

# HiGHS: Theory, software and Impact

Julian Hall

School of Mathematics  
University of Edinburgh

jajhall@ed.ac.uk

Seminar

KTH Royal Institute of Technology

1 October 2021



THE UNIVERSITY  
*of* EDINBURGH



# 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)



## 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)

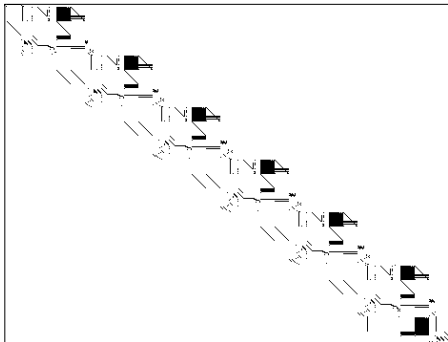
Branch-and-cut solver (Gottwald)

## Quadratic programming (QP)

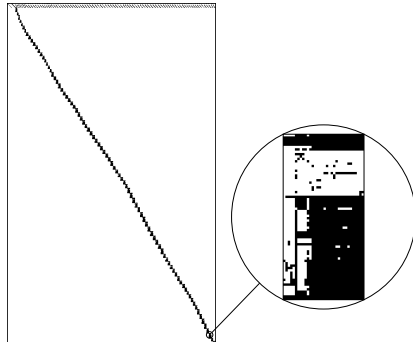
Active set solver (Feldmeier)

# 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

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

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

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

# Solving dual LP problems: Optimality conditions

Consider the corresponding **dual problem**

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

- 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  $s_N = \hat{c}_N + N^T B^{-T} s_B$ ; some  $s_B \geq 0$
  - Reduced objective is  $f_D = \hat{f} - \hat{b}^T s_B$
- Solution is optimal when  $s_B = 0$  if there is
  - **Dual feasibility**  $\hat{c}_N \geq 0$
  - **Primal feasibility**  $\hat{b} \geq 0$
- **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  $\hat{c}_N \geq 0$     Seek  $\hat{b} \geq 0$

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

|               | $\mathcal{N}$ | RHS                      |
|---------------|---------------|--------------------------|
| $\mathcal{B}$ |               | $\hat{b}$<br>$\hat{b}_p$ |
|               |               |                          |

# Dual simplex algorithm: Choose a column

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  $\mathcal{N}$

|               | $\mathcal{N}$                                                                       | RHS |
|---------------|-------------------------------------------------------------------------------------|-----|
| $\mathcal{B}$ |  |     |
|               |  |     |

# Dual simplex algorithm: Update cost and RHS

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

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

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

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

# Dual simplex algorithm: Data required

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

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

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

Data required

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

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

# 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       |
| $\mathcal{B}$ | $\hat{N}$     | $\hat{b}$ |
|               | $\hat{c}_N^T$ |           |

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

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  $B^T \pi_p = e_p$  **BTRAN** and  $\hat{a}_p^T = \pi_p^T N$  **PRICE**

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

Update  $B^{-1}$  exploiting  $\bar{B} = B + (a_q - B e_p) e_p^T$  **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

- Run time is limited: failures and timeouts are given this 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: Solve $Bx = r$ for sparse $r$

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

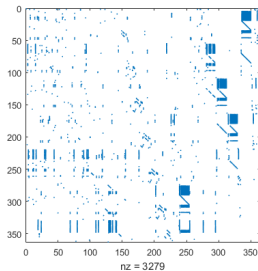
$$Ly = r; \quad Ux = y$$

- 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*  $x$

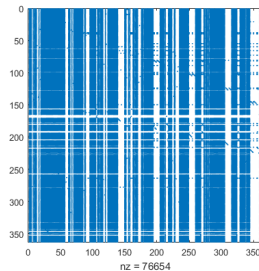
# Hyper-sparsity: Inverse of a sparse matrix

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

Optimal  $B$  for LP problem STAIR



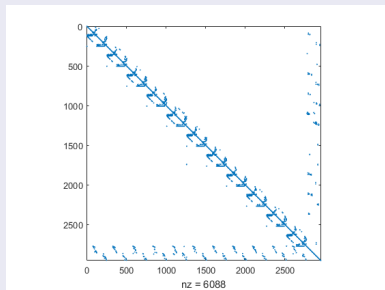
$B^{-1}$  has density of 58%, so  $B^{-1}r$  is typically dense



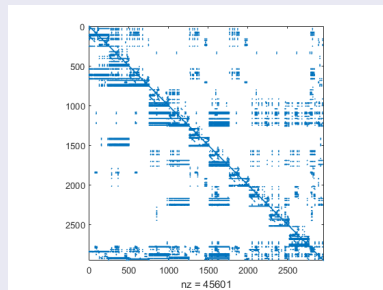
# Hyper-sparsity: Inverse of a sparse matrix

