Large neighbourhood Benders' search

Stephen J. Maher

Lancaster University Management School,
s.maher3@lancaster.ac.uk

8th March 2018

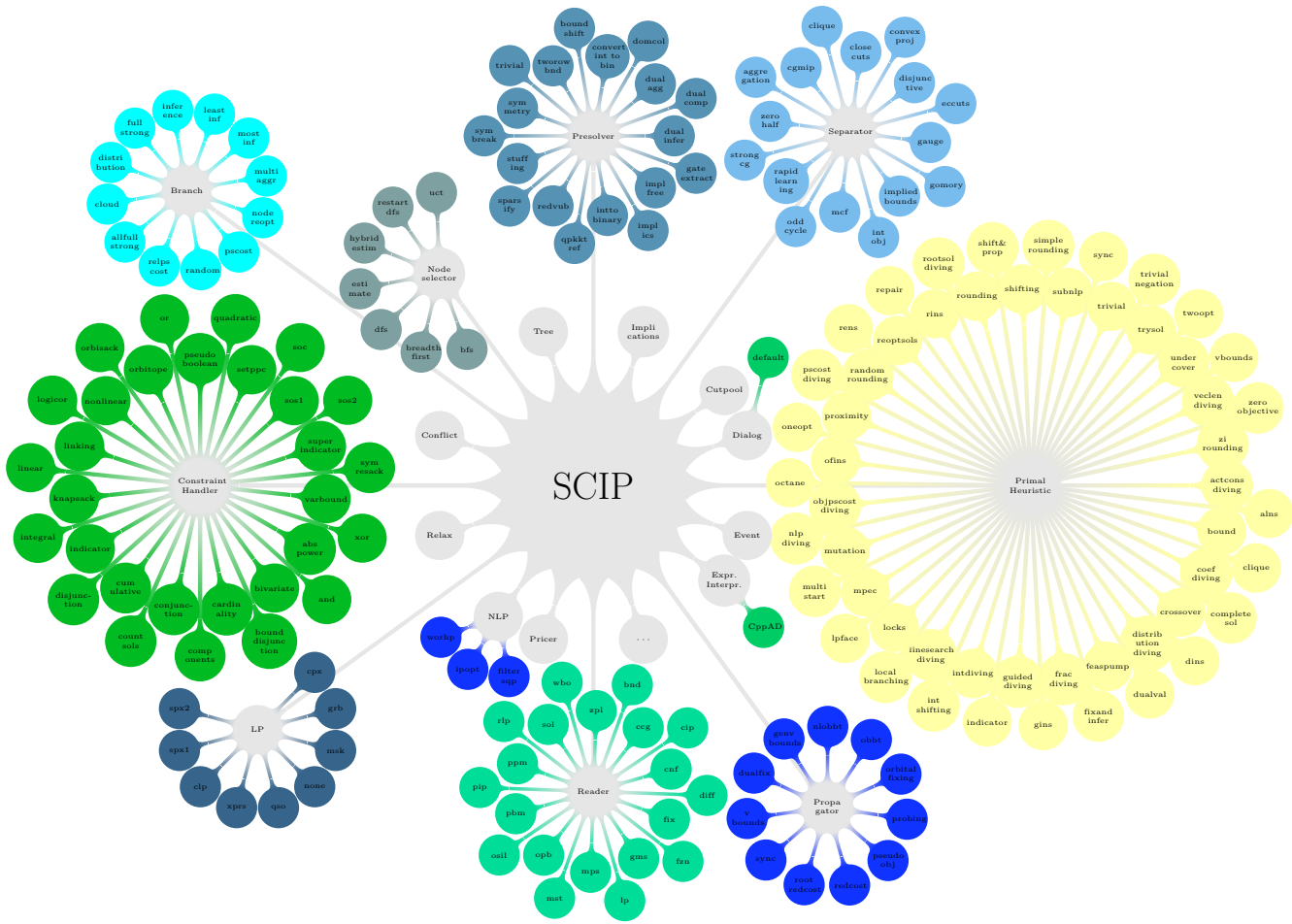
Structured mixed integer programming

Basic idea: Minimise a linear objective function over a set of solutions satisfying a structured set of linear constraints.

