HiGHS: Theory, software and Impact

Julian Hall

School of Mathematics
University of Edinburgh

jajhall@ed.ac.uk

Optimization: Theory, Algorithms, Applications

The Fields Institute for Research in Mathematical Sciences

11 August 2021

THE UNIVERSITY of EDINBURGH

HiGHS

HiGHS: The team

What's in a name?

HiGHS: **H**all, **i**vet **G**alabova, **H**uangfu and **S**chork

Team HiGHS

- Julian Hall (1990–date)
- Ivet Galabova (2016–date)
- Michael Feldmeier (2018–date)
- Leona Gottwald (2020–date)

HiGHS: Solvers

Linear programming (LP)

- Dual simplex (Huangfu and Hall)
  - Serial techniques exploiting sparsity
  - Parallel techniques exploiting multicore architectures
- Interior point (Schork)
  - Highly accurate due to its iterative linear system solver
  - Crossover to a basic solution

Mixed-integer programming (MIP)

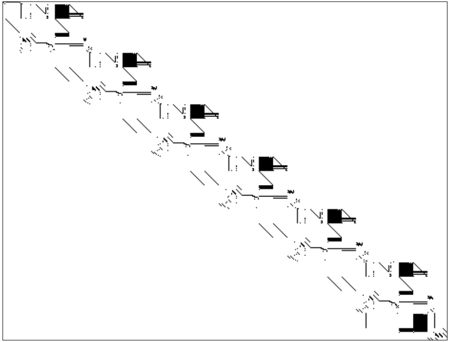
Branch-and-cut solver (Gottwald)

Quadratic programming (QP)

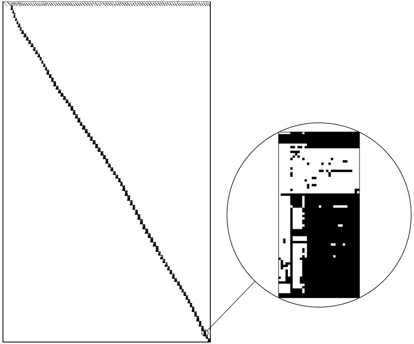
Active set solver (Feldmeier)

Practical LP problems

$$\text{minimize } f = c^T x \quad \text{such that} \quad Ax = b \quad \text{and} \quad x \geq 0$$



STAIR: 356 rows; 467 columns; 3856 nonzeros



DCP1: 4950 rows; 3007 columns; 93853 nonzeros

Large-scale practical LP problems have $O(10^7)$ variables and constraints

Solving primal LP problems: Optimality conditions

$$\text{minimize } f = \mathbf{c}^T\mathbf{x} \quad \text{such that} \quad A\mathbf{x} = \mathbf{b} \quad \text{and} \quad \mathbf{x} \geq 0$$

For a partition $\mathcal{B} \cup \mathcal{N}$ of $\{1, \ldots, n\}$ with nonsingular **basis matrix** B

- Equations partitioned as $B\mathbf{x}_B + N\mathbf{x}_N = \mathbf{b}$ so $\mathbf{x}_B = B^{-1}\mathbf{b} - B^{-1}N\mathbf{x}_N$; some $\mathbf{x}_N \geq 0$
- Objective partitioned as $f = \mathbf{c}_B^T\mathbf{x}_B + \mathbf{c}_N^T\mathbf{x}_N$
- Reduced objective is $f = \widehat{f} + \widehat{\mathbf{c}}_N^T\mathbf{x}_N$, where
  - $\widehat{f} = \mathbf{c}_B^T\widehat{\mathbf{b}}$, for **reduced RHS** $\widehat{\mathbf{b}} = B^{-1}\mathbf{b}$
  - $\widehat{\mathbf{c}}_N^T = \mathbf{c}_N^T - \mathbf{c}_B^T B^{-1}N$ is the vector of **reduced costs**
- Partition yields an optimal solution when $\mathbf{x}_N = 0$ if there is
  - Primal feasibility $\widehat{\mathbf{b}} \geq 0$
  - Dual feasibility $\widehat{\mathbf{c}}_N \geq 0$

