

Fast solution of LP problems

Julian Hall and Ivet Galabova

School of Mathematics
University of Edinburgh

`jajhall@ed.ac.uk`; `I.L.Galabova@sms.ed.ac.uk`

Zuse Institute Berlin

10 October 2017



THE UNIVERSITY
of EDINBURGH

- Dual simplex method
- Three novel update techniques
- High performance dual simplex solver
- Open-source linear optimization package `h2gmp`
- Idiot crash and fast solution of LP problems

Solving LP problems: Optimality conditions at a vertex

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

- A **vertex** of the **feasible region** corresponds to a partition $\mathcal{B} \cup \mathcal{N}$ of $\{1, \dots, n\}$ into **basic** components $\mathbf{x}_B$, and zero-valued **nonbasic** components $\mathbf{x}_N$
- Equations partitioned as $B\mathbf{x}_B + N\mathbf{x}_N = \mathbf{b}$ with nonsingular **basis matrix** B
- Substituting $\mathbf{x}_B = B^{-1}(\mathbf{b} - N\mathbf{x}_N) = \hat{\mathbf{b}} - B^{-1}N\mathbf{x}_N$ into the objective

$$f = \mathbf{c}_B^T \mathbf{x}_B + \mathbf{c}_N^T \mathbf{x}_N \quad \text{gives} \quad f = \hat{f} + \hat{\mathbf{c}}_N^T \mathbf{x}_N$$

where $\hat{f} = \mathbf{c}_B^T \hat{\mathbf{b}}$ and $\hat{\mathbf{c}}_N^T = \mathbf{c}_N^T - \mathbf{c}_B^T B^{-1}N$ is the vector of **reduced costs**

- Vertex is optimal if there is
 - Primal feasibility $\hat{\mathbf{b}} \geq \mathbf{0}$
 - Dual feasibility $\hat{\mathbf{c}}_N \geq \mathbf{0}$

Dual simplex algorithm

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

Scan $\hat{b}_i < 0$ for p to leave $\mathcal{B}$

Scan $\hat{c}_j / \hat{a}_{pj} < 0$ for q to leave $\mathcal{N}$

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

Update $\hat{\mathbf{b}} := \hat{\mathbf{b}} - \alpha_P \hat{\mathbf{a}}_q$ $\alpha_P = \hat{b}_p / \hat{a}_{pq}$

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

	$\mathcal{N}$		RHS
$\mathcal{B}$	$\hat{\mathbf{a}}_q$		$\hat{\mathbf{b}}$
	$\hat{a}_{pq}$	$\hat{\mathbf{a}}_p^T$	$\hat{b}_p$
	$\hat{c}_q$	$\hat{\mathbf{c}}_N^T$	

Data required

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

Why does it work?

Objective improves by $\frac{\hat{b}_p \times \hat{c}_q}{\hat{a}_{pq}}$ each iteration

Simplex method: Computation

Standard simplex method (SSM): Major computational component

	$\mathcal{N}$	RHS
$\mathcal{B}$	$\hat{N}$	$\hat{\mathbf{b}}$
	$\hat{\mathbf{c}}_N^T$	

Update of tableau: $\hat{N} := \hat{N} - \frac{1}{\hat{a}_{pq}} \hat{\mathbf{a}}_q \hat{\mathbf{a}}_p^T$

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

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

Revised simplex method (RSM): Major computational components

Pivotal row via $B^T \pi_p = \mathbf{e}_p$ **BTRAN** and $\hat{\mathbf{a}}_p^T = \pi_p^T N$ **PRICE**

Pivotal column via $B \hat{\mathbf{a}}_q = \mathbf{a}_q$ **FTRAN** Represent B^{-1} **INVERT**

