- Simplifying notation, this is

$$\boldsymbol{y}^T = \boldsymbol{\pi}^T A = \sum_{\pi_i \neq 0} \pi_i \times \boldsymbol{a}_i^T \quad \text{where} \quad A = \begin{bmatrix} \boldsymbol{a}_1^T \\ \boldsymbol{a}_2^T \\ \vdots \\ \boldsymbol{a}_m^T \end{bmatrix}$$

- In detail
  - Start with zero full-length vector $\boldsymbol{y}^T$

Hyper-sparsity: PRICE

- PRICE forms $\widehat{\boldsymbol{a}}_p^T = \boldsymbol{\pi}_p^T N$
- Simplifying notation, this is

$$\boldsymbol{y}^T = \boldsymbol{\pi}^T A = \sum_{\pi_i \neq 0} \pi_i \times \boldsymbol{a}_i^T \quad \text{where} \quad A = \begin{bmatrix} \boldsymbol{a}_1^T \\ \boldsymbol{a}_2^T \\ \vdots \\ \boldsymbol{a}_m^T \end{bmatrix}$$

- In detail
  - Start with zero full-length vector $\boldsymbol{y}^T$
  - For each nonzero π_i

$$\boldsymbol{y}^T := \boldsymbol{y}^T + \pi_i \boldsymbol{a}_i^T$$

Hyper-sparsity: PRICE

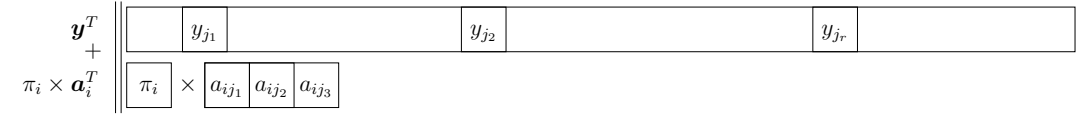
- PRICE forms $\widehat{\boldsymbol{a}}_p^T = \boldsymbol{\pi}_p^T N$
- Simplifying notation, this is

$$\boldsymbol{y}^T = \boldsymbol{\pi}^T A = \sum_{\pi_i \neq 0} \pi_i \times \boldsymbol{a}_i^T \quad \text{where} \quad A = \begin{bmatrix} \boldsymbol{a}_1^T \\ \boldsymbol{a}_2^T \\ \vdots \\ \boldsymbol{a}_m^T \end{bmatrix}$$

- In detail
  - Start with zero full-length vector $\boldsymbol{y}^T$
  - For each nonzero π_i

$$\boldsymbol{y}^T := \boldsymbol{y}^T + \pi_i \boldsymbol{a}_i^T$$

  - Pass through each entry in (packed) sparse row $\boldsymbol{a}_i^T$



PRICE: What more can be done?

PRICE: What more can be done?

How sparse is π?

- Problem-dependent!
- Extreme case
  - m =1,143,250
  - Average π has 82 nonzeros
  - One nonzero for every 14,000 entries!

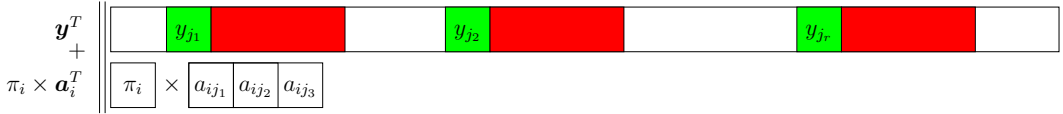
PRICE: What more can be done?

How sparse is π?

- Problem-dependent!
- Extreme case
  - m =1,143,250
  - Average π has 82 nonzeros
  - One nonzero for every 14,000 entries!

Cache inefficiency

- Read nonzeros of π - in cache anyway?
- Read row $\boldsymbol{a}_i^T$ - contiguous
- Read entries in $\boldsymbol{y}^T$- in cache?



PRICE: Ultra-sparsity (1)

Aim: Avoid full-length `double` $\boldsymbol{y}^T$

- Pack nonzeros in `double` $\boldsymbol{v}$
- Store full-length array of pointers $\boldsymbol{p}$ into $\boldsymbol{v}$

PRICE: Ultra-sparsity (1)

Aim: Avoid full-length `double` $\boldsymbol{y}^T$

- Pack nonzeros in `double` $\boldsymbol{v}$
- Store full-length array of pointers $\boldsymbol{p}$ into $\boldsymbol{v}$
  - Up to 255 pointers using `unsigned char` - 8 times smaller
  - Up to 65535 pointers using `unsigned short` - 4 times smaller
  - Switch to standard hyper-sparse technique if $\boldsymbol{y}^T$ fills in too much
- Hope that $\boldsymbol{v}$ fits into cache

PRICE: Ultra-sparsity (1)

Aim: Avoid full-length `double` $\boldsymbol{y}^T$

- Pack nonzeros in `double` $\boldsymbol{v}$
- Store full-length array of pointers $\boldsymbol{p}$ into $\boldsymbol{v}$
  - Up to 255 pointers using `unsigned char` - 8 times smaller
  - Up to 65535 pointers using `unsigned short` - 4 times smaller
  - Switch to standard hyper-sparse technique if $\boldsymbol{y}^T$ fills in too much
- Hope that $\boldsymbol{v}$ fits into cache

Does it work?

- Yes! (Poirrier: ISMP 2018)
- No! (Mészáros: FICO)

My experiments showed no performance improvement on amenable problems

PRICE: Ultra-sparsity (2)

Aim: Avoid any full-length $\boldsymbol{y}^T$!

- Store nonzeros and indices in C++ `map<int, double>`
- Use C++ utilities to `find` nonzeros
- Search cost is ln $|\boldsymbol{y}|$

PRICE: Ultra-sparsity (2)

Aim: Avoid any full-length $\boldsymbol{y}^T$!

- Store nonzeros and indices in C++ `map<int, double>`
- Use C++ utilities to `find` nonzeros
- Search cost is ln $|\boldsymbol{y}|$

Does it work?

My experiments showed no performance improvement on amenable problems

PRICE: Ultra-sparsity

Why doesn't it work

Example problem

- m =1,143,250; n =1,367,843
- Average $\boldsymbol{\pi}$ has 82 nonzeros
- Average $\boldsymbol{y}^T = \boldsymbol{\pi}^T A$ has 175 nonzeros

CPU cache structure

- L1d cache: 32K
- L1i cache: 32K
- L2 cache: 256K
- L3 cache: 8192K

PRICE: Ultra-sparsity

Why doesn't it work

Example problem

- m =1,143,250; n =1,367,843
- Average $\boldsymbol{\pi}$ has 82 nonzeros
- Average $\boldsymbol{y}^T = \boldsymbol{\pi}^T A$ has 175 nonzeros

CPU cache structure

- L1d cache: 32K
- L1i cache: 32K
- L2 cache: 256K
- L3 cache: 8192K

How might it work?

- Better implementation
- Better understanding of memory access, cache occupancy and CPU activity
- Force $\boldsymbol{y}$ pointer array to remain in cache
- Ideas?

Conclusions

Ultra-sparsity

- Exists
- Leads to cache inefficiency
- Hard to exploit!

Conclusions

Ultra-sparsity

- Exists
- Leads to cache inefficiency
- Hard to exploit!

Slides:
http://www.maths.ed.ac.uk/hall/SparseDays18

Code:
https://github.com/ERGO-Code/HiGHS


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.
Mathematical Programming Computation, 10(1):119–142, 2018.