$$
\begin{aligned}
\min \quad & c^\top x + d^\top y, \\
\text{subject to} \quad & Ax \geq b, \\
& Bx + Dy \geq g, \\
& x \in \mathbb{Z}_+^{p_1} \times \mathbb{R}_+^{n_1 - p_1}, \\
& y \in \mathbb{Z}_+^{p_2} \times \mathbb{R}_+^{n_2 - p_2}.
\end{aligned}
$$

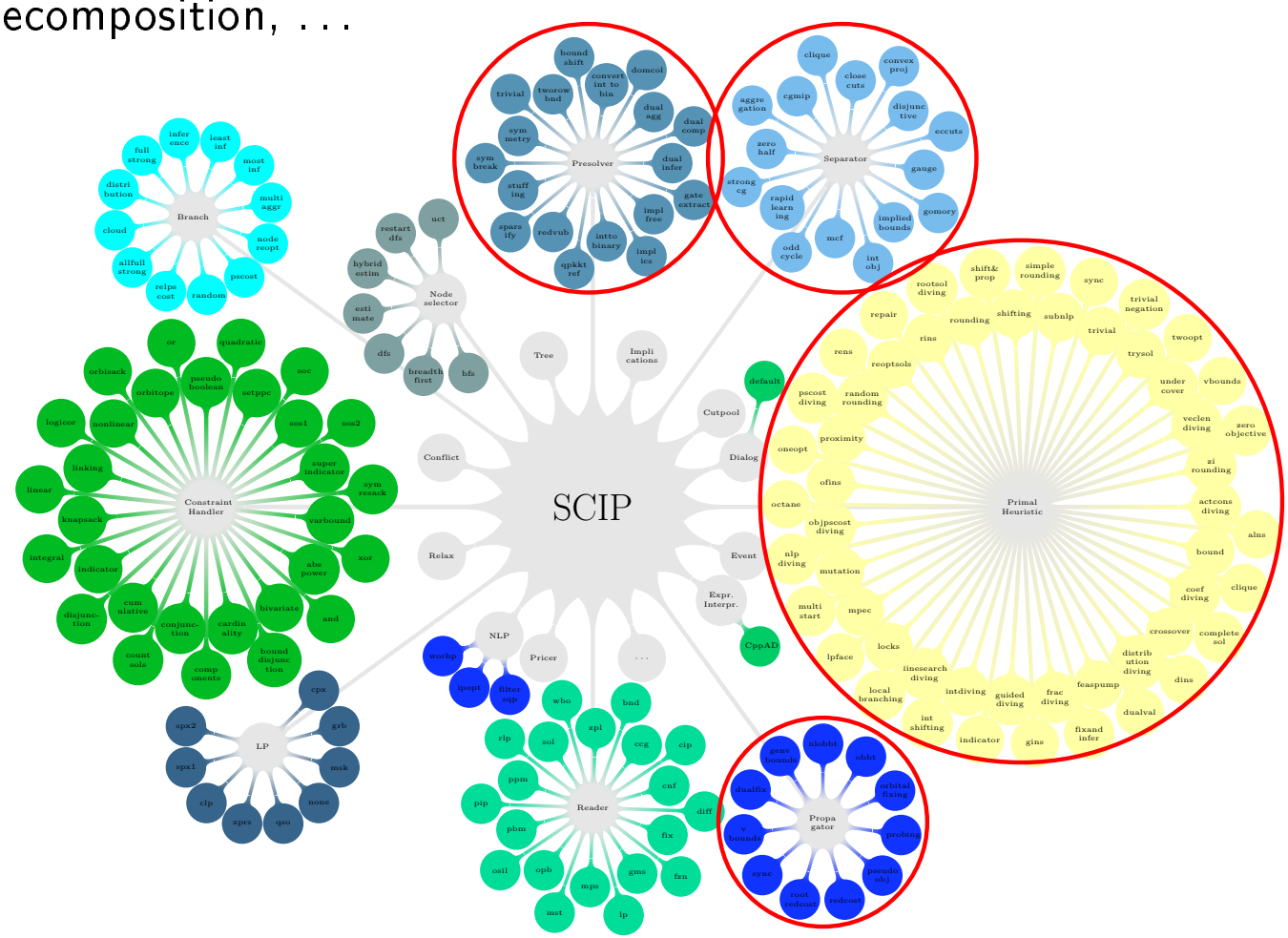
Solving structured mixed integer programs