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

Simplex method: Product form update (PFI) for B

- Given

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

- Take B out as a factor on the left

$$\bar{B} = B[I + (B^{-1}\mathbf{a}_q - \mathbf{e}_p)\mathbf{e}_p^T] = BE$$

$$\text{where } E = I + (\hat{\mathbf{a}}_q - \mathbf{e}_p)\mathbf{e}_p^T = \begin{bmatrix} 1 & & \eta_1 & & \\ & \ddots & \vdots & & \\ & & \mu & & \\ & & \vdots & \ddots & \\ & & \eta_m & & 1 \end{bmatrix}$$

$\mu = \hat{a}_{pq}$ is the **pivot**; remaining entries in $\hat{\mathbf{a}}_q$ form the **eta vector** $\boldsymbol{\eta}$

- Can solve $\bar{B}\mathbf{x} = \mathbf{r}$ as $B\mathbf{x} = \mathbf{r}$ then $\mathbf{x} := E^{-1}\mathbf{x}$ as

$$x_p := x_p / \mu \quad \text{then} \quad \mathbf{x} := \mathbf{x} - x_p \boldsymbol{\eta}$$

Dantzig and Orchard-Hays (1954)

Simplex method: Forrest-Tomlin update (FT) for B

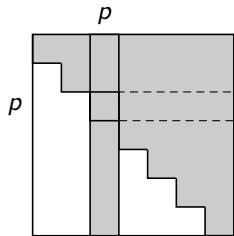
- Given

$$\bar{B} = B + (\mathbf{a}_q - B\mathbf{e}_p)\mathbf{e}_p^T \quad \text{where (wlog)} \quad B = LU$$

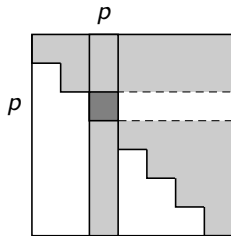
- Multiply $\bar{B}$ by L^{-1} to give

$$L^{-1}\bar{B} = U + (L^{-1}\mathbf{a}_q - U\mathbf{e}_p)\mathbf{e}_p^T = U + (\tilde{\mathbf{a}}_q - \mathbf{u}_p)\mathbf{e}_p^T = U' \quad (\text{a})$$

- Eliminate entries in row p to give $R^{-1}U' = \bar{U}$ (b)



(a) Spiked upper: U'



(b) After elimination: $\bar{U}$

- Yields $\bar{B} = LR\bar{U}$
- Compute $\tilde{\mathbf{a}}_q$ when forming $\hat{\mathbf{a}}_q$
- Represent R like E
- FT more efficient than PFI with respect to sparsity

Forrest and Tomlin (1972)

Novel update techniques for the revised simplex method: 1

Alternative product form update (APF)

- **Recall:** Column p of B is replaced by $\mathbf{a}_q$ to give $\bar{B} = B + (\mathbf{a}_q - B\mathbf{e}_p)\mathbf{e}_p^T$
 - Traditional PFI takes B out as a factor on the left so $\bar{B} = BE$
- **Idea:** Why not take it out on the right!

$$\bar{B} = [I + (\mathbf{a}_q - B\mathbf{e}_p)\mathbf{e}_p^T B^{-1}]B = TB$$

$$\text{where } T = I + (\mathbf{a}_q - \mathbf{a}_{p'})\hat{\mathbf{e}}_p^T$$

- T is formed of known data and readily invertible (like E for PFI)
Naturally compute $\hat{\mathbf{e}}_p$ when solving $B^T\boldsymbol{\pi}_p = \mathbf{e}_p$
- **But:** Is this useful?

