Numerics in LP and MIP Solvers

PD Dr. Timo Berthold
Private Lecturer @ TU Berlin
Director, Mixed-Integer Optimization @ FICO

We need to talk...

Numerical Trouble and Inaccuracies, examples from SCIP

```
SCIP> display solution

objective value:                 147726.165057267
x2826                                           1   (obj:13.8132)
x2840                                           1   (obj:17.3592)
x2842                             1.000000000318   (obj:17.2398)
x2845                                           1   (obj:12)
x2850                                           1   (obj:12.1173)
x2851                             0.999999999301   (obj:14.4105)
x2852                             0.999999999987   (obj:12)
x2854                                           1   (obj:14.9862)
x2857                                           1   (obj:18.7107)
x2861                                           1   (obj:15.7479)
```

```
  9.7s|     1 |     0 |  3507 |     - |   154M |   0 |2712 |  13k|  13k|  68 |  8 |  18 |   0 | 9.625279e+04 | 3.723994e+05 | 286.90%| unknown
 10.5s|     1 |     0 |  3645 |     - |   157M |   0 |2712 |  13k|  13k|  93 |  9 |  18 |   0 | 9.625416e+04 | 3.723994e+05 | 286.89%| unknown
 11.0s|     1 |     0 |  3718 |     - |   158M |   0 |2712 |  13k|  13k| 110 | 10 |  18 |   0 | 9.625435e+04 | 3.723994e+05 | 286.89%| unknown
 11.1s|     1 |     0 |  3782 |     - |   160M |   0 |2712 |  13k|  13k| 124 | 11 |  18 |   0 | 9.625474e+04 | 3.723994e+05 | 286.89%| unknown
 11.3s|     1 |     0 |  3893 |     - |   161M |   0 |2712 |  13k|  13k| 144 | 12 |  18 |   0 | 9.625534e+04 | 3.723994e+05 | 286.89%| unknown
 time | node  | left  |LP iter|LP it/n|mem/heur|mdpt |vars |cons |rows |cuts |sepa|confs|strbr|  dualbound   | primalbound  |  gap   | compl.
 24.3s|     1 |     2 | 35722 |     - |   163M |   0 |2712 |  13k|  13k| 144 | 12 |  44 |  19 | 9.626681e+04 | 3.723994e+05 | 286.84%| unknown
(node 5) numerical troubles in LP 104 -- unresolved
L 129s|    76 |    73 |127950 |1663.7 |    alns|  20 |2712 |  13k|  13k| 264 |  2 |  75 | 519 | 9.628491e+04 | 3.530913e+05 | 266.72%| unknown
 144s|   100 |    97 |148598 |1469.0 |   187M |  23 |2712 |  13k|  13k| 285 |  1 |  94 | 553 | 9.628491e+04 | 3.530913e+05 | 266.72%| unknown
```

```
2800s| 93200 |   348 |  4749k|  49.7 |   253M |  55 | 289 |1925 |1764 | 207k|  0 |5691 |8539 | 3.027253e+03 | 3.336240e+03 |  10.21%|  98.40%

SCIP Status        : problem is solved [optimal solution found]
Solving Time (sec) : 2802.85
Solving Nodes      : 93263 (total of 95347 nodes in 2 runs)
Primal Bound       : +3.33623984845755e+03 (43 solutions)
Dual Bound         : +3.33623984845755e+03
Gap                : 0.00 %
[linear] <c13452>:  -0.002186551<x12>[C] (+38233.728) +<x47>[C] (+85.2158153) +1600000<x477>[B] (+0) +1600000<x622>[I] (-1.15885912e-11)
+1600000<x682>[I] (+0) >= 1.615819082;
                                    ;

violation: left hand side is violated by 1.85415070872441e-05
best solution is not feasible in original problem
```

Numerical Trouble and Inaccuracies, examples from Xpress

```
                                Max primal violation     (abs/rel) : 9.082e-05 / 9.082e-05
                                Max integer violation    (abs    ) : 1.016e-06

Numerical issues encountered:
   Dual failures    :      27 out of    475642 (ratio: 0.0001)
   Primal failures  :      11 out of     44260 (ratio: 0.0002)
   Singular bases   :       1 out of   1506198 (ratio: 0.0000)
   Nodes w/LP fails :      10 out of    365211 (ratio: 0.0000)


      Node     BestSoln    BestBound   Sols Active  Depth     Gap     GInf   Time
    233058  1041.108026   391.326068     98 173098     16   62.41%     868    366
    243060  1041.108026   391.445426     98 179888    241   62.40%     539    378
  ?262 Warning: Unable to remove shift infeasibilities of .000157046
    253063  1041.108026   391.840607     98 187133     27   62.36%     801    392
    263065  1041.108026   392.210378     98 195618    136   62.33%     704    405
    273068  1041.108026   392.329985     98 203498     61   62.32%     903    418
  ?262 Warning: Unable to remove shift infeasibilities of 1.13342e-5
    283068  1041.108026   393.050907     98 211041     33   62.25%     989    432

     19925      -942187230.1     QP      0     0         .000000     3
  Refining primal/dual infeasibility of 1.920e-06 / 9.888e-07
     19925      -942187230.1     QP      2     0         .000000     3
```

Why is this?