Solving dual LP problems: Optimality conditions

Consider the corresponding **dual problem**

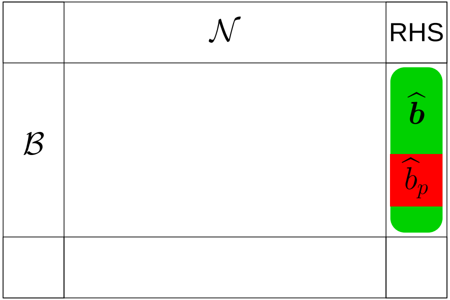
$$\text{maximize} \quad f_D = b^T y \qquad \text{subject to} \quad A^T y + s = c \qquad s \geq 0$$

- For a partition $\mathcal{B} \cup \mathcal{N}$, the partitioned equations are solved by
  - $y = B^{-T}(c_B - s_B)$
  - $s = \begin{bmatrix} s_B \\ s_N \end{bmatrix}$ for $s_N = \widehat{c}_N + N^T B^{-T} s_B$; some $s_B \geq 0$
  - Reduced objective is $f_D = \widehat{f} - \widehat{b}^T s_B$
- Solution is optimal when $s_B = 0$ if there is
  - Dual feasibility $\widehat{c}_N \geq 0$
  - Primal feasibility $\widehat{b} \geq 0$
- **Dual simplex algorithm** for an LP is *primal algorithm* applied to the *dual problem*
- Structure of dual equations allows dual simplex algorithm to be applied to primal simplex tableau

Dual simplex algorithm: Choose a row

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

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

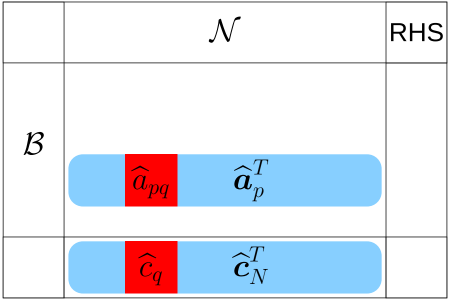


Dual simplex algorithm: Choose a column

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

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

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



Dual simplex algorithm: Update cost and RHS

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

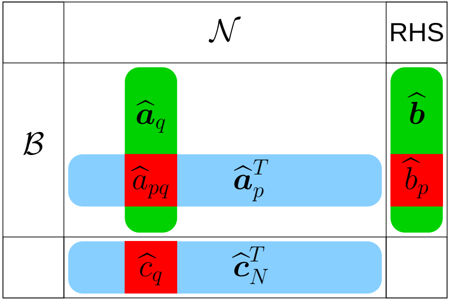
Scan $\widehat{b}_i < 0$ for p to leave $\mathcal{B}$

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

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

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

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



Dual simplex algorithm: Data required

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

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

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

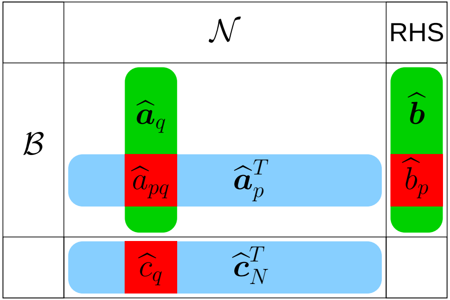
Update: Exchange p and q between $\mathcal{B}$ and $\mathcal{N}$

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

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

Data required

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



Solving LP problems: Primal or dual simplex?

Primal simplex algorithm

- Traditional variant
- Solution generally not primal feasible when (primal) LP is tightened

Dual simplex algorithm

- Preferred variant
- Easier to get dual feasibility
- More progress in many iterations
- Solution dual feasible when primal LP is tightened

Simplex method: Computation

Standard simplex method (SSM): Major computational component

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

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

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

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

Revised simplex method (RSM): Major computational components

Pivotal row via	$B^T\boldsymbol{\pi}_p = e_p$	BTRAN	and	$\widehat{a}_p^T = \boldsymbol{\pi}_p^T N$	PRICE
Pivotal column via	$B\widehat{a}_q = a_q$	FTRAN		Represent B^{-1}	INVERT
Update B^{-1} exploiting $\bar{B} = B + (a_q - Be_p)e_p^T$					UPDATE