Novel update techniques for the revised simplex method: 2

Middle product form update (MPF)

- **Recall:** Column p of B is replaced by $\mathbf{a}_q$ to give $\bar{B} = B + (\mathbf{a}_q - B\mathbf{e}_p)\mathbf{e}_p^T$
- **Idea:** Substitute $B = LU$ and take factors L on the left and U on the right!

$$\begin{aligned}\bar{B} &= LU + (\mathbf{a}_q - B\mathbf{e}_p)\mathbf{e}_p^T \\ &= LU + LL^{-1}(\mathbf{a}_q - B\mathbf{e}_p)\mathbf{e}_p^T U^{-1}U \\ &= L[I + (\tilde{\mathbf{a}}_q - U\mathbf{e}_p)\tilde{\mathbf{e}}_p^T]U \\ &= LMU \quad \text{where } M = I + (\tilde{\mathbf{a}}_q - \mathbf{u}_p)\tilde{\mathbf{e}}_p^T\end{aligned}$$

- M is formed of known data and readily invertible (like E for PFI)
Naturally compute $\tilde{\mathbf{a}}_q$ when solving $B\hat{\mathbf{a}}_q = \mathbf{a}_q$ and $\tilde{\mathbf{e}}_p$ when solving $B^T\pi_p = \mathbf{e}_p$
- **But:** Is this useful?

Novel update techniques for the revised simplex method: 3

Forrest-Tomlin update

- **Recall:** Column p of B is replaced by $\mathbf{a}_q$ to give $\bar{B} = B + (\mathbf{a}_q - B\mathbf{e}_p)\mathbf{e}_p^T$
- **Idea:** Substitute $B = LU$ and take factor L on the left

$$\bar{B} = L[U + (L^{-1}\mathbf{a}_q - U\mathbf{e}_p)\mathbf{e}_p^T] = LU'$$

where $U' = U + (\tilde{\mathbf{a}}_q - \mathbf{u}_p)\mathbf{e}_p^T$ is “spiked” upper triangular and then eliminate

Collective Forrest-Tomlin (CFT) update

- Update Forrest-Tomlin representation of B after multiple basis changes
- Don't have data to perform a sequence of standard FT updates
- Have to perform elimination corresponding to multiple spikes
- **But:** Is this useful?

Huangfu and H (2013)

Simplex method: Multiple updates

- Suppose $B_0 = L_0 U_0$ and k updates are performed to obtain B_k
 - Pivot in rows $\{p_i\}_{i=1}^k$
 - Introduce columns $\{\mathbf{a}_{q_i}\}_{i=1}^k$

- PFI generalizes as

$$B_k = B_0 E_1 E_2 \dots E_k$$

- FT generalizes as

$$B_k = L_0 R_1 R_2 \dots R_k U_k$$

- APF generalizes as

$$B_k = T_k \dots T_2 T_1 B$$

- MPF generalizes as

$$B_k = L_0 M_1 M_2 \dots M_k U_0$$

Eventually more computationally efficient or numerically prudent to reinvert some B_k

Novel update techniques for the revised simplex method: Results

- Test environment
 - 30 representative LP problems from standard test sets
 - Same sequence of basis changes for all update techniques
- Geometric mean time for operations to form and use update data

Update	PFI	FT	APF	MPF	CFT
Mean	1.00	0.30	0.95	0.45	0.29

Conclusions:

- FT much better than PFI
- APF little better than PFI
- MPF closer to FT than PFI
- CFT as good as FT

Overview

- Written in C++ to study parallel simplex
- Dual simplex with steepest edge and BFRT
- Forrest-Tomlin update
 - **complex** and **inherently serial**
 - **efficient** and **numerically stable**

Concept

- Exploit limited task and data parallelism in standard dual RSM iterations (sip)
- Exploit greater task and data parallelism via minor iterations of dual SSM (pami)
- Test-bed for research
- Work-horse for consultancy

Huangfu, H and Galabova (2011–date)

h2gmp: A high performance simplex solver

Features

- Model management: Add/delete/modify problem data
- Efficiency: High performance serial and parallel computational components
- Open-source C++
- SCIP interface

Presolve

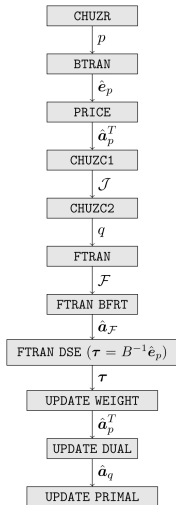
- Presolve (and corresponding postsolve) has been implemented efficiently
Remove redundancies in the LP to reduce problem dimension Galabova (2016)