Getting trust issues?

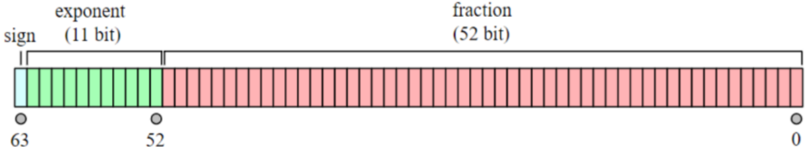
- There is no such thing as $\mathbb{R}$
  - or $\mathbb{Q}$
  - or $\mathbb{Z}$
  - or an infinite set
  - or continuous variables
- There is only `int64` and `double` (and some even less permissive data types)*
- We were promised ~~jetpacks~~ mathematical and algorithmic beauty
  - Today, the time has come to open the poison cabinet…

*= Yes, I know quad precision is a thing. But not a commonly used one or a fast one or one that would invalidate any of the above bold claims

Floating-Point Arithmetic and MIP Tolerances

Floating-Point Arithmetic

Virtually all MIP solvers are built on double-precision floating-point arithmetic (IEEE754):



- Real numbers stored as $(-1)^{\text{sign}} \cdot 1.\text{fraction} \cdot 2^{\text{exponent}-1023}$
  - This "only" allows for numbers between $\pm 2^{2^{10}} \approx \pm 1.8 \cdot 10^{308} =$
    ± 1800 00 $000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000$
- Enough to represent about 15 digits ⇝ round-off errors afterwards, e.g.
  - $0.07 =_{fp} 0.07000000000000000666\ldots$
  - $100 \cdot 0.07 =_{fp} 7.00000000000000000888\ldots$
  - $10^{14} \cdot 0.7 - 7 \cdot 10^{13} =_{fp} -0.0078125 \ldots$
- For values larger than 10^{15}, integrality becomes meaningless
  - Infinity threshold in many solvers: $|x| = \infty \Leftrightarrow_{tol} |x| \geq 10^{20}$
  - Fun fact: 10^{20} is exactly fp64-representable. The largest finite number is 99999999999999983616

Feasibility and Optimality in Floating-Point Solvers

- MIP solvers use numerical tolerances, typically in the range 10^{-6} to 10^{-9}:
  - Integrality tolerance ε_{int}: $\alpha \in \mathbb{Z} \Leftrightarrow_{\text{tol}} \alpha \in \mathbb{Z} + [-\varepsilon_{\text{int}}, \varepsilon_{\text{int}}]$, e.g., $0.9999999 =_{\text{tol}} 1$
  - Feasibility tolerance $\varepsilon_{\text{feas}}$: $a^T x \leq b \Leftrightarrow_{\text{tol}} \ldots$

