SCIP Beyond 8.0

Ksenia Bestuzheva, bestuzheva@zib.de

Workshop on Future Algorithms and Applications,
Berlin, Germany

September 29, 2023

ZIB ZUSE INSTITUTE BERLIN
MODAL FORSCHUNGSCAMPUS
SPONSORED BY THE Federal Ministry of Education and Research

The SCIP Optimization Suite

A toolbox for generating and solving MILPs, MINLPs, and CIPs:

- **SCIP** : MIP solver and CIP framework,
- **SoPlex**: LP solver,
- **PaPILO**: parallel presolver for integer and linear optimization,
- **ZIMPL**: mathematical programming language,
- **UG**: parallel framework for MIPs,
- **GCG**: generic branch-cut-and-price solver,
- **SCIP-SDP**: extension for solving MISDPs,
- **SCIP-Jack**: extenstion for solving Steiner tree and related problems.

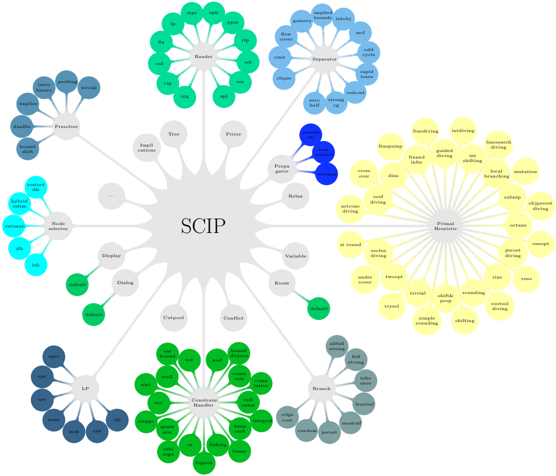
The SCIP Optimization Suite

A toolbox for generating and solving MILPs, MINLPs, and CIPs:

- **SCIP : MIP solver and CIP framework**,
- **SoPlex**: LP solver,
- **PaPILO**: parallel presolver for integer and linear optimization,
- **ZIMPL**: mathematical programming language,
- **UG**: parallel framework for MIPs,
- **GCG**: generic branch-cut-and-price solver,
- **SCIP-SDP**: extension for solving MISDPs,
- **SCIP-Jack**: extenstion for solving Steiner tree and related problems.

SCIP (Solving Constraint Integer Programs)

- Provides a **full-scale MILP and MINLP solver**,
- is **constraint based**,
- is a **branch-cut-and-price framework**,
- incorporates
  - **MILP features** (cutting planes, LP relaxation),
  - **MINLP features** (spatial branch-and-bound, OBBT)
  - **CP features** (domain propagation),
  - **SAT-solving features** (conflict analysis, restarts),
- has a modular structure via **plugins**,
- is **licensed under Apache 2.0**,
- and is **available in source-code** under `https://scipopt.org` !

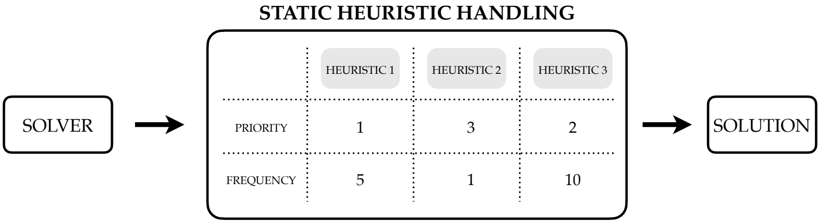


Overview of Recent Developments

- **Primal heuristics:**
  - Online learning for scheduling heuristics
  - Feasibility jump
  - Indicator diving
- **Cutting planes:**
  - Lift-and-project cuts
  - Lagromory cuts
  - Improved implicit product filtering for RLT cuts
  - Monoidal strengthening of intersection cuts for MIQCPs
- **Branching** via cutting plane selection
- Pseudo-Boolean **conflict analysis**
- Updates to the **exact solving** framework for MILPs
- Improvements to **symmetry handling**
- New and improved **interfaces**
  - SCIP will be able to call HiGHS (`https://highs.dev`) as an LP solver
  - New interface: Rust
  - Improvements to the Julia interface