- State-of-the-art general purpose solvers
  - CPLEX, Gurobi, Xpress, **SCIP**, ...



Solving structured mixed integer programs

- State-of-the-art general purpose solvers
  - CPLEX, Gurobi, Xpress, **SCIP**, ...
- Decomposition techniques
  - Column generation, Lagrangian relaxation/decomposition, Benders' decomposition, ...



Solving large scale optimisation problems

Aim: Embed Benders' decomposition within a state-of-the-art solver to provide effective tools to employ decomposition to solve large-scale optimisation problems.

Solving large scale optimisation problems

Aim: Embed Benders' decomposition within a state-of-the-art solver to provide effective tools to employ decomposition to solve large-scale optimisation problems.

- Develop a general Benders' decomposition framework
- Harness the capabilities of state-of-the-art MIP solvers when using decomposition techniques
  - Integration of Benders' decomposition and large neighbourhood search heuristics.

Benders' decomposition

Original problem

$$
\begin{aligned}
\min \quad & c^\top x + d^\top y, \\
\text{subject to} \quad & Ax \geq b, \\
& Bx + Dy \geq g, \\
& x \in \mathbb{Z}_+^{p_1} \times \mathbb{R}_+^{n_1 - p_1}, \\
& y \in \mathbb{R}_+^{n_2}.
\end{aligned}
$$

Benders' decomposition

$$\begin{aligned} \min \quad & c^\top x + f(x), \\ \text{subject to} \quad & Ax \geq b, \\ & x \in \mathbb{Z}_+^{p_1} \times \mathbb{R}_+^{n_1 - p_1}. \end{aligned}$$

where

$$f(x) = \min_{y \in \mathbb{R}_+^{n_2}} \{ d^\top y \mid Bx + Dy \geq g \}$$

Benders' decomposition

Master problem

$$
\begin{aligned}
\min \quad & c^\top x + \varphi, \\
\text{subject to} \quad & Ax \geq b, \\
& \varphi \geq u_\omega^\top (g - Bx) \quad \forall \omega \in \mathcal{O}, \\
& 0 \geq u_\omega^\top (g - Bx) \quad \forall \omega \in \mathcal{F}, \\
& \varphi \in \mathbb{R}_+, \\
& x \in \mathbb{Z}_+^{p_1} \times \mathbb{R}_+^{n_1 - p_1}.
\end{aligned}
$$

Subproblem

$$
\begin{aligned}
z(\hat{x}) = \min \quad & d^\top y, \\
\text{subject to} \quad & Dy \geq g - B\hat{x}, \\
& y \in \mathbb{R}_+^{n_2 - p_2}.
\end{aligned}
$$