Simplex method: Mittelmann test set

Industry standard set of 40 LP problems

	Rows	Cols	Nonzeros	$\frac{\text{Rows}}{\text{Cols}}$	$\frac{\text{Nonzeros}}{\text{Rows} \times \text{Cols}}$	$\frac{\text{Nonzeros}}{\max(\text{Rows},\text{Cols})}$
Min	960	1560	38304	1/255	0.0005%	2.2
Geomean	54256	72442	910993	0.75	0.02%	6.5
Max	986069	1259121	11279748	85	16%	218.0

Mittelmann measure for solvers

- Unsolved problems given "timeout" solution time
- Shift all solution times up by 10s
- Compute geometric mean of logs of shifted times
- **Solution time measure** is exponent of geometric mean shifted down by 10s
- **Mittelmann measure** for a solver is its solution time measure relative to the best

Hyper-sparsity

Hyper-sparsity: Solve $Bx = r$ for sparse r

- Given $B = LU$, solve $Bx = r$ as

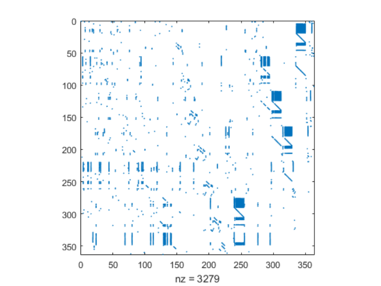
$$Ly = r; \quad Ux = y$$

- In revised simplex method, r is sparse: consequences?
  - If B is irreducible then x is full
  - If B is highly reducible then x can be **sparse**
- Phenomenon of **hyper-sparsity**
  - Exploit it when *forming* x
  - Exploit it when *using* x

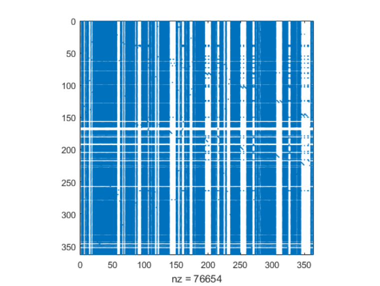
Hyper-sparsity: Inverse of a sparse matrix

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

Optimal B for LP problem STAIR



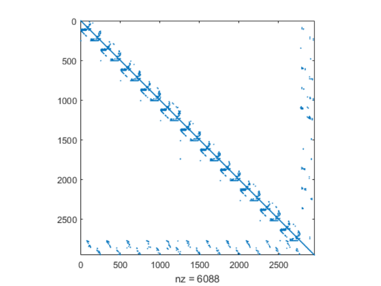
B^{-1} has density of 58%, so $B^{-1}r$ is typically dense



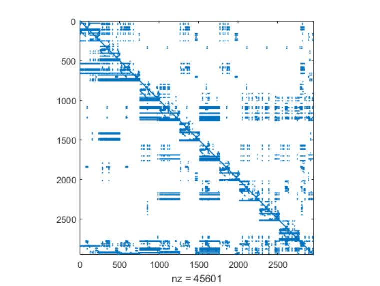
Hyper-sparsity: Inverse of a sparse matrix

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

Optimal B for LP problem PDS-02



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



Hyper-sparsity

- Use solution of $Lx = b$
  - To illustrate the phenomenon of hyper-sparsity
  - To demonstrate how to exploit hyper-sparsity
- Apply principles to other triangular solves in the simplex method

Hyper-sparsity: Solving $Lx = b$

Recall: Solve $Lx = b$ using

function `ftranL`(L, b, x)

$r = b$

for all $j \in \{1, \dots, m\}$ **do**

for all $i : L_{ij} \neq 0$ **do**

$r_i = r_i - L_{ij} r_j$

$x = r$

When b is **sparse**

- Inefficient until r fills in

Hyper-sparsity: Solving $Lx = b$

Better: Check r_j for zero

function `ftranL`(L, b, x)
  r = b
  for all $j \in \{1, \ldots, m\}$ **do**
    if $r_j \neq 0$ **then**
      for all $i : L_{ij} \neq 0$ **do**
        $r_i = r_i - L_{ij} r_j$
  x = r