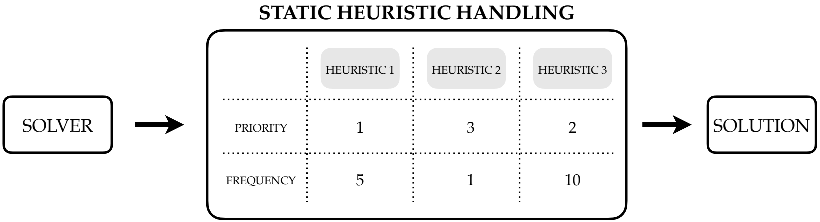
Scheduling Primal Heuristics: Motivation

- MIP solving executes a broad range of primal heuristics for finding good solutions.
- The settings of heuristics are static with strict working limits.



Scheduling Primal Heuristics: Motivation

- MIP solving executes a broad range of primal heuristics for finding good solutions.
- The settings of heuristics are static with strict working limits.



Question

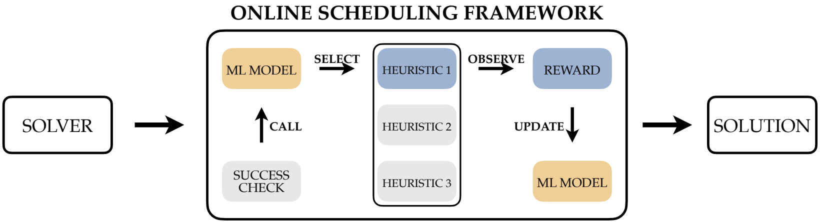
Static settings derived from heterogeneous benchmark test sets might not yield best performance since performance of heuristics is highly instance-dependent.

Idea

Make the execution of heuristics adaptive by learning which heuristics perform well for the current instance.

Scheduling Primal Heuristics: Online Learning

- A Chmiela, A Gleixner, P Lichocki, S Pokutta *Online Learning for Scheduling MIP Heuristics*
- Online scheduling framework manages (i) selection and (ii) working limits by learning from past observations.
- A novel reward function catches heuristics' impact on the solving process beyond simply finding new solutions.
- General framework enables us to schedule complex heuristics of different types simultaneously.



- Consistent node reductions over the MIPLIB 2017 Benchmark set.
- Speedup of 4% for instances that take at least 1000s to solve.

The Feasibility Jump Heuristic

B. Luteberget, G. Sartor *Feasibility Jump: an LP-free Lagrangian MIP heuristic*

- 1st place: MIP 2022 Computational Competition

Computational results on the MIPLIB benchmark:

- High success rate: Finds feasible solutions for over 30% of the instances
- Between 3 and 8% faster to the first feasible solution on average
- On average slightly slower

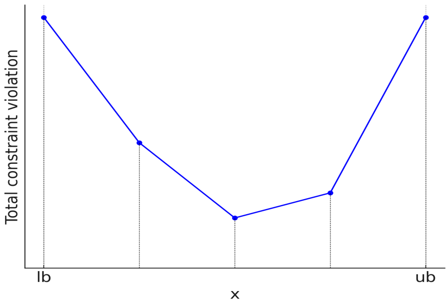
The Feasibility Jump Heuristic

It's a Lagrangian heuristic method: $\min c^T x$ s.t. $Ax \leq b \quad \rightarrow \quad \min \mathcal{L}(x, \lambda) = \lambda^\top (b - Ax)$