    Absolute: $|a^T x - b| \leq \varepsilon_{\text{feas}}$

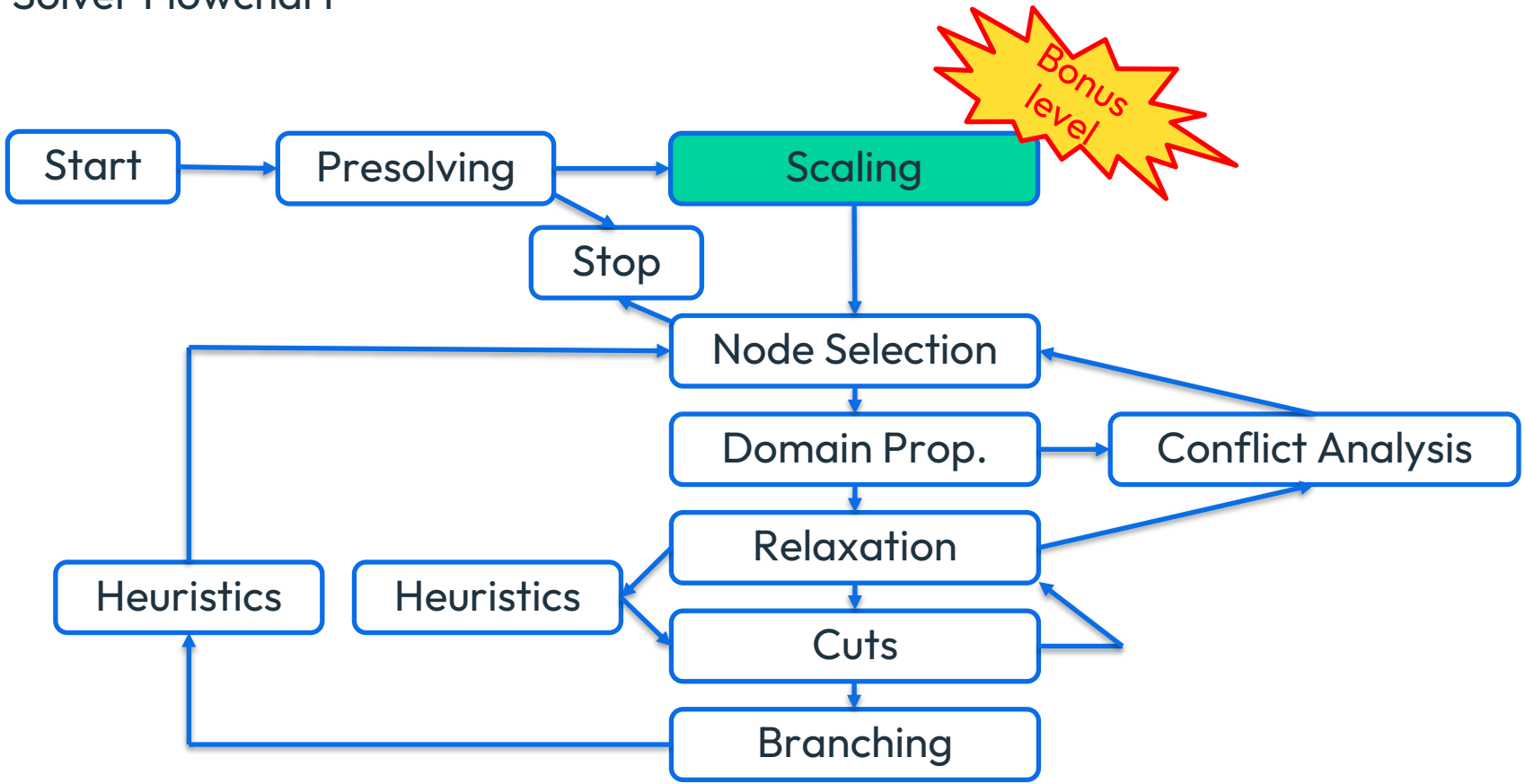
    Relative: $|aTx-b| / |b| \leq \varepsilon_{feas}$ (problematic for $|b| \approx 0$)

    Mixed (SCIP): $|aTx - b| / \max\{|b|,1\} \leq \varepsilon_{\text{feas}}$
  - LP tolerances for dual feasibility, barrier convergence, …

Note: not invariant under scaling!

Scaling?

MIP Solver Flowchart



What is Scaling?

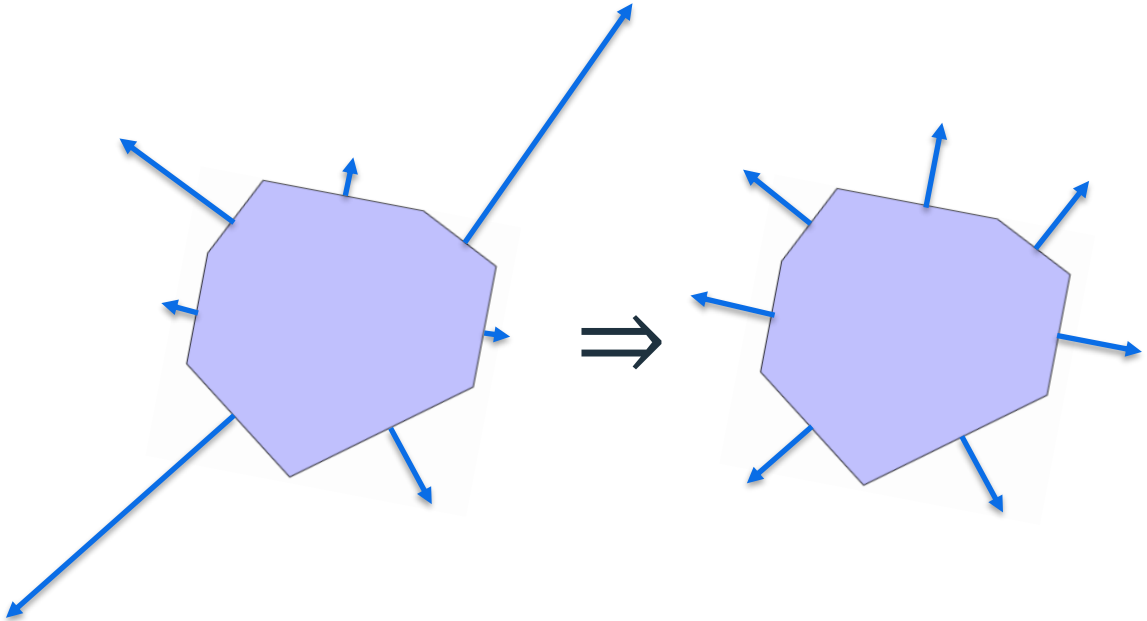
- Scaling is a widely used preconditioning technique, used by various kinds of algorithms
  - to improve the numeric behavior of the algorithms
  - to reduce error propagation
  - to reduce the number of iterations required to solve the problem
- LP Scaling refers to the (iterative) multiplication of rows and columns by scalars
  - to reduce the absolute magnitude and relative difference…
    - …of nonzero coefficients in matrix, rhs, and objective
- Goal in LP/MIP:
  - Improve simplex behavior: avoid ill-conditioned bases, unstable pivots, inaccurate duals
  - Avoid cuts that are too weak / numerically unstable
  - Reminder: Optimization solvers are based on floating-point arithmetic