When x is **sparse**

- Few values of r_j are *nonzero*
- Check for zero dominates
- Requires more efficient identification of set $\mathcal{X}$ of indices j such that $r_j \neq 0$

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

Hyper-sparsity: Other components

Recall: major computational components

- FTRAN: Form $\hat{a}_q = B^{-1}a_q$
- BTRAN: Form $\pi_p = B^{-T}e_p$
- PRICE: Form $\hat{a}_p^T = \pi_p^T N$

BTRAN: Form $\pi_p = B^{-T}e_p$

- Transposed triangular solves
- $L^T x = b$ has $x_i = b_i - l_i^T x$
  - Hyper-sparsity: $l_i^T x$ typically zero
  - Also store L (and U) row-wise and use FTRAN code

PRICE: Form $\hat{a}_p^T = \pi_p^T N$

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

H and McKinnon (1998–2005)

Hyper-sparsity: Effectiveness

Testing environment

- Mittelmann test set of 40 LPs
- HiGHS dual simplex solver with/without exploiting hyper-sparsity
- Time limit of 10,000 seconds

Results

- When exploiting hyper-sparsity: solves 37 problems
- When not exploiting hyper-sparsity: solves 34 problems

	Min	Geomean	Max
Iteration count increase	0.75	1.08	3.17
Solution time increase	0.83	2.31	67.13
Iteration speed decrease	0.92	2.14	66.43
Mittelmann measure		2.57	

Parallel solution of structured LP problems

Parallel solution of stochastic MIP problems

Two-stage stochastic LPs have column-linked block angular (BALP) structure

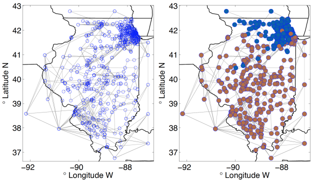
$$
\begin{array}{rccccccccccc}
\text{minimize} & c_0^T x_0 & + & c_1^T x_1 & + & c_2^T x_2 & + & \dots & + & c_N^T x_N & & \\
\text{subject to} & A x_0 & & & & & & & & & = & b_0 \\
& T_1 x_0 & + & W_1 x_1 & & & & & & & = & b_1 \\
& T_2 x_0 & & & + & W_2 x_2 & & & & & = & b_2 \\
& \vdots & & & & & & \ddots & & & & \vdots \\
& T_N x_0 & & & & & & & + & W_N x_N & = & b_N \\
& x_0 \geq 0 & & x_1 \geq 0 & & x_2 \geq 0 & & \dots & & x_N \geq 0 & &
\end{array}
$$

- Variables $x_0 \in \mathbb{R}^{n_0}$ are **first stage** decisions
- Variables $x_i \in \mathbb{R}^{n_i}$ 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

Parallel solution of stochastic MIP problems

- Power systems optimization project at Argonne
- Integer second-stage decisions
- Stochasticity from wind generation
- 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`



PIPS-S: Exploiting problem structure

Convenient to permute the LP thus:

$$
\begin{array}{rccccccccccc}
\text{minimize} & c_1^T x_1 & + & c_2^T x_2 & + & \ldots & + & c_N^T x_N & + & c_0^T x_0 & & \\
\text{subject to} & W_1 x_1 & & & & & & & + & T_1 x_0 & = & b_1 \\
& & & W_2 x_2 & & & & & + & T_2 x_0 & = & b_2 \\
& & & & & \ddots & & & \vdots & & & \vdots \\
& & & & & & & W_N x_N & + & T_N x_0 & = & b_N \\
& & & & & & & & & A x_0 & = & b_0 \\
& x_1 \geq 0 & & x_2 \geq 0 & & \ldots & & x_N \geq 0 & & x_0 \geq 0 & &
\end{array}
$$