Benders' decomposition

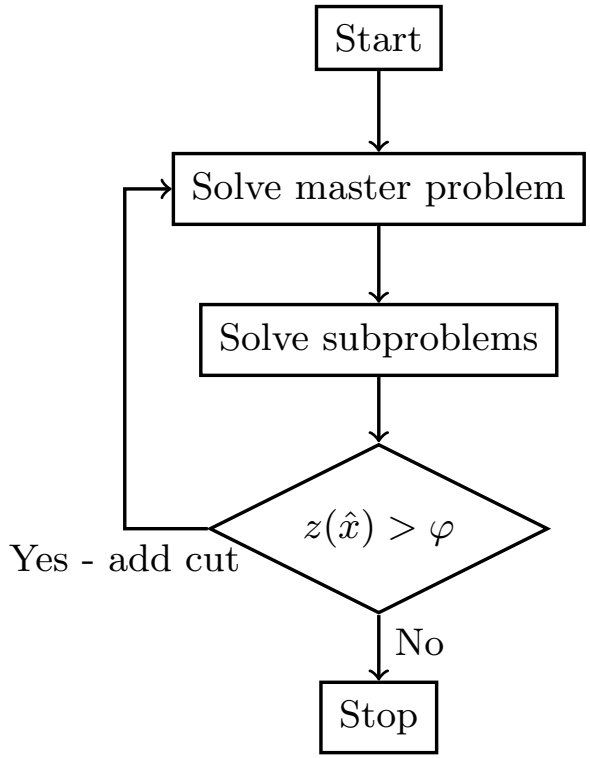
Master problem

$$
\begin{aligned}
\min \quad & c^\top x + \varphi, \\
\text{subject to} \quad & Ax \geq b, \\
& \varphi \geq u_\omega^\top (g - Bx) \quad \forall \omega \in \mathcal{O}, \\
& 0 \geq u_\omega^\top (g - Bx) \quad \forall \omega \in \mathcal{F}, \\
& \{\text{no-good/integer cuts}\}, \\
& \varphi \in \mathbb{R}_+, \\
& x \in \mathbb{Z}_+^{p_1} \times \mathbb{R}_+^{n_1 - p_1}.
\end{aligned}
$$

Subproblem

$$
\begin{aligned}
z(\hat{x}) = \min \quad & d^\top y, \\
\text{subject to} \quad & Dy \geq g - B\hat{x}, \\
& y \in \mathbb{Z}_+^{p_2} \times \mathbb{R}_+^{n_2 - p_2}.
\end{aligned}
$$