- Start with an incumbent vector x^*
- Choose a single variable
- "Jump" to the value that minimizes the weighted sum of constraint violations (taking integrality into account)
- The neighborhood, defined by scores, is updated after each jump "lazily"
- Score: decrease in total constraint violation

$$\max\{\lambda^\top (b - Ax), 0\} - \max\{\lambda^\top (b - Ax^*), 0\}$$

  (i.e. violation before the jump - violation after the jump)
- Update weights in the Lagrangian function



Indicator Diving Heuristic

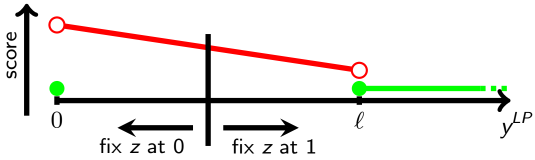
- A diving heuristic simulates a depth-first search.
  It alternates between tightening variable bounds and solving LP relaxations.

- Indicatordiving is a diving heuristic with focus on (unbounded) semi-continuous variables.

- Semi-cont. variables $y \in \{0\} \cup [\ell, u]$ with $u \in \mathbb{R}_+ \cup \{\infty\}$
  are modeled with a binary indicator variable: $z = 0 \rightarrow y = 0$
  $z = 1 \rightarrow y \geq \ell$

- During the diving process z is fixed depending on the LP solution value y^{LP} of the semi-cont. variable y:

$$y^{LP} < 0.5\,\ell \rightarrow z = 0$$
$$y^{LP} \geq 0.5\,\ell \rightarrow z = 1$$



Lift-and-project and Lagromory Cuts for MILPs

Lift-and-project cuts:

- Based on Bonami's 2012 work "On optimizing over lift-and-project closures"
- Goal: find cuts for the convex hull of a disjunction (e.g. branching)
- A trivial normalization constraint (NC) accounts for coefficient scaling
- NC $\rightarrow$ reduce the cut generating LP (CGLP) based on certain inferences
- Dualize the reduced CGLP $\rightarrow$ membership LP
- Solve membership LP, obtain dual information, and generate a cut

Lift-and-project and Lagromory Cuts for MILPs

Lift-and-project cuts:

- Based on Bonami's 2012 work "On optimizing over lift-and-project closures"
- Goal: find cuts for the convex hull of a disjunction (e.g. branching)
- A trivial normalization constraint (NC) accounts for coefficient scaling
- NC $\rightarrow$ reduce the cut generating LP (CGLP) based on certain inferences
- Dualize the reduced CGLP $\rightarrow$ membership LP
- Solve membership LP, obtain dual information, and generate a cut

Lagromory cuts:

- Based on Fischetti and Salvagnin's 2011 work "A relax-and-cut framework for Gomory mixed-integer cuts"
- In the root node consider Lagrangian dual problem, add GMI cuts as soft constraints
- GMI cuts 'tilt' the objective $\rightarrow$ explore nearby bases, add more GMI cuts
- Solve this problem iteratively by updating the Lagrangian multipliers
- Select cuts from the set of all thus generated GMI cuts to add to cut pool

Branching via Cutting Plane Selection: Motivation

Many cutting planes are derived from disjunctions. Most commonly from split disjunctions.

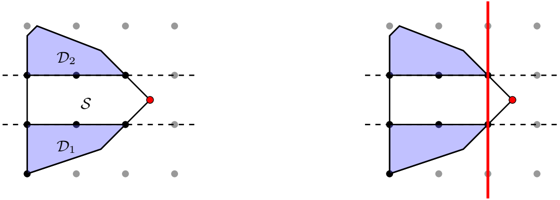
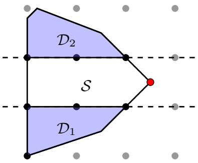


Figure: (Left) An example (simple) split. (Right) An example (simple) split cut.

Branching via Cutting Plane Selection: Motivation

Many cutting planes are derived from disjunctions. Most commonly from split disjunctions.