Scaling in Linear Programming

- Basic Linear Program (LP):

$$\begin{aligned} \max \quad & c^T x \\ \text{s.t.} \quad & Ax \leq b \end{aligned}$$

- Scaling multiplies rows and columns
  - to bring coefficients "on one scale"
  - Typically, close to 1 (normalization)



$$\begin{aligned} \max \quad & (cC)^T (C^{-1}x) \\ \text{s.t.} \quad & (RAC)(C^{-1}x) \leq Rb \end{aligned}$$

$$c' = cC,\ A' = RAC,\ b' = Rb$$

$$x' = C^{-1}x$$

$$\begin{aligned} \max \quad & c'^T x' \\ \text{s.t.} \quad & A'x' \leq b' \end{aligned}$$

R: Square diagonal matrix of row scalars
C: Square diagonal matrix of column scalars

Can only be meaningfully applied to continuous columns

Two scaling methods

Equilibrium scaling (1975):

- Divide rows by largest coefficient
- Then divide columns by largest coefficient
- Potentially iterate

Curtis-Reid (1972):

- Minimize least-squares deviation from 1:
- $\min \sum_{i=1}^{m} \sum_{j=1}^{n} \left(\log R_{ii} C_{jj} |A_{ij}| \right)^2$
- Positive semidefinite, unconstrained → conjugate gradient
- Binary logarithm, round scaling factors to powers of two
  - → scaling is just a shift of the exponent, no rounding errors

Example

- We want to set up a home business to make boxes or chess pieces
  - We want to maximize profit [$5/box, $10/chess piece]
  - We have a limited amount of wood [100]
  - We have to buy tools [$30 for boxes, $500 for chess sets]

- A mixed integer programming (MIP) problem:

$$
\begin{aligned}
max \quad & 5x^{box} + 10x^{chess} - 30b^{box} - 500b^{chess} \\
s.t. \quad & x^{box} + x^{chess} \leq 100 \\
& x^{box} \leq 100b^{box} \\
& x^{chess} \leq 100b^{chess} \\
& b^{box}, b^{chess} \in \{0,1\}
\end{aligned}
$$

- Coefficient matrix:

  … with a potential basis matrix

$$
\begin{bmatrix}
1 & 1 & & \\
1 & & -100 & \\
 & 1 & & -100
\end{bmatrix}
$$

Example

Basis inverse

- Unscaled:

$$\begin{array}{rrl} x^{box} + x^{chess} & & \leq 100 \\ x^{box} & - 100b^{box} & \leq 0 \\ x^{chess} & - 100b^{chess} & \leq 0 \end{array} \qquad \begin{bmatrix} -\frac{1}{100} & -\frac{1}{100} & \frac{1}{100} \\ & -1 & 1 \\ & 1 & \end{bmatrix} \qquad \kappa \approx 245$$

- Equilibrium scaling:

$$\begin{array}{rrl} x^{box} + x^{chess} & & \leq 100 \\ \frac{1}{100}x^{box} & - b^{box} & \leq 0 \\ \frac{1}{100}x^{chess} & - b^{chess} & \leq 0 \end{array} \qquad \begin{bmatrix} -1 & -1 & \frac{1}{100} \\ & -100 & 1 \\ & 100 & \end{bmatrix} \qquad \kappa \approx 245$$

Same!

- Curtis-Reid scaling:

$$\begin{array}{rrl} x^{box} + x^{chess} & & \leq 1 \\ x^{box} & - b^{box} & \leq 0 \\ x^{chess} & - b^{chess} & \leq 0 \end{array} \qquad \begin{bmatrix} -1 & -1 & 1 \\ & -1 & 1 \\ & 1 & \end{bmatrix} \qquad \kappa \approx 25$$

Best!

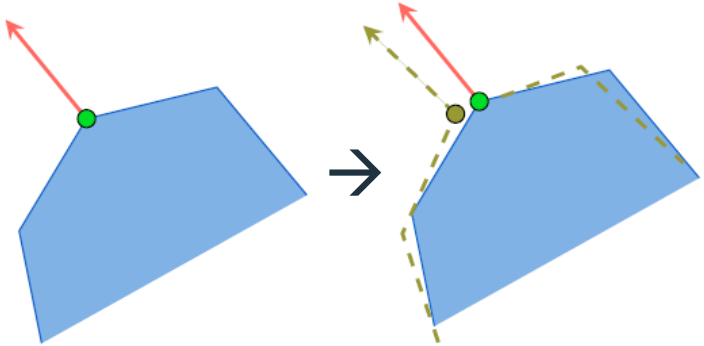
Numerical Issues

Feasibility and Optimality in Floating-Point Solvers

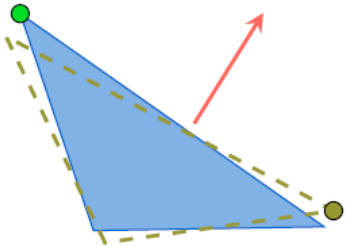
- MIP solvers use numerical tolerances, typically in the range 10^{-6} to 10^{-9}:

Hope:
- Optimal solution with small residual errors is close to an exact optimal solution without violations

But really: exact solution to a **perturbed problem**



Hope



Reality

Nightmare

Sources of Numerical Issues: Large Big-M's

Example:

$$\begin{aligned} &\min y \\ &\text{s.t. } x \geq 1 \\ &\quad x \leq 10^6 y \\ &\quad y \in \{0,1\} \end{aligned}$$

$$\begin{aligned} &\min y \\ &\text{s.t. } x \geq 1 \\ &10^{-6} x \leq y \\ &\quad y \in \{0,1\} \end{aligned}$$

Assuming an absolute tolerance of 10^{-6}, we have that:

- $x = 1, y = 0$ feasible in the scaled problem w.r.t. tolerances, but infeasible in the original
- $x = 1, y = 10^{-6}$ feasible in both, the scaled and original problem w.r.t. tolerances
  - but when you fix y = 0 and reoptimize, the result will be infeasible
- x = y = 1 is exactly feasible

What is the intended behavior?

What is the intended behavior?

$$\begin{aligned} \min\ & y \\ \text{s.t. } & x \geq 1 \\ & 10^{-6}x \leq y \\ & y \in \{0,1\} \end{aligned}$$

- What if this was you original problem?
- What about $0.33333333333333333x = 1, \mathrm{x} \in \mathbb{Z}$?
  - Infeasible?
- What about $0.1x = 1\,, \mathrm{x} \in \mathbb{Z}$?
  - Infeasible?
  - Because $0.1 =_{\mathrm{fp}} 0.1000000000000000005551115\ldots$

Some Guidelines and Tools

Some Guidelines

- Good input, good output
  - Scale data to avoid extreme values: absolute and relative — look at which units to use, e.g. ratio of largest to smallest coefficient $\leq 10^6$ in any row and column
  - Ensure that tolerances make sense relative to the input data
  - Round insignificant, tiny data values to zero
  - Avoid using truncated or single-precision data
- Modelling and solving
  - Try different scaling parameters
  - Try to avoid large big-M's
  - If you don't have a reasonable big-M, use indicator constraints
  - If the objective is a hierarchical combination of multiple objectives: try a sequential approach
    - Many solvers natively support multi-objective optimization!

Note: Poor scaling and imprecise input are neither necessary nor sufficient for numerical problems

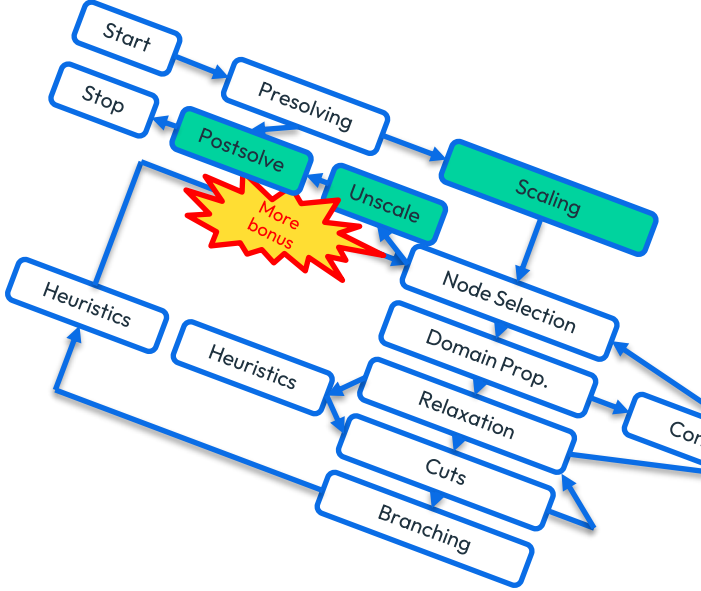
A-priori information on numeric stability

- (some) MIP solvers provide numeric analysis tools
- A priori: spread of matrix, objective, rhs coefficients

```
Coefficient range                    original                    solved
  Coefficients   [min,max] : [ 9.17e-05,  4.20e+02] / [ 1.17e-02,  1.31e+01]
  RHS and bounds [min,max] : [ 3.00e-02,  1.00e+04] / [ 5.00e-01,  4.35e+02]
  Objective      [min,max] : [ 1.00e+00,  7.16e+00] / [ 1.56e-02,  1.86e+02]
```

- comprises effects of presolving AND scaling
- Terminology: Matrix coefficients span seven orders of magnitude in the original problem and three orders of magnitude in the presolved problem
- Typical: By presolving, some of the coefficient spread gets pushed from matrix to objective and vice versa