Crash

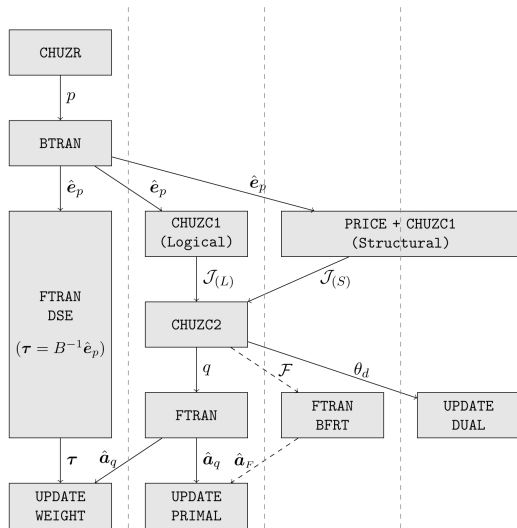
- Dual simplex “triangular basis” crash
- Alternative crash techniques being studied H and Galabova (2016–date)

h₂gmp: Single iteration parallelism with sip option

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



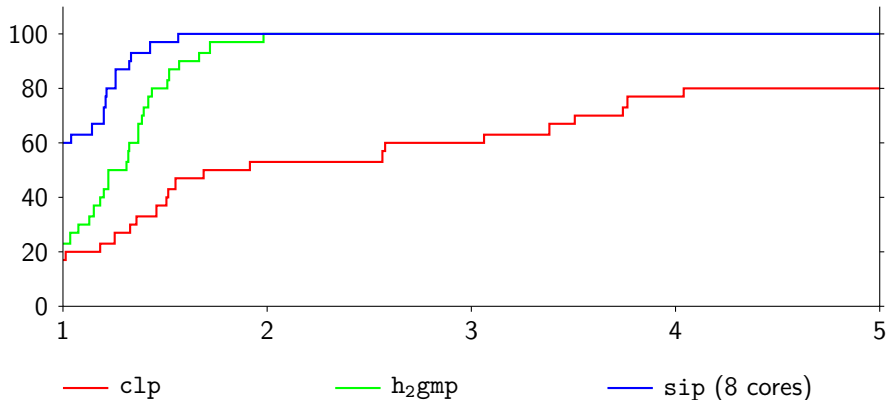
h_2gmp : Single iteration parallelism with sip option



- Parallel PRICE to form $\hat{\mathbf{a}}_p^T = \boldsymbol{\pi}_p^T N$
- 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)

h_2gmp : clp vs h_2gmp vs sip



Performance on spectrum of 30 significant LP test problems

- sip on 8 cores is 1.15 times faster than h_2gmp
- h_2gmp (sip on 8 cores) is 2.29 (2.64) times faster than clp

h2gmp: Multiple iteration parallelism with pami option

- 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*

	$\mathcal{N}$	RHS
$\mathcal{B}$	$\hat{\mathbf{a}}_{\mathcal{P}}^T$	$\hat{\mathbf{b}}$
		$\hat{b}_{\mathcal{P}}$
	$\hat{\mathbf{c}}_N^T$	

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

Huangfu and H (2011–2014)
COAP best paper prize (2015)

PAMI algorithm (without steepest edge)

Scan $\hat{b}_i < 0$ for $\mathcal{P}$ candidates to leave $\mathcal{B}$ (CHUZR)

Form $\pi_{\mathcal{P}} = B^{-1} \mathbf{e}_{\mathcal{P}}$ (multiple BTRAN); $\mathcal{Q} = \emptyset$

while $\exists \hat{b}_i < 0$ **do**

 Scan $\hat{\mathbf{b}}_{\mathcal{P}} < \mathbf{0}$ for $p \in \mathcal{P}$ to leave $\mathcal{B}$ (CHUZR-MI)