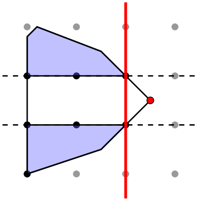


Figure: (Left) An example (simple) split. (Right) An example (simple) split cut.

Idea

Make branching decisions based on history of cut strength from similar disjunctions.

Branching via Cutting Plane Selection: Details

- Branching rule-1
  - Generate Gomori Mixed-Integer cuts from tableau rows corresponding to fractional basic variables.
  - Select a branching candidate that generated the cut with largest efficacy.

Branching via Cutting Plane Selection: Details

- Branching rule-1
  - Generate Gomori Mixed-Integer cuts from tableau rows corresponding to fractional basic variables.
  - Select a branching candidate that generated the cut with largest efficacy.
- Branching rule-2
  - Similar to above, but based on weak-GMI cuts.

Branching via Cutting Plane Selection: Details

- Branching rule-1
  - Generate Gomori Mixed-Integer cuts from tableau rows corresponding to fractional basic variables.
  - Select a branching candidate that generated the cut with largest efficacy.
- Branching rule-2
  - Similar to above, but based on weak-GMI cuts.
- Branching rule-3
  - Generate GMI cuts similar to above.
  - Calculate the average cut strength.
  - Incorporate this as an additional metric into SCIP's default branching scoring function.
  - Select a branching candidate based on the cut with largest score.

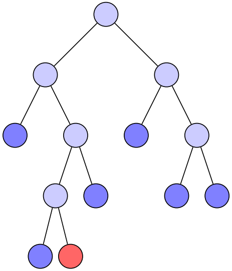
Results on MIPLIB 2017 benchmark:

- Rule 3 affects 67% of instances
- 4% reduction in mean time on affected instances

Conflict Analysis: Brief Introduction

When MIP solving reaches an infeasible subproblem, analyze the infeasibility to

- extract a shorter reason
- that prunes other parts of the tree
- and also helps in backtracking

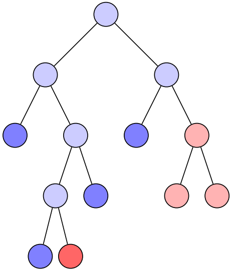


- Generate a bound disjunction explaining the infeasibility similar to SAT solving.
  - Operates on clauses and not on the more expressive linear constraints
- Generate the Farkas constraint $(y^T A)x \geq y^T b$ for infeasible LPs.
  - May be dense with bad numerics

Conflict Analysis: Brief Introduction

When MIP solving reaches an infeasible subproblem, analyze the infeasibility to

- extract a shorter reason
- that prunes other parts of the tree
- and also helps in backtracking

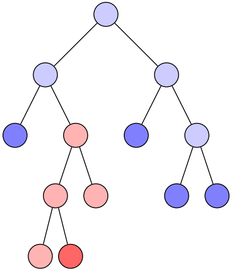


- Generate a bound disjunction explaining the infeasibility similar to SAT solving.
  - Operates on clauses and not on the more expressive linear constraints
- Generate the Farkas constraint $(y^T A)x \geq y^T b$ for infeasible LPs.
  - May be dense with bad numerics

Conflict Analysis: Brief Introduction

When MIP solving reaches an infeasible subproblem, analyze the infeasibility to

- extract a shorter reason
- that prunes other parts of the tree
- and also helps in backtracking



- Generate a bound disjunction explaining the infeasibility similar to SAT solving.
  - Operates on clauses and not on the more expressive linear constraints
- Generate the Farkas constraint $(y^T A)x \geq y^T b$ for infeasible LPs.
  - May be dense with bad numerics