A posteriori information on numeric stability



- Report on the violations in LP and MIP

```
Final objective                          : 4.082673935185180e+03
  Max primal violation      (abs/rel) : 1.110e-16 / 3.701e-17
  Max dual violation        (abs/rel) : 1.516e-14 / 4.042e-15
  Max complementarity viol. (abs/rel) :       0.0 /       0.0

Max primal violation      (abs/rel) : 9.082e-05 / 9.082e-05
Max integer violation     (abs    ) : 1.016e-06
```

- Violations beyond the threshold can occur due to unscaling and postsolving
- Potentially, report on numeric issues that the solver encountered:

```
Numerical issues encountered:
   Dual failures     :      78 out of      2194 (ratio: 0.0356)
   Primal failures   :       5 out of       247 (ratio: 0.0202)
   Singular bases    :       5 out of     11180 (ratio: 0.0004)
   Nodes w/LP fails  :       9 out of        70 (ratio: 0.1286)
```

numeric information raises awareness!

Adapt models or use non-default controls to prevent issues

Condition Number & Attention Level

- The condition number κ of a matrix A provides a bound on how much a small change in b can affect x, when $Ax = b$
- For a square, invertible matrix A_B

$$\kappa = \|A_B\| \cdot \left\|A_B^{-1}\right\|$$

#errorpropagation

- Sampling the condition number is an optional feature of some solvers

```
Nodes kappa stable     :      3757 (ratio: 0.0051)
Nodes kappa suspicious :      8476 (ratio: 0.0115)
Nodes kappa unstable   :    723171 (ratio: 0.9831)
Nodes kappa ill-posed  :       193 (ratio: 0.0003)
Largest kappa seen     : 4.959805e+14
Kappa attention level  : 0.2953
```

- Summarized in a single attention level from 0.0 (all stable) to 1.0 (anything goes)
  - Non-default feature: Expensive

Numerical information – attention level

- In Xpress, set MIPKAPPAFREQ=*n* to sample κ every *n* levels of the tree search. Use *n*=1 to sample every node, including between cutting
- Calculated kappa values grouped into categories:

stable	$\kappa \leq 10^7$	Numerical issues are not expected
suspicious	$10^7 < \kappa \leq 10^{10}$	Occasional minor issues
unstable	$10^{10} < \kappa \leq 10^{13}$	Numerical issues expected
ill-posed	$10^{13} < \kappa$	Expect serious numerical issues
attention level	Aggregation of κ into value from 0 [*good*] to 1 [*ill-posed*]	

- $att = \frac{0 \cdot n_{stable} + 0.01 \cdot n_{suspicious} + 0.3 \cdot n_{unstable} + 1 \cdot n_{ill-posed}}{n_{samples}}$

What can we do, as a modeler?

One possible tool: MIP-DD: A Delta-Debugger for MIP

- Goal: try to reproduce unwanted behavior in a MIP solver, e.g., a numerical issue, on a significantly smaller and easier to analyze MIP
- inspired by delta debugging heavily used in the SAT and SMT community (Zeller 1999, Brummayer and Biere 2009, Niemetz and Biere 2013, Kaufmann and Biere 2022, Paxian and Biere 2023)
- open-source package MIP-DD available at *github.com/scipopt/mip-dd*
  - very successful in increasing the number of bug fixes in the last SCIP releases
  - for details see Hoen, Kampp, G. 2024: “MIP-DD: A Delta Debugger for Mixed Integer Programming Solvers”, *arxiv.org/abs/2405.19770v1*

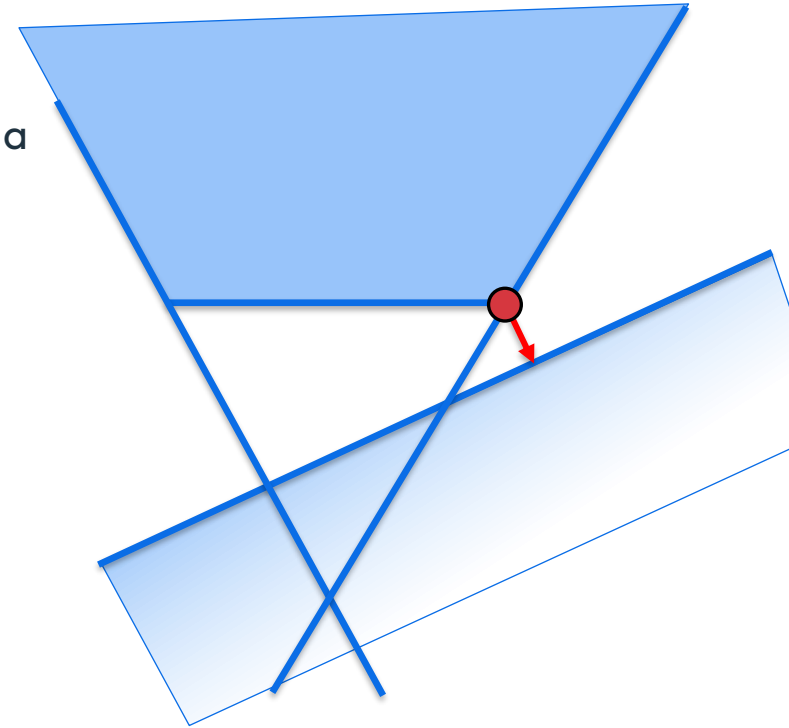
Repairing Infeasibility: RepairInfeas (aka FeasOpt aka FeasRelax)