Standard Benders' implementation

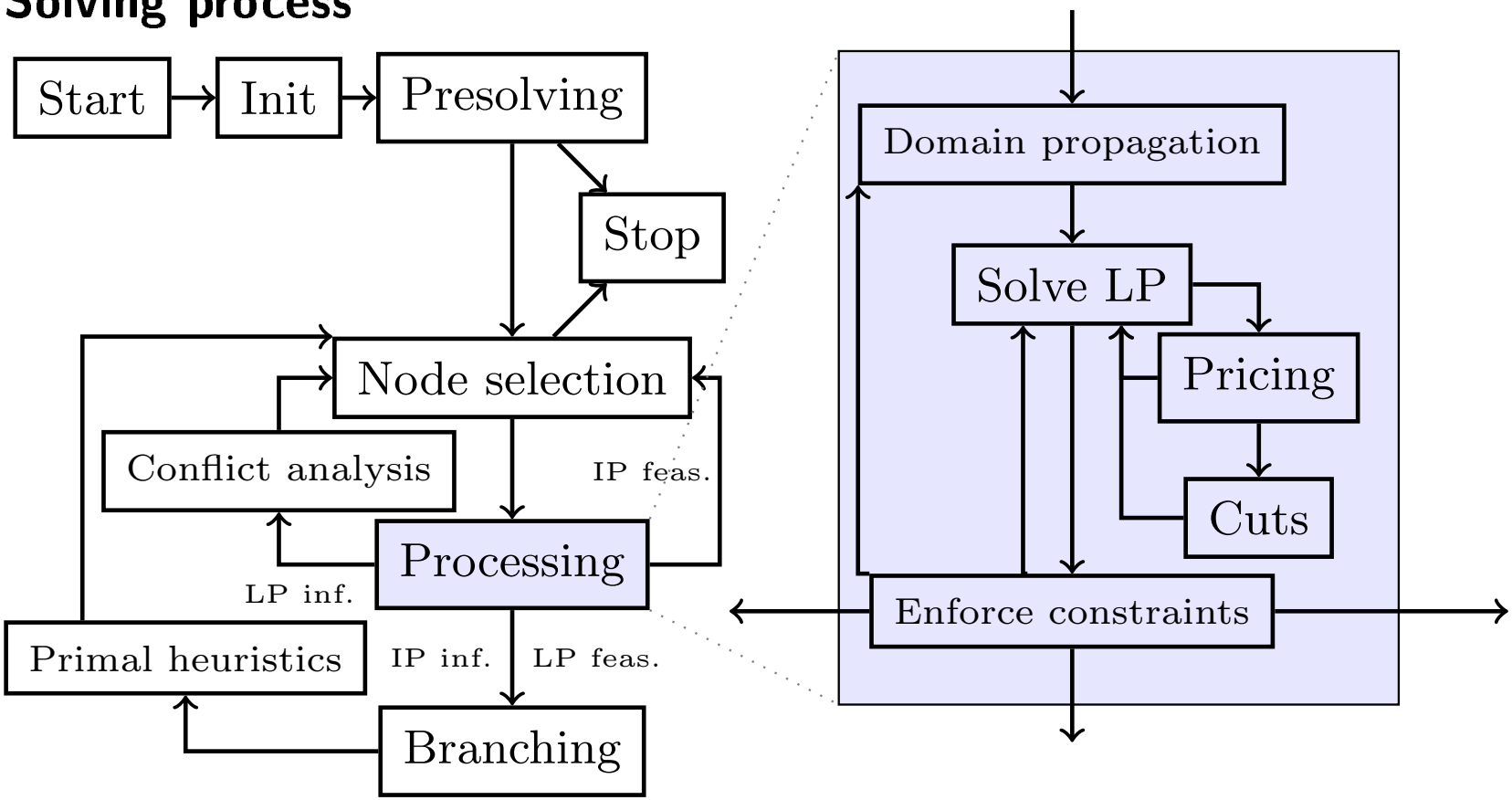


- Easy to understand and simple to implement.
- Not always effective, large overhead in repeatedly solving master problem.

Branch-and-check

- Modern solvers pass through a number of different stages during node processing.
- Some of these stages can be used to generate Benders' cuts.
- By interrupting node processing, Benders' cuts are generated during the tree search.

