LP Solution Polishing to improve MIP Performance

Matthias Miltenberger

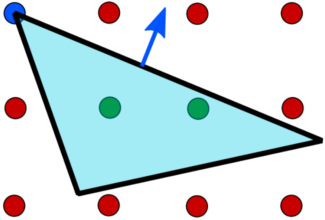
Zuse Institute Berlin

ICCOPT 2016 - Tokyo, August 9^{th} 2016

Mathematical Optimization and Data Analysis Laboratories

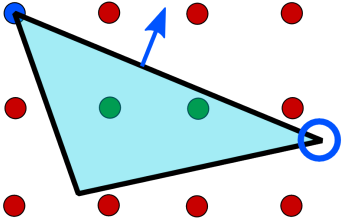
Motivation

- Solving a MIP $\min\{c^T x \mid Ax = b,\ x_i \in \mathbb{Z} \text{ for } i \in I\}$ involves solving many LPs as linear relaxations
- LP solutions are rarely unique



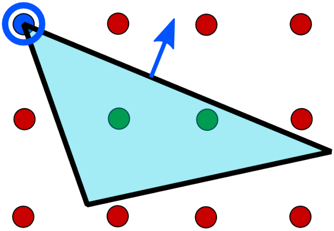
Motivation

- Solving a MIP $\min\{c^T x \mid Ax = b,\ x_i \in \mathbb{Z} \text{ for } i \in I\}$ involves solving many LPs as linear relaxations
- LP solutions are rarely unique



Motivation

- Solving a MIP $\min\{c^T x \mid Ax = b,\ x_i \in \mathbb{Z} \text{ for } i \in I\}$ involves solving many LPs as linear relaxations
- LP solutions are rarely unique



- **How to find the best one?**

Outline

LP Solution Polishing to improve MIP Performance

1. Introduction
   - Dual Degeneracy
   - Performance Variability
2. Related Work
3. Solution Polishing
   - Integrality of Variables
4. Computational Results
   - SCIP Optimization Suite
5. Conclusion and Outlook

Dual Degeneracy

- Two types of degeneracy in LP:
  - primal: multiple bases defining one vertex of the polyhedron
  - dual: facet of the polyhedron parallel to the objective function
- Most (practical) problems are primal and dual degenerate
- Degeneracy is the most prominent cause of MIP performance variability

Performance Variability

- Performance of a MIP solver may vary drastically when the data changes
  - change row and column order
  - use a different random seed
  - implement a different tie breaker
  - ...
- Several causes for variability
  - different LP optima are probably most influencial
- Explained in
  - *Danna, E.*: **Performance variability in mixed integer programming** MIP Workshop (2008)
  - *Koch, T., et al.*: **MIPLIB 2010**, Math. Program. Comp. (2011)

k-Sample

Improving branch-and-cut performance by random sampling
M. Fischetti, A. Lodi, M. Monaci, D. Salvagnin, A. Tramontani
Math. Program. Comp. (2016), Vol. 8

1. perform preprocessing on one core
2. solve root LP on $k-1$ cores with different random seeds
   - collect primal solutions and generated cuts
3. complete solving process on one core with yet another random seed

- previously collected information helps to improve the performance
- performance variability is reduced
- contained in the latest CPLEX release for $k = 3$

Pure Cutting Plane Algorithm

Lexicography and degeneracy: Can a pure cutting plane algorithm work?
A. Zanette, M. Fischetti, E. Balas
Math. Program. (2011) Vol. 130

- answer: Yes, it can!
- ...when choosing the correct LP basis
  - cutting plane method adds many cuts (almost) parallel to objective
  - use the lexicographic dual simplex to deal with high dual degeneracy
  - or modify the objective to mimic the lexicographic behavior
- standard cutting plane approach suffers from bad numerical stability

IBM CPLEX Patent

LP relaxation modification and cut selection in a MIP solver
T. Achterberg
US Patent (2011)

- similar to k-Sample, another optimal LP basis is constructed
  - fix some non-basic variables and modify the objective
  - use new basis to collect more information, e.g. for cuts
- implemented in CPLEX

LP Solution Polishing to improve MIP Performance

General Idea of Solution Polishing

- Dual simplex algorithm terminates at first primal feasible, optimal basis
- Perform additional polishing steps altering this basis
- Reminder:
  - Basic indices: B, non-basic indices: N
  - Nonbasic variables are on their bound:
    $x_N = 0$ or $x_N = u$
  - Basic variables can be between bounds (depending on x_N)
    $x_B = A_B^{-1}(b - A_N x_N)$

- Polishing steps are primal iterations (to preserve feasibility):

1. find non-basic indices to enter the basis ($\hat{=}$ pricing step)
   - choose one with zero reduced costs to stay on optimal hyperplane
2. try pivoting and check whether leaving index is good ($\hat{=}$ ratio test)
3. repeat

Two Objectives

1. Decrease fractionality $\hat{=}$ push integer variables out of basis
   - less branching candidates
   - hopefully *closer* to an integer feasible solution

2. Increase fractionality $\hat{=}$ push integer variables into basis
   - may generate better cuts (basis matrix contains less slack)