Goal: relax problem until feasible / minimize violations in a genuinely infeasible probme

- relax variable bounds and constraint sides with penalty slack variables
- Phase I: minimize total (weighted) violation

$$\min \sum |y_i|$$
$$Ax + Ey = b$$
$$y \in \mathbb{R}$$

- Phase II (optional): reoptimize the original objective among all minimum-violation solutions
- Comparison to IIS: Different approach to find the infeasibility bottleneck
- Even for feasible problems: Decent heuristic



What can we do, algorithmically?

Goal: reduce or remove primal, dual and integrality violations in incumbents and final solution by:

- performing extra simplex iterations with tighter tolerances
- pushing fractional integer variables out of the basis when possible
- recomputing in quad precision
  - Or arbitrary precision over rationals, e.g., via GMP
- performing additional branches to force integer variables to integer values
- fixing integer variables and solving the remaining LP

→ All of this comes at an expense, and those procedures are usually summarized as refinement.
Let's discuss some of the options..

Solution refinement:

Heuristically eliminate violations

MIP Solution Refinement

- Automatic refiner of every new incumbent MIP solution to reduce:
  - Violations of bounds and constraints
  - Violations of integrality constraints
- Remove small violations introduced by scaling and presolve.
- For MIP-feasible solutions, triggers extra LP iterations with tighter tolerances if violations exist

```
     Node     BestSoln    BestBound   Sols Active  Depth     Gap     GInf    Time
*     197  15183.85668  15227.15725      8      1     19    0.28%       0       3
      200  15183.85668  15220.67819      8      1      8    0.24%       5       3
  *** Search completed ***     Time:     4 Nodes:         253
Number of integer feasible solutions found is 8
Best integer solution found is  15183.85668
Best bound is  15183.85668
Refining MIP solution (1.116e-06 fractionality 3.617e-05 abs infeasiblity)
P     253  15183.86782  15183.85668      9      0     10    0.00%       0       4
Refined MIP solution (0.0 fractionality 9.890e-07 abs infeasiblity)
```

- On by default and automatic

Iterative Refinement for LP

Recall

Basic solution: primal–dual

- W.l.o.g., let $\text{rank}(A) = m < n$ and consider the computational form

$$\min\{c^T x \mid Ax = b, x \geq 0\} = \max\{y^T b \mid y^T A \leq c, y \text{ free}\}$$

- For every vertex there is a non-singular $m \times m$ sub-matrix B of A

$$A = [\, B \mid N \,]$$

- and the corresponding basic solution is given by

$$x_B = B^{-1} b, \qquad x_N = 0, \qquad y^T = c_B^T B^1$$

- In practice: No explicit inversion, but solve linear systems $Bx_B = b$ and $B^T y = c_B$ by factorization $B = LU$
- Floating-point arithmetic results in residual errors $\hat{b} = b - Bx_B \neq 0$ and $\hat{c} = c_B - B^T y \neq 0$

Iterative Refinement

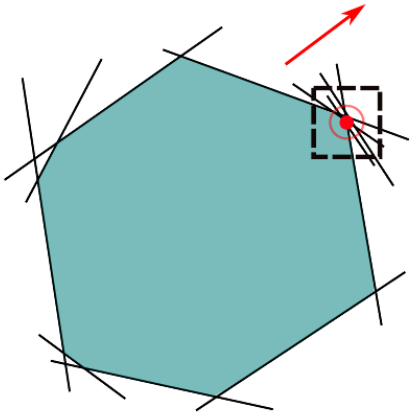
[Applegate et al. 2007, Gleixner, Steffy, Wolter 2012, 2016, 2020, Gleixner, Nicolas-Thouvenin, Eifler 2024]

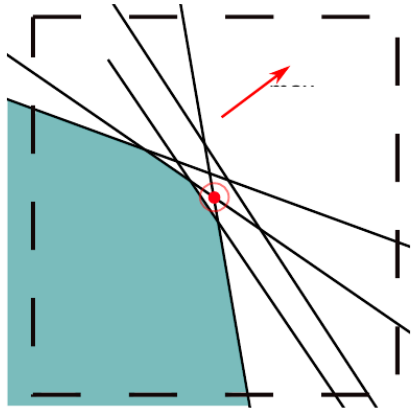
- Iterative refinement:
  - Solves are performed in floating-point arithmetic (fast).
  - Residuals are computed in extended-precision or exact arithmetic
  - Residual problem is constructed and solved (in floating point arithmetic)
    - Can be warm-started
  - Correction terms are added (in extended/exact arithmetic)
  - Only very few computations in slow extended/exact arithmetic