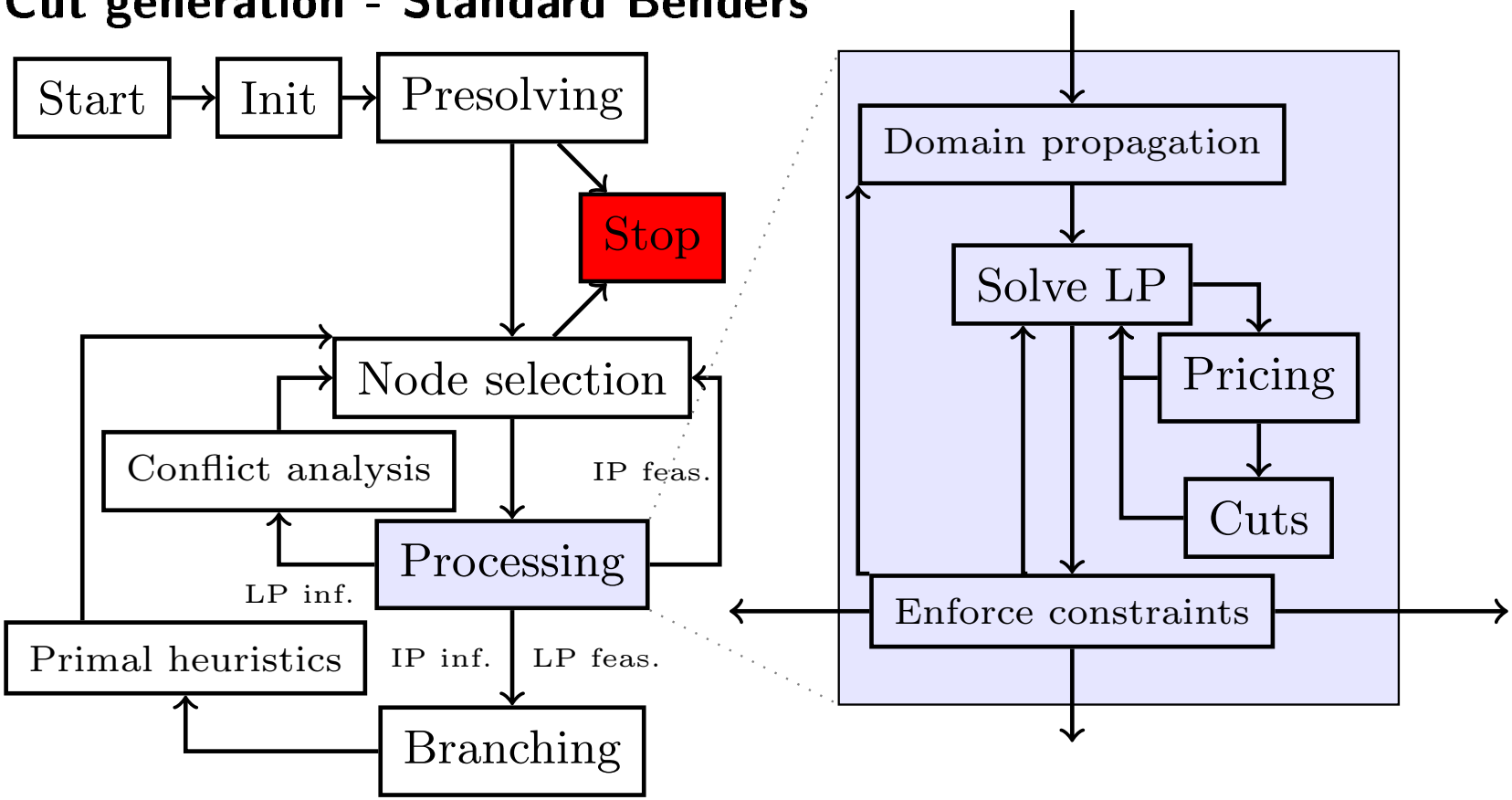
Solving process



Branch-and-check

- Modern solvers pass through a number of different stages during node processing.
- Some of these stages can be used to generate Benders' cuts.
- By interrupting node processing, Benders' cuts are generated during the tree search.