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



- 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  $j \in \{1, \dots, m\}$  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:  $\pi_p^T$  is sparse
- Store  $N$  row-wise
- Form  $\hat{a}_p^T$  as a combination of rows of  $N$  for nonzeros in  $\pi_p^T$

H and McKinnon (1998–2005)

# Hyper-sparsity: Effectiveness

## Testing environment

- Mittelman 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 |
| Mittelman measure        | 2.57 |         |       |

# 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*

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

Huangfu and H (2011–2014)

## 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 |
| Mittelman 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 |

## 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 |
| Mittelman 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: Features and interfaces

## Features

- Load/add/delete/modify model data
- Presolve/postsolve (LP and MIP)
- Solution sensitivity/ranging (LP)
- Open source (MIT license) GitHub: [ERGO-Code/HiGHS](#)
- No third-party code required
- Builds under Linux, Windows and MacOS using CMake

## Interfaces

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

## Two routes

- Use SciPy
  - Currently only LP
  - MIP interface planned
- Install HiGHS and call it directly
  - Slightly clunky
  - No Python programmers in the HiGHS teamExternal contributions welcome!

## Illustration

Use [SciPy-HiGHS-KTH.py](#) to solve

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

for a given matrix [A](#), RHS [b](#) and cost vector [c](#)

- Problems
  - LP simplex: Mittelmann's 40 problems
  - LP interior point: Mittelmann's 47 problems
  - MIP: MIPLIB 2017 240 problems
- Time allowance
  - LP: 15000s
  - MIP: 7200s
- Solvers
  - LP: C1p
  - MIP: Cbc and SCIP
- Machines: mixed!

# HiGHS: Simplex performance

| Closed source |        |       |         |        |       |        |
|---------------|--------|-------|---------|--------|-------|--------|
| Solver        | Huawei | COPT  | Alibaba | Gurobi | Mosek | Matlab |
| Speed         | 1      | 1.13  | 1.29    | 1.64   | 15.3  | 33.2   |
| Solved/40     | 39     | 39    | 39      | 40     | 35    | 29     |
| Open source   |        |       |         |        |       |        |
| Solver        | Clp    | HiGHS | Glop    | Soplex | Glpk  |        |
| Speed         | 6.33   | 14.6  | 23.6    | 34.4   | 88.1  |        |
| Solved/40     | 37     | 31    | 27      | 31     | 25    |        |

- Benchmarks dominated by three Chinese solvers
- HiGHS better than Mosek and Matlab, and all open source solvers except Clp
- 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

| Closed source |       |        |         |       |        |        |
|---------------|-------|--------|---------|-------|--------|--------|
| Solver        | COPT  | Gurobi | Alibaba | Mosek | Knitro | Matlab |
| Speed         | 1     | 1.38   | 2.44    | 4.17  | 12.6   | 32.5   |
| Solved/47     | 47    | 47     | 46      | 42    | 37     | 35     |
| Open source   |       |        |         |       |        |        |
| Solver        | HiGHS | Tulip  | Clp     |       |        |        |
| Speed         | 13.5  | 49.5   | 60.8    |       |        |        |
| Solved/47     | 41    | 31     | 35      |       |        |        |

HiGHS vastly faster than the other open source solvers - and Matlab

# HiGHS: MIP performance

- HiGHS not yet listed on Mittelmann's MIP benchmarks
  - Cbc and SCIP results are on Mittelmann's machine
  - HiGHS results are on Gottwald's

|                     | Solved/240 | Time |
|---------------------|------------|------|
| Cbc                 | 89         | 8.6  |
| HiGHS (June 2021)   | 104        | 7.6  |
| HiGHS (July 2021)   | 113        | 6.4  |
| HiGHS (August 2021) | 128        | 5.3  |
| SCIP                | 125        | 4.5  |
| COPT                | 132        | 4.2  |
| Gurobi              | 201        | 1    |

- Significant further performance improvement expected
  - Some problems not solved due to simplex bug
  - Scope for greater efficiency in simplex-MIP solver interaction
  - Many more features still to be added to MIP solver

- First formal release! End of 2022?
- Further interfaces
  - AMPL
  - MATLAB
  - Mosel
  - OpenSolver
  - OR-Tools
  - Pyomo
  - 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”



## Animal feed formulation

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

$$\begin{array}{ll} \min & c_1^T x_1 + \dots + c_N^T x_N \\ \text{s.t.} & A_{01} x_1 + \dots + A_{0N} x_N \leq b_0 \\ & A_{11} x_1 = b_1 \\ & \quad \quad \quad \ddots \quad \quad \quad \vdots \\ & \quad \quad \quad A_{N1} x_1 = b_N \\ & x_1 \geq 0 \dots x_N \geq 0 \end{array}$$



## 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

**Slides:** <http://www.maths.ed.ac.uk/hall/KTH21>

**HiGHS:** <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.