Exploring the Numerics of Branch-and-Cut for Mixed Integer Linear Optimization

Matthias Miltenberger **Ted Ralphs** **Daniel E. Steffy**

Zuse Institute Berlin · miltenberger@zib.de
Lehigh University · ted@lehigh.edu
Oakland University · steffy@oakland.edu

OR 2017
7 September 2017

ZIB

MODAL
Mathematical Optimization and Data Analysis Laboratories

Motivation and Leading Questions

- How to measure numerical stability of a MIP solver?
- How does the stability evolve during optimization?
- Can we keep numerical stability under control?
- What is the influence of cutting planes and branching?

Outline

Outline

Numerical Stability

How does the output of an algorithm change after small changes to its input?

- How do numerical errors accumulate?
- Up to which precision can the result be trusted?

Branch-and-Bound for MIP

1. solve LP relaxation
2. add cutting planes
   - re-solve LP relaxation including new inequalities
3. branch on a fractional variable
   - repeat process on both resulting sub-problems

cutting:

- should be preferred to branching
- no additional sub-problems, only one re-optimization
- often struggles from numerical difficulties ("parallel" cuts)
- cut quality degrades over time (tailing off effect)

branching:

- branching disjunctions represent "best cuts possible"
- guaranteed to be orthogonal to each other
- only have one nonzero component (very sparse)
- drawback: effectively doubles the problem

Measuring the Stability

- We use the condition number of the optimal basis of the LP relaxation
- This basis is used for generating Gomory cutting planes
- This basis determines how accurate the LP solution is ($x_B = A_B^{-1} b$)

- Excellent and detailed reference for condition and numerical stability:

  P. Bürgisser and F. Cucker
  Condition - The Geometry of Numerical Algorithms
  Vol. 349. Grundlehren der math. Wissenschaften. Springer, 2013.

Outline

Computing κ

- Always consider the condition number wrt $\|\|_2$:

$$\kappa := \|A\|_2 \cdot \|A^{-1}\|_2$$

$$\text{where } \|A\|_2 := \max_{\|x\|_2=1} \|Ax\|_2$$

- Computing κ can be expensive and is as stable as κ itself
- Accuracy is not too important
- Sufficient to inspect $\log_{10}(\kappa)$

Computing κ

Power method:

- perform subsequent multiplications with A and A^T until convergence to largest singular value σ_1 of A
- repeat with the inverse A^{-1} to get smallest singular value σ_m
- $\kappa = \sigma_1 / \sigma_m$

- Provides a precise value, but requires expensive computation
- Faster approximations to estimate the condition number are available

Experimental Setup

Software and tools:

- SCIP Optimization Suite
  `http://scip.zib.de`
- GrUMPy - Graphics for Understanding Mathematical Programming in Python
  `http://github.com/coin-or/GrUMPy`

Test set:

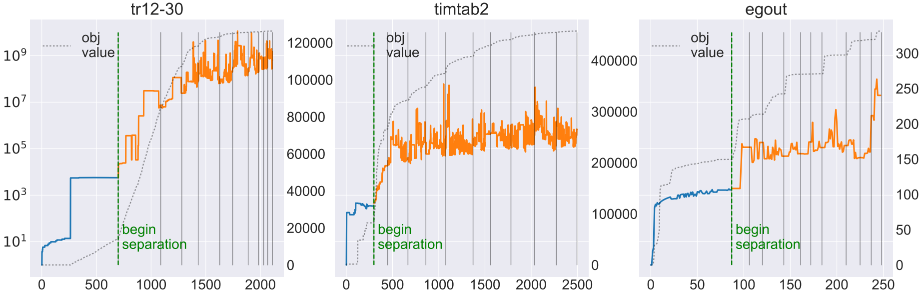
- combined benchmark sets of all three MIPLIBs (2003, 3, and 2010)

Settings:

- aggressive Gomory separator: generate cuts at all nodes and add more cuts (default SCIP only separates cutting planes at the root node)
- deactivate other cut generators
- time limit: 1h
- node limit: 10 000

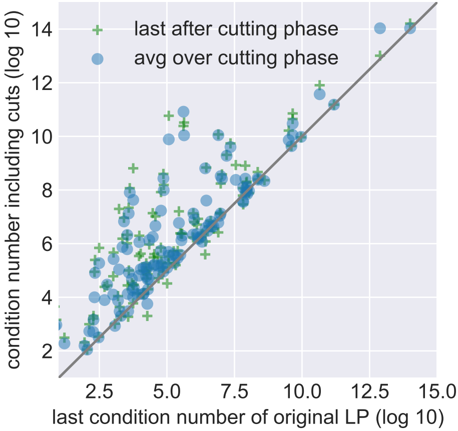
Outline

Condition Numbers during Root Simplex Optimization



- Dual simplex starting with slack basis ($A_B = I$):
  - κ increases quickly from the initial 1.0
  - almost monotone increase until characteristic κ is reached
  - adding cuts often increases κ significantly
  - tailing off effect can be seen