Integrality of Variables

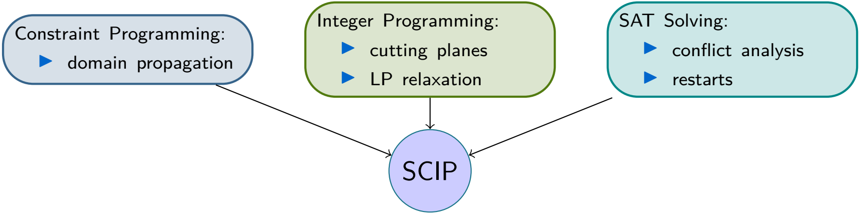
- Usually, LP solver has no knowledge of integrality
- Unlucky scenario:
  1. push continuous variable out of basis
  2. remaining basic integer variables are moved away from bounds
- Remedy:
  1. transfer information about integer variables to LP solver
  2. push only basic integer variables to their bounds

LP Solution Polishing to improve MIP Performance

Setup

- Test set: MIPLIB 3 + MIPLIB 2003 + MIPLIB 2010, 168 instances
- All runs sequentially on one core
- SCIP Optimization Suite 3.2.1 with modifications

What is SCIP?

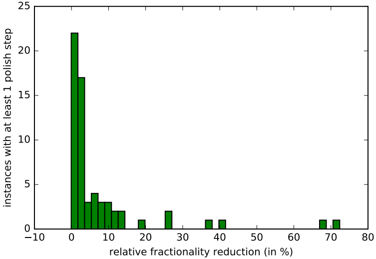


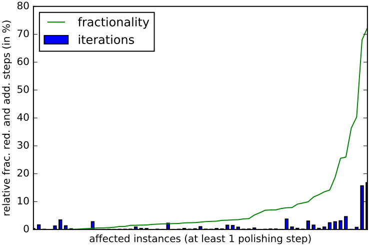
SCIP (Solving Constraint Integer Programs) ...

- has a modular structure via plugins,
- provides a full-scale global MINLP solver,
- part of the SCIP Optimization Suite (incl. SoPlex, ZIMPL, GCG, and UG),
- is free for academic purposes,
- and is available in source-code under `http://scip.zib.de`

Root Node Fractionality

- Compare fractionality before and after solution polishing
- Only root LP is solved
- With integrality information in SoPlex

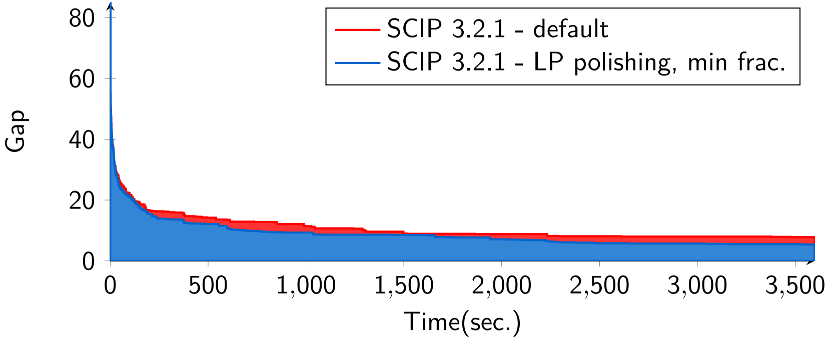




number of affected instances (of 168):	63
number of instances with a reduction of more than 5%:	22
mean percentage reduction of fractionality:	7.74
mean percentage of additional steps:	1.29

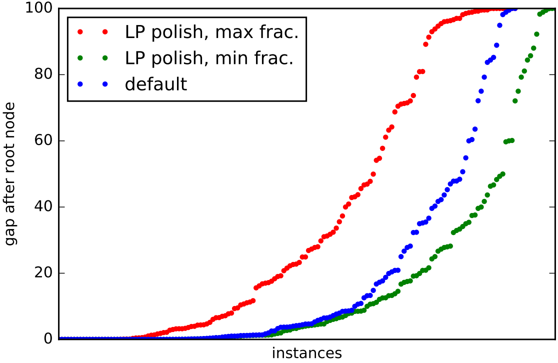
Impact on MIP Performance

- Polishing reduces number of nodes by 2-3 %
- Transferring integrality information is expensive
- Mean primal integral improvement: 38481.0 → 31316.1



Root Node Cut Performance

- Increasing fractionality leads to a high increase in nodes and deteriorates the root gap
- Reducing fractionality leads to a smaller root gap



LP Solution Polishing to improve MIP Performance

Drawbacks of our Implementation

- Polished LP optimum is still not unique
  - maximum or minimum of fractionalities is not guaranteed

Possible improvements:

- Implement a (more expensive) technique to find the best basis
- Transfer of integrality information needs a more efficient implementation
  - use integrality information also in other parts of the LP solver
- Make use of several optimal bases

Conclusion and Outlook

- Solution polishing is cheap to apply
  - ...when used to reduce fractionality
  - ...when transfer of integrality information is improved
- Does not modify the LP problem data
- Already provides promising results concerning fractionality and gap reduction
- No effect on reducing performance variability observed yet

- More refinement and tuning possible
  - especially regarding fractionality increase
  - polishing could be applied more selectively
- Reduce performance variability by LP solution polishing

Conclusion and Outlook

- Solution polishing is cheap to apply
  - ...when used to reduce fractionality
  - ...when transfer of integrality information is improved
- Does not modify the LP problem data
- Already provides promising results concerning fractionality and gap reduction
- No effect on reducing performance variability observed yet

- More refinement and tuning possible
  - especially regarding fractionality increase
  - polishing could be applied more selectively
- Reduce performance variability by LP solution polishing

Thank you for your attention!
ご清聴ありがとうございました