PIPS-S: Exploiting problem structure

- Inversion of the basis matrix B is key to revised simplex efficiency

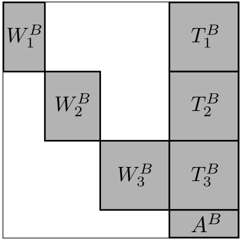
$$B = \begin{bmatrix} W_1^B & & & T_1^B \\ & \ddots & & \vdots \\ & & W_N^B & T_N^B \\ & & & A^B \end{bmatrix}$$

- 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

PIPS-S: Exploiting problem structure

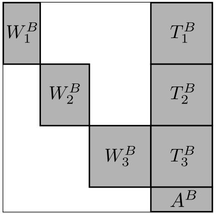
- Inversion of the basis matrix B is key to revised simplex efficiency

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

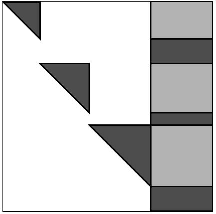


- B is nonsingular so
  - W_i^B are "tall": full column rank
  - $\begin{bmatrix} W_i^B & T_i^B \end{bmatrix}$ are "wide": full row rank
  - A^B is "wide": full row rank
- Scope for parallel inversion is immediate and well known

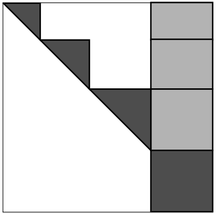
PIPS-S: Exploiting problem structure



- 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

PIPS-S: Overview

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

Lubin, H, Petra and Anitescu (2013)

PIPS-S: Results

On Fusion cluster: Performance relative to Clp

Dimension	Cores	Storm	SSN	UC12	UC24
$m+n=O(10^6)$	1	0.34	0.22	0.17	0.08
	32	8.5	6.5	2.4	0.7
$m+n=O(10^7)$	256	299	45	67	68

On Blue Gene

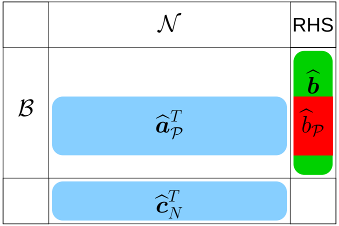
- Instance of UC12
- $m+n=O(10^8)$
- Requires 1 TB of RAM
- Runs from an advanced basis

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

Parallel solution of general LP problems

Parallel solution of general LP problems via multiple iterations: `pami`

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

Huangfu and H (2011–2014)

pami: Effectiveness

Serial overhead of pami

- HiGHS pami solver in serial: solves 34/40 problems

	Min	Geomean	Max
Iteration count increase	0.43	1.02	2.98
Solution time increase	0.31	1.62	5.36
Iteration speed decrease	0.69	1.59	5.11
Mittelmann measure	2.08		

Parallel speed-up of pami with 8 threads

	Min	Geomean	Max
Iteration count decrease	1.00	1.00	1.00
Solution time decrease	1.15	1.88	2.39
Iteration speed increase	1.15	1.88	2.39

pami: Effectiveness

Performance enhancement using parallel pami with 8 threads

	Min	Geomean	Max
Iteration count decrease	0.34	0.98	2.34
Solution time decrease	0.34	1.16	6.44
Iteration speed increase	0.38	1.18	2.75
Mittelmann measure		1.21	

Observations

- There is significant scope to improve pami performance further
- Use pami tactically: switch it off if it is ineffective

Commercial impact

- Huangfu applied the parallel dual simplex techniques within the Xpress solver
- For much of 2013–2018 the Xpress simplex solver was the best in the world

HiGHS

HiGHS: Features and interfaces

Features

- Load/add/delete/modify model data
- Presolve/postsolve (LP and MIP)
- Solution sensitivity/ranging (LP)

Interfaces

- Language
  - C++ HiGHS class
  - C
  - C#
  - FORTRAN
  - Python
  - JavaScript
  - Rust
- Applications
  - JuliaOpt
  - SciPy
  - PuLp
  - GAMS*
  - OSI*
  - SCIP*
- Future
  - AMPL
  - MATLAB
  - Mosel
  - OpenSolver
  - R

  Suggestions?