 Form $\hat{\mathbf{a}}_p^T = \pi_p^T N$ (PRICE)

 Scan $\hat{c}_j / \hat{a}_{pj} < 0$ for q to leave $\mathcal{N}$ (CHUZR-BFRT)

 Exchange p and q between $\mathcal{B}$ and $\mathcal{N}$; $\mathcal{Q} := \mathcal{Q} \cup \{q\}$

 Update $\hat{\mathbf{b}}_{\mathcal{P}} := \hat{\mathbf{b}}_{\mathcal{P}} - \alpha_P \hat{\mathbf{a}}_{\mathcal{P}q}$ (UPDATE-PRIMAL-MI)

 Update $\hat{\mathbf{c}}_N^T := \hat{\mathbf{c}}_N^T + \alpha_D \hat{\mathbf{a}}_p^T$ (UPDATE-DUAL)

Form $\hat{\mathbf{a}}_{\mathcal{Q}} = B^{-1} \mathbf{a}_{\mathcal{Q}}$ (multiple FTRAN)

Update $\hat{\mathbf{b}} := \hat{\mathbf{b}} - \alpha_P^T \hat{\mathbf{a}}_{\mathcal{Q}}$ (UPDATE-PRIMAL)

Update $\hat{\mathbf{b}}$ with respect to multiple BFRT

Numerical linear algebra:

- At the start of minor iterations, B_k^{-1} is represented using FT updates after INVERT
- In the t minor iterations the basis matrix is B_{k+i} , for $i = 1, \dots, t$
- To operate with B_{k+i}^{-1} for $i = 1, \dots, t-1$, use the **alternative product form (APF)** representation
 - **Recall:** $\bar{B} = TB$ with T easily invertible
 - Hence $B_{k+i}^{-1} = B_k^{-1} T_1^{-1} \dots T_i^{-1}$
- To operate with B_{k+t}^{-1} , perform **collective Forrest-Tomlin (CFT)** update of B_k^{-1}

Novel update procedures: are they useful?

Update 1: Alternative product form update (APF)

In minor iteration i

- $B_{k+i-1} = T_{i-1} \dots T_2 T_1 B_k$
- For chosen row p , have $\pi_p = B_k^{-T} \mathbf{e}_p$
- Get $\bar{\pi}_p = B_{k+i-1}^{-T} \mathbf{e}_p$ as

$$\begin{aligned} B_{k+i-1}^{-T} \mathbf{e}_p &= T_{i-1}^{-T} \dots T_1^{-T} B_k^{-T} \mathbf{e}_p \\ &= T_{i-1}^{-T} \dots T_1^{-T} \pi_p \end{aligned}$$

Novel update procedures: are they useful?

Update 1: Alternative product form update (APF)

- After t minor iterations with BFRT flips $\{\mathcal{F}_i\}_{i=1}^t$
 - Using PFI, change to the primal variables is

$$\sum_{i=1}^t B_{k+i}^{-1} \mathbf{a}_{Fi} = \sum_{i=1}^t \left(\left[\prod_{j=i-1}^1 E_j^{-1} \right] B_k^{-1} \mathbf{a}_{Fi} \right) \Rightarrow t \times \text{FTRAN-BFRT}$$

- Using APF $B_{k+i}^{-1} = B_k^{-1} T_1^{-1} \dots T_i^{-1}$, change to the primal variables is

$$\begin{aligned} \sum_{i=1}^t B_{k+i}^{-1} \mathbf{a}_{Fi} &= \sum_{i=1}^t \left(B_k^{-1} \prod_{j=1}^{i-1} T_j^{-1} \mathbf{a}_{Fi} \right) \\ &= B_k^{-1} \left(\sum_{i=1}^t \left[\prod_{j=1}^{i-1} T_j^{-1} \right] \mathbf{a}_{Fi} \right) \Rightarrow 1 \times \text{FTRAN-BFRT} \end{aligned}$$

Novel update procedures: are they useful?