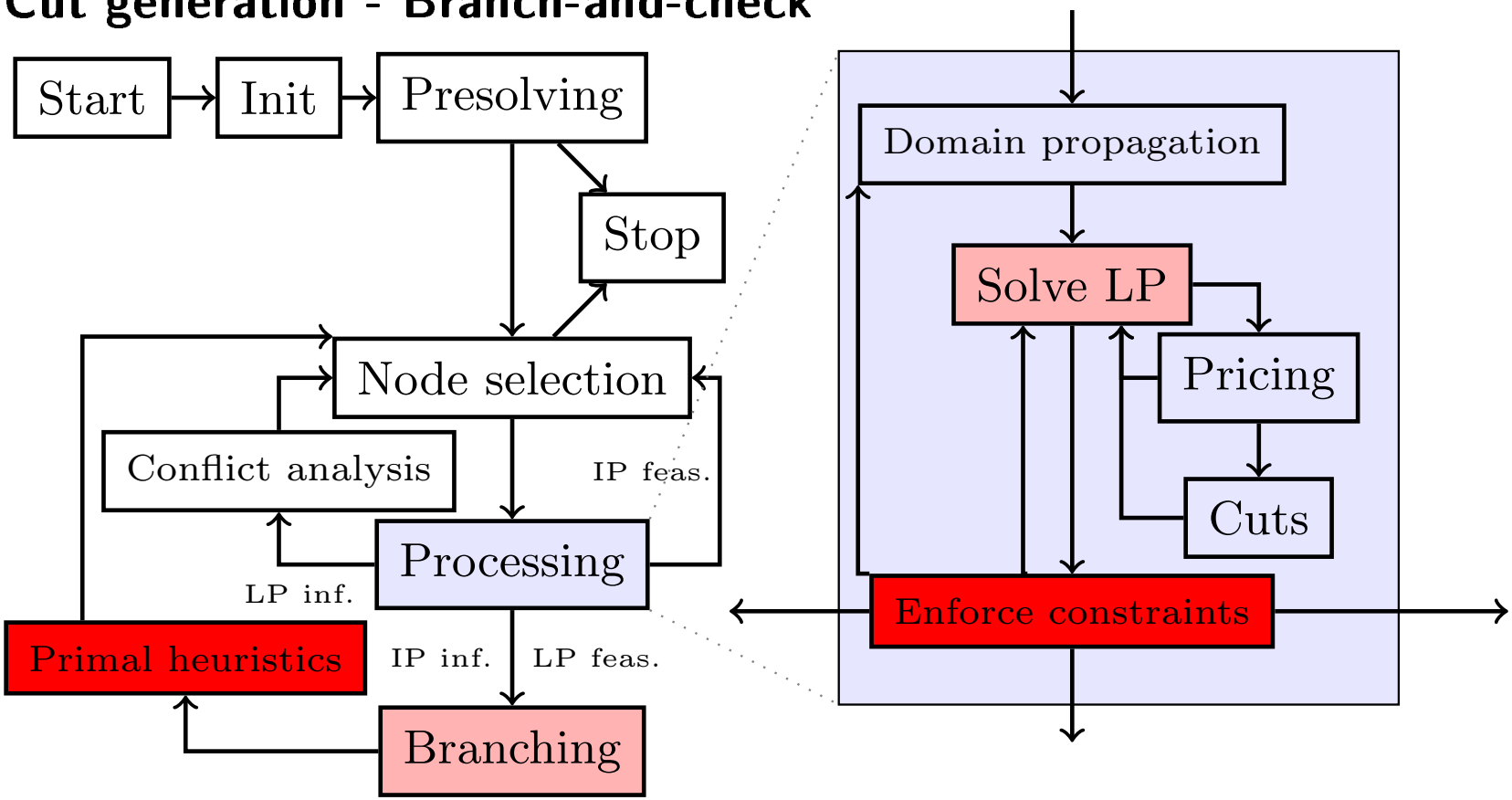
Cut generation - Standard Benders'



Branch-and-check

- Modern solvers pass through a number of different stages during node processing.
- Some of these stages can be used to generate Benders' cuts.
- By interrupting node processing, Benders' cuts are generated during the tree search.