HiGHS: Access

- Open-source (MIT license)
  - GitHub: ERGO-Code/HiGHS
  - COIN-OR
- No third-party code required
- Runs under Linux, Windows and MacOS
- Build requires `CMake` 3.15
- Compilation requires (eg) GNU `gcc/g++` 4.9
- Parallel code uses `OpenMP`
- Documentation: https://www.HiGHS.dev/

HiGHS: Performance

- Problems
  - LP simplex: Mittelmann's 40 problems
  - LP interior point: Mittelmann's 46 problems
  - MIP: MIPLIB 2017 240 problems
- Time allowance
  - LP: 3600s
  - MIP: 7200s
- Solvers
  - LP: `Clp`
  - MIP: `Cbc` and `SCIP`
- Machines: mixed!

HiGHS: Simplex performance

	Solved	Time
Clp	38	1
HiGHS	32	2.5

- Mittelmann benchmarks use an old version of HiGHS and give 29 solved and time of 3.8 relative to Clp
- Simplex performance improvement due to improved presolve and more reliable primal simplex clean-up
- Much scope for further improvement
  - Add dualisation
  - Add primal simplex switch
  - Add sifting
  - Use and improve parallel solver
  - Add Idiot crash and crossover
  - Improve Idiot crash

HiGHS: Interior point performance

	Solved	Time
Clp	31	2.9
HiGHS	42	1

- Clp fails on (only) 3 due to insufficient memory
- HiGHS simply faster

HiGHS: MIP performance

- Cbc and SCIP results are on Mittelmann's machine
- HiGHS results are on Gottwald's: comparable

	Solved	Time
Cbc	89	1.9
HiGHS (June 2021)	104	1.7
HiGHS (July 2021)	113	1.4
SCIP	125	1

"I must admit that HiGHS is beating Cbc big time" [PuLP user]

- Significant further performance improvement expected
  - 17 Problems not solved due to simplex bug
  - Scope for greater efficiency in simplex-MIP solver interaction
  - More features still to be added to MIP solver

HiGHS: QP performance

- No results to show
- Solves 3 of the 7 strictly convex QPLIB instances
- Very much "work in progress"

HiGHS: The future

- First formal release! End of August 2021?
- Further interfaces
  - AMPL
  - MATLAB
  - Mosel
  - OpenSolver
  - R
- Displace Clp, Cbc
  - Performance of HiGHS is generally superior
  - HiGHS is being developed actively
  - HiGHS is agile to demands of interfaces

"We could get rid of COIN-OR and just use HiGHS and IPOPT"

HiGHS: Impact

HiGHS: Impact (REF2021)

Animal feed formulation

- Animal feed is blended from many ingredients
  - Multiple animals; multiple diets
  - Shared raw materials
  - Formulate at minimum cost!
- Dantzig-Wolfe structure

$$
\begin{array}{llcccll}
\min & c_1^T x_1 & + \ldots + & c_N^T x_N & & \\
\text{s.t.} & A_{01} x_1 & + \ldots + & A_{0N} x_N & \leq & b_0 \\
 & A_1 x_1 & & & = & b_1 \\
 & & \ddots & & & \vdots \\
 & & & A_N x_N & = & b_N \\
 & \multicolumn{3}{c}{x_1 \geq 0 \;\ldots\; x_N \geq 0} & &
\end{array}
$$

Cargill (Format Solutions)

- 25-year relationship
- Format software blends half the world's manufactured animal food
  - Farm food: $O(\$1\text{bn})$
  - Pet food: $O(\$10\text{bn})$
  - Trade commodities
- HiGHS ten times faster than EMSOL

- High performance LP solvers: simplex and IPM
- High performance MIP solver
- Prototype QP solver
- Language interfaces: C++, `C`, `C#`, `FORTRAN`, `Python`
- Application interfaces: `JuliaOpt`, `SciPy`, `PuLp`, ...
- Permissive license and no third-party code
- Available for research and consultancy

https://www.HiGHS.dev/

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

Q. Huangfu and J. A. J. Hall.
Parallelizing the dual revised simplex method.
Mathematical Programming Computation, 10(1):119–142, 2018.

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.

L. Schork and J. Gondzio.
Implementation of an interior point method with basis preconditioning.
Mathematical Programming Computation, 12:603–635, 2020.