Update 2: Middle product form update (MPF)

- **Recall:** $\bar{B} = LMU$ with M easily invertible
- Offers FT-like performance with PF-like code complexity
- Not used by pami!
- Used by Google in glom

Update 3: Multiple Forrest-Tomlin update

Used to perform multiple Forrest-Tomlin updates after minor iterations in pami

h2gmp: Performance and reliability

Extended testing using 159 test problems

- 98 Netlib
- 16 Kennington
- 4 Industrial
- 41 [Mittelman](#)

Exclude 7 which are “hard”

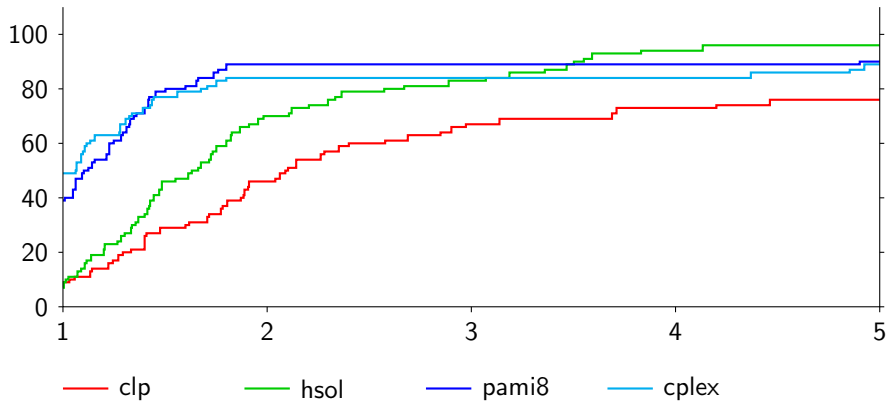
Performance

Benchmark against c1p (v1.16) and cplex (v12.5)