Generalized Resolution Conflict Analysis

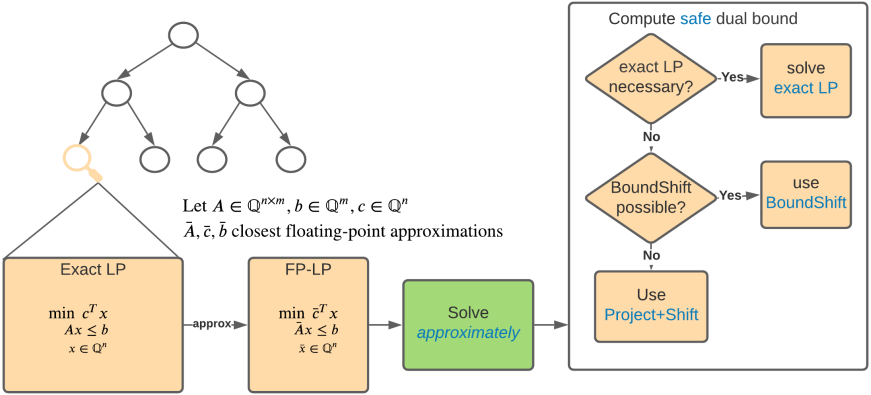
Goal: Given the infeasible system

$$\begin{cases} a^\top x \geq a_o & \text{(reason: for propagating } x_i \geq \alpha) \\ b^\top x \geq b_o & \text{(conflict: infeasible for } x_i \geq \alpha) \\ x \in [\ell', u'] \subset [\ell, u]. \end{cases}$$

Can we find a single constraint that proves the infeasibility?

- G Mexi, T Berthold, A Gleixner, J Nordstroem *Improving Conflict Analysis in MIP Solvers by Pseudo-Boolean Reasoning*
- Applicable to pure binary constraints
- "massage" reason constraint until it propagates x_i tightly.
  - Weakening: Set variables at global bounds **and**
  - Stengthening: MIR, CG, Coef. Tightening
- "Resolve" x_i (add the two constraints so that x_i disappears)

Exact MILP Solving: Hybrid Branch-and-Bound



Implemented in SCIP: more details in Cook, Koch, Steffy, Wolter 2013.

Uses floating-point + directed rounding + rational arithmetic.

Exact MILP Solving: New Exact SCIP Features

Eifler, Gleixner 2021 & 2022:

- thorough revision of hybrid-precision branch and bound
- integrate SoPlex as exact LP solver (Gleixner, Steffy 2019 & 2020)
- addition of rational presolving (Gleixner, Gottwald, Hoen 2023)
- addition of primal heuristics
- output of VIPR certificates (Cheung, Gleixner, Steffy 2017)

Eifler, Gleixner 2023 (preprint available)

- safe, verified generation of Gomory mixed-integer cuts

Published soon:

- domain propagation + conflict analysis (Borst, Eifler, Gleixner)
- precision boosting + iterative refinement in exact LP (Eifler, Gleixner, Thouvenin)

Symmetries in MIPs

Symmetries of a MIP

$$\max\{c^T x \ : \ Ax \leq b, \ x \in \mathbb{Z}^n\}$$

are bijections $f\colon \mathbb{R}^n \to \mathbb{R}^n$ such that $x \in \mathbb{R}^n$ is feasible iff $f(x)$ is feasible and both have the same objective value.

Issue Branch-and-bound trees become unnecessarily large since symmetric subproblems are explored multiple times.

$$\begin{aligned} \max x_1 + x_2 & \\ x_1 \qquad + 2x_3 &\leq 3 \\ x_2 + 2x_3 &\leq 3 \end{aligned}$$

SCIP's Symmetry Handling Tools

Tools in SCIP 8.0

- automatic symmetry detection
- symmetry handling constraints (orbitopes, orbisacks, symresacks, SST cuts)
- propagation algorithms (orbital fixing)

Issue: Constraint-based and propagation-based methods can not be combined.

Latest Symmetry Handling Changes

- completely revised symmetry handling framework that allows to combine constraints and propagation algorithms.
- at the time of merging, the new framework improves on the old framework by 5.9% (25.4% on instances running at least 1000s).
- interface to graph automorphims code sassy to accelerate symmetry detection

Thank you!