Cut generation - Branch-and-check

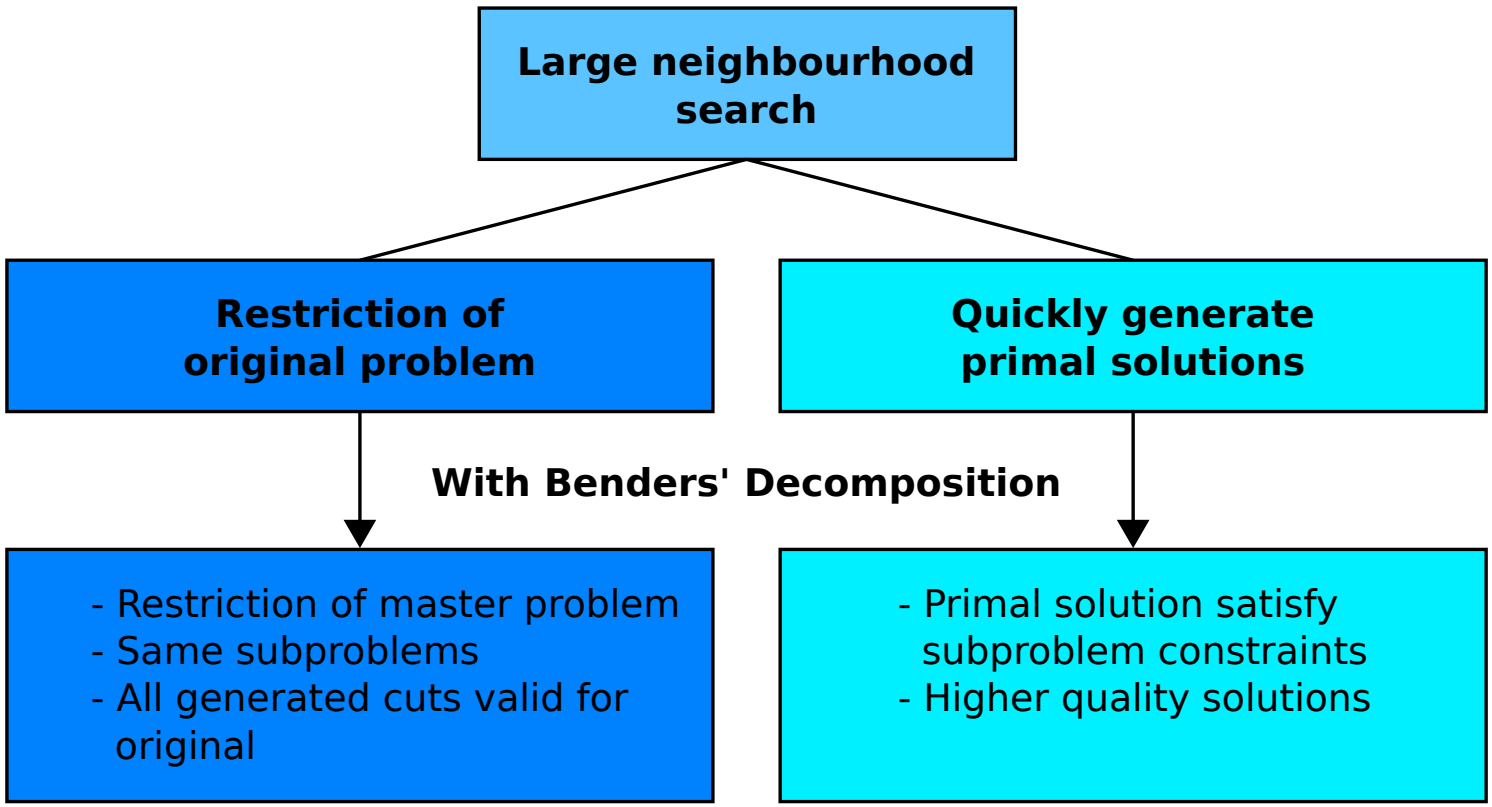


Large neighbourhood search heuristics

- *Concept:* Identify improved primal solutions by solving an auxiliary problem that is a restriction of the original problem.
- The auxiliary problem is typically formed by fixing variables or the addition of constraints.
- The restricted auxiliary problem is expected to be easier to solve than the original problem.
- State-of-the-art solvers employ many variants of large neighbourhood search heuristics
  - Crossover, DINS, Local branching, proximity search, RENS, ...
- Very effective in finding solutions to difficult optimisation problems.

Large Neighbourhood Benders' Search

- *Concept:* Using large neighbourhood search to improve the convergence of the Benders' decomposition algorithm.



- Builds upon successful trust region approaches.

Benders' decomposition everywhere