Iterative Refinement: Visualization

$$x \mapsto \Delta_{\text{prim}}(x - \hat{x})$$
$$y \mapsto \Delta_{\text{dual}}(y - \hat{y})$$

$$P \left\{ \begin{array}{ll} \min & c^T x \\ \text{s.t.} & Ax = b \\ & x \geqslant 0 \end{array} \right. \qquad \left. \begin{array}{ll} \min & \Delta_{dual} \hat{c}^T x \\ \text{s.t.} & Ax = \Delta_{prim} \hat{b} \\ & x \geqslant -\Delta_{prim} \hat{x} \end{array} \right\} \hat{P}$$



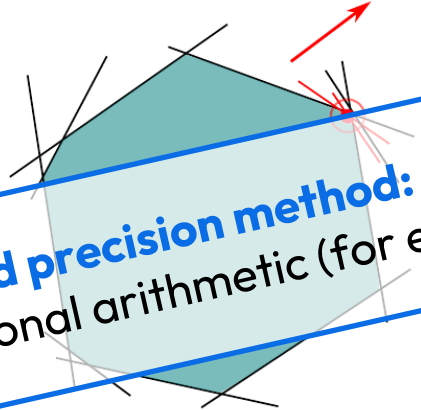


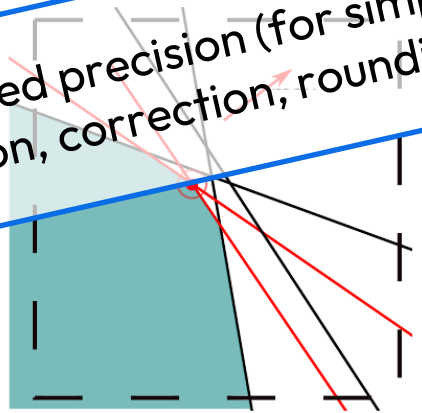
Iterative Refinement: Visualization

$$x \mapsto \Delta_{\text{prim}}(x - \hat{x})$$
$$y \mapsto \Delta_{\text{dual}}(y - \hat{y})$$

$$P \left\{ \begin{array}{lll} \min & c^T x & \\ \text{s.t.} & Ax = b & \\ & x \geqslant 0 & \end{array} \right. \qquad \left. \begin{array}{lll} \min & \Delta_{dual} \hat{c}^T x & \\ \text{s.t.} & Ax = \Delta_{prim} \hat{b} & \\ & x \geqslant -\Delta_{prim} \hat{x} & \end{array} \right\} \hat{P}$$

$$y \mapsto y/\Delta_{\text{dual}} + \hat{y}$$
$$x \mapsto x/\Delta_{\text{prim}} + \hat{x}$$





Hybrid precision method: double + extended precision (for simplex)
+ rational arithmetic (for error computation, correction, rounding, LU)

Fractionality refinement to eliminate integer violations

MIP fractionality refinement

- Reduce the fractionality of integer variables in any returned solution.
- Integrated with the MIP branch-and-bound search:
  - Pushes fractional integer variables out of the basis when possible
  - Performs additional branches to force integer variables to integer values.
  - Applies an optional ‘fixing’ test to verify that integer variables can be fixed feasibly.
    - In the original problem: Fix all integer variables to exact values and reoptimize LP.
    - Use case: rolling horizon: Integer variables need to be fixed to the returned values without having to worry about infeasibility or relaxing tolerances.
- Performance (with ‘fixing’ test):
  - Solutions with any fractionality dropped from 28.5% to 0.2%
  - Performance change: +7% → off by default

Refinement summary

- Different refinement strategies take care of different aspects of numeric violations
  - Can be combined
  - In particular, Iterative Refinement and Fractionality Refinement are completely orthogonal
- They all come at a cost
  - Only the heuristic solution refinement is applied by default in Xpress
- They are all curing symptoms, not causes
- Floating-point precision poses a natural boundary to all approaches

All glory to GPUs and the age of AI?

- While it is true that some high-end GPUs do have good support for fp64
  - Most GPUs are still optimized for fp16 and fp32
  - With the some trends going to fp8 or fp4
- fp4 computation has a factor 500 higher throughput than FP64 on Nvidia's HGX B200…

Precision	Peak throughput
FP64	37 TFLOP/s
FP32	75 TFLOP/s
FP4	18,000 TFLOP/s

- Exhaustive list of all fp4-representable numbers:
  - $x \in \{-6, -4, -3, -2, -1.5, -1, -0.5, \pm 0, +0.5, +1, +1.5, +2, +3, +4, +6\}$
  - $0.3 =_{fp4} 0.5, \quad 10000000000 =_{fp4} 6$
- Fancy new hardware might not boost our optimization capabilities
  - All glory to optimization software development!

Thank You!

PD Dr. Timo Berthold
Private Lecturer @ TU Berlin
Director, Mixed-Integer Optimization @ FICO