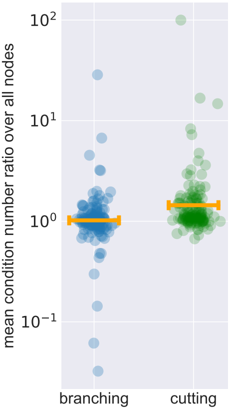
Aggregated Root Cutting Results



- Condition number increase due to cuts in the root node
- Mean and final difference over cutting phase

Outline

Branching and Cutting



- Changes in condition number wrt branching and cutting
- Branching:
  Compare parent node after cutting and child node before cutting
- Cutting:
  Compare before and after cutting at each node

- Branching does not significantly increase or decrease κ
- Cutting leads to an increased κ over all nodes

Condition Numbers in the Tree

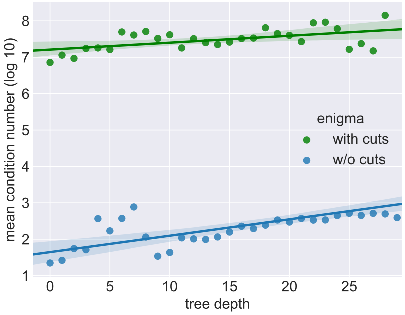
Different approach:

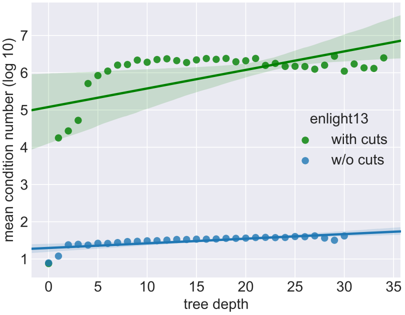
- Compare a run using cutting with a pure branch-and-bound optimization

Condition Numbers in the Tree

Different approach:

- Compare a run using cutting with a pure branch-and-bound optimization

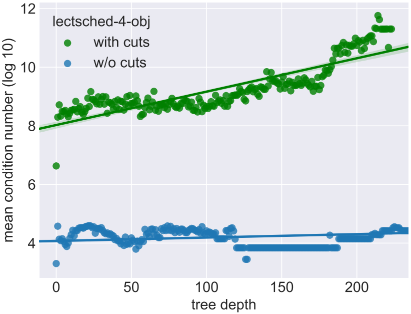


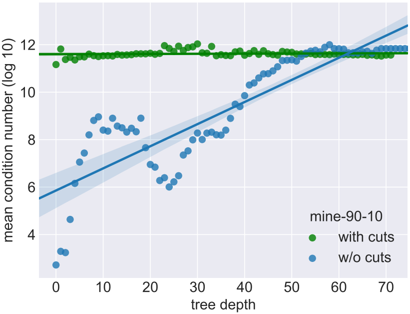


Condition Numbers in the Tree

Different approach:

- Compare a run using cutting with a pure branch-and-bound optimization

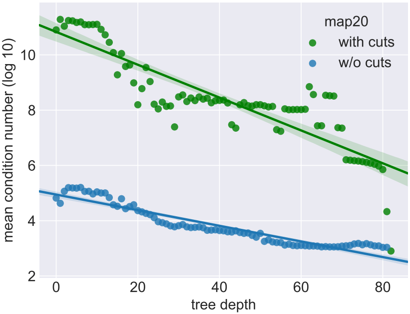


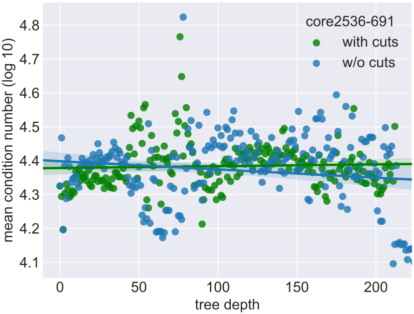


Condition Numbers in the Tree

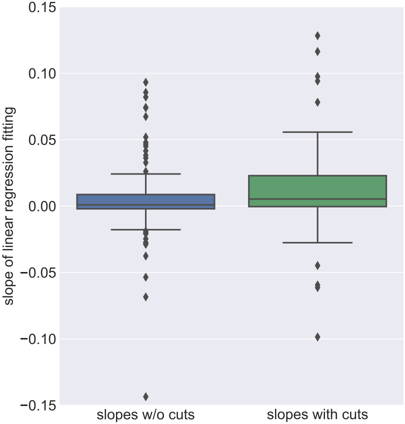
Different approach:

- Compare a run using cutting with a pure branch-and-bound optimization





Aggregated Results



- regression slopes for all instances
- tree depth of at least 5

- no significant degradation or stabilization through branching
- clearly positive slope when adding cuts

- no satisfactory feature for "good" cuts found yet

Outline

Conclusion and Outlook

- Intuitive assumptions are not always correct
- Numerical analysis is often more demanding than expected
- More evaluations and experiments are necessary
- Try different stability measures / different condition numbers
- Test other solvers (both LP and MIP)
- Find better cut selection and filtering
- Gain a more holistic approach to algorithmic control

Conclusion and Outlook

- Intuitive assumptions are not always correct
- Numerical analysis is often more demanding than expected
- More evaluations and experiments are necessary
- Try different stability measures / different condition numbers
- Test other solvers (both LP and MIP)
- Find better cut selection and filtering
- Gain a more holistic approach to algorithmic control

Thank you for your attention!