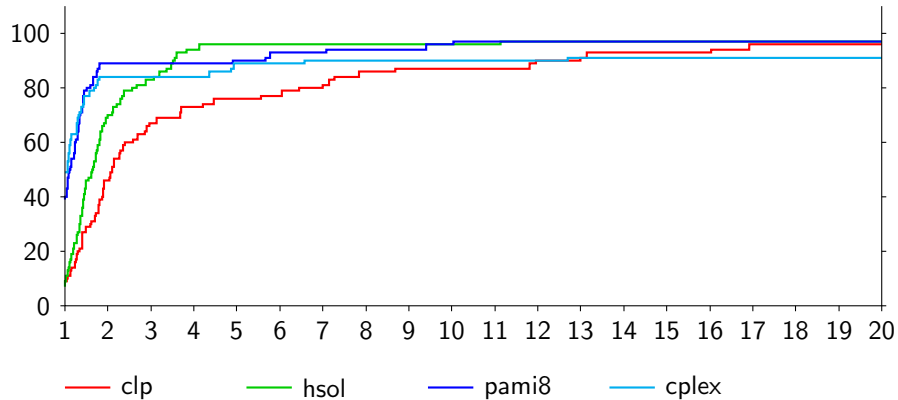
- Dual simplex
- No presolve
- No crash

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

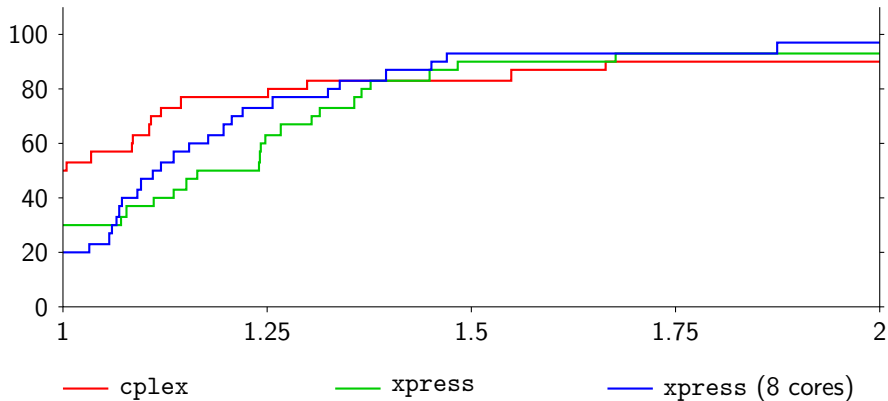
h_2 gmp: Performance



h₂gmp: Reliability



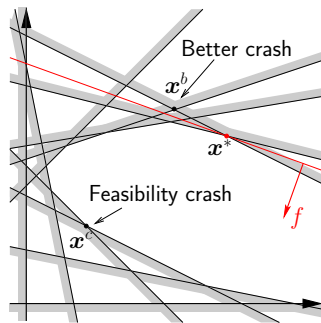
h₂gmp: Impact



- pami ideas incorporated in [FICO Xpress](#) (Huangfu 2014)
- Made Xpress simplex solver the fastest commercial code

A linear programming problem is

$$\begin{array}{ll}\min & \mathbf{c}^T \mathbf{x} \\ \text{s.t.} & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}\end{array}$$



$$\mathbf{x}^0 \xrightarrow{\text{Crash}} \mathbf{x}^k \xrightarrow{\text{Crossover}} \mathbf{x}^b \xrightarrow{\text{Simplex}} \mathbf{x}^*$$

- A good starting basis is essential for the solution of large problems
- The Idiot crash is a heuristic aiming to improve feasibility
- Implemented by John Forrest in CLP
- Seeking primal feasibility prior to primal simplex

$$\mathbf{x}^0 \xrightarrow{\text{Crash}} \mathbf{x}^k \xrightarrow{\text{Crossover}} \mathbf{x}^b \xrightarrow{\text{Simplex}} \mathbf{x}^*$$

Idiot is Augmented Lagrangian

- No documentation but the code
 - John Forrest: "so bad that I called it the Idiot algorithm"
- At each iteration minimize a weighted average of objective and infeasibilities
- Augmented Lagrangian (AL)
- Aspects that make it different from AL
 - Approximate minimization
 - Parameter adjustment
 - Other details

Augmented Lagrangian

- In iteration k

$$\mathbf{x}^{k+1} = \arg \min_{\mathbf{x}} \mathbf{c}^T \mathbf{x} + \sum_{i \in 1..m} \sigma_i^k (\mathbf{r}_i(\mathbf{x}^k) + \theta_i^k)^2,$$

- σ_i and θ_i are parameters corresponding to the i^{th} row
- $\mathbf{r}_i(\mathbf{x}^k)$ is the residual of the i^{th} row
- Adjust parameters

At start	After some iterations s	After some iterations l
$\boldsymbol{\sigma}^0 = \mathbf{1}$		$\boldsymbol{\sigma}^{l+1} := 10\boldsymbol{\sigma}^l$
$\boldsymbol{\theta}^0 = \mathbf{0}$	$\boldsymbol{\theta}^{s+1} := \boldsymbol{\theta}^s + \mathbf{r}(\mathbf{x}^s)$	$\boldsymbol{\theta}^{l+1} := \boldsymbol{\theta}^l / 10$

Alternating directions approach

- Similar to Alternating Direction Method of Multipliers (ADMM)

$$\min_{\mathbf{x}} h(\mathbf{x}) \quad \left| \quad \begin{array}{l} \min_{x_1} h(\mathbf{x}) \\ \min_{x_2} h(\mathbf{x}) \\ \min_{x_3} h(\mathbf{x}) \\ \dots \\ \min_{x_n} h(\mathbf{x}) \end{array} \right.$$

- Approximate solution
- repeated ≈ 100 times to improve accuracy

The Idiot Crash structure

Initialize $\mathbf{x}^0$, σ^0 , $\theta^0 = \mathbf{0}$

For $l \in 1..nMajor$

For $i \in 1..nMinor$

$$\forall j \in \{1..n\} \quad \min_{x_j} \quad \mathbf{c}^T \mathbf{x} + \sum_{i \in 1..m} \sigma^k (r_i(\mathbf{x}^k) + \theta_i^k)^2$$

end

Choose new σ or θ

end

- $\sigma^k = \sigma_1^k = \sigma_2^k = \dots = \sigma_m^k$
- $\theta_i^{s+1} := \frac{1}{2} r_i(\mathbf{x}^s)$

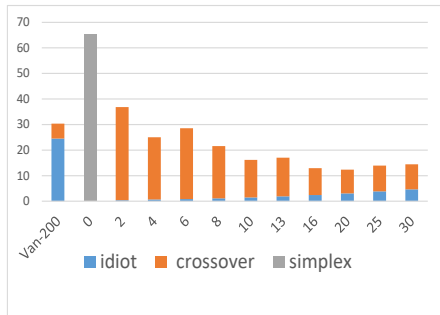
At the end of each major iteration the values of θ or σ are re-adjusted:

- Start with σ^0 , $\theta^0 = 0$, $\text{nMinor} = 2$
- At the end of each major iteration:
 - Until 10% reduction in primal infeasibility set $\sigma^{k+1} = 10\sigma^k$
- Set $\text{nMinor} = 105$
 - The first major iteration with a new σ , θ is unchanged
 - The next few major iterations set $\theta_i^{k+1} = \frac{1}{2}r_i(\mathbf{x}^k)$
 - Increase σ and repeat

- Generally useless
 - Single variable change limitation
- Solves Quadratic Assignment Problems (QAPs)
 - Combinatorial optimization problem, special case of facility location
 - Well known for being hard to solve
 - The Idiot crash reduces solution time significantly

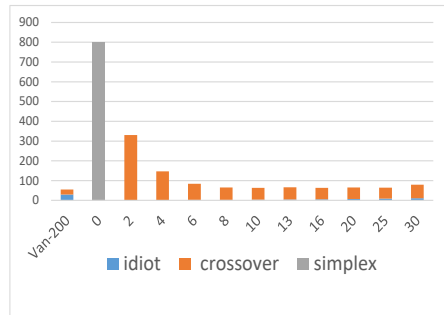
Quadratic Assignment Problems

QAP12
Solution time (s)



nMajor

QAP15
Solution time (s)



nMajor

$$\begin{array}{ll}\min & f_P = \mathbf{c}^T \mathbf{x} \\ \text{s.t.} & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}\end{array} \quad (\text{P})$$

$$\begin{array}{ll}\max & f_D = \mathbf{b}^T \mathbf{y} \\ \text{s.t.} & A^T \mathbf{y} \leq \mathbf{c}\end{array} \quad (\text{D})$$

Idiot applied to (P)

- Primal feasible
- Near-optimal objective
 $f_P \geq f^*$

Idiot applied to (D)

- Dual feasible
- Objective $f_D \leq f^*$

Problem	LB	UB	f^*	Time (s)		
				Bounds	Idiot+Crossover	Simplex
qap08	141.39	204.57	203.50	0.55	0.61	1.80
qap12	481.50	523.60	522.89	3.85	12.64	65.21
qap15	950.05	1042.05	1040.99	11.53	64.48	803.30

- QAPs require integer solution to linearization
- Fast bounds on f^* accelerate branch & bound algorithms

- Novel updates for the revised simplex method
- High performance dual simplex solver
- Open-source linear optimization package `h2gmp`
- Idiot crash and fast solution of LP problems

Slides: <http://www.maths.ed.ac.uk/hall/ZIB17>



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*.



Q. Huangfu and J. A. J. Hall.

Novel update techniques for the revised simplex method.

Computational Optimization and Applications, 60(4):587–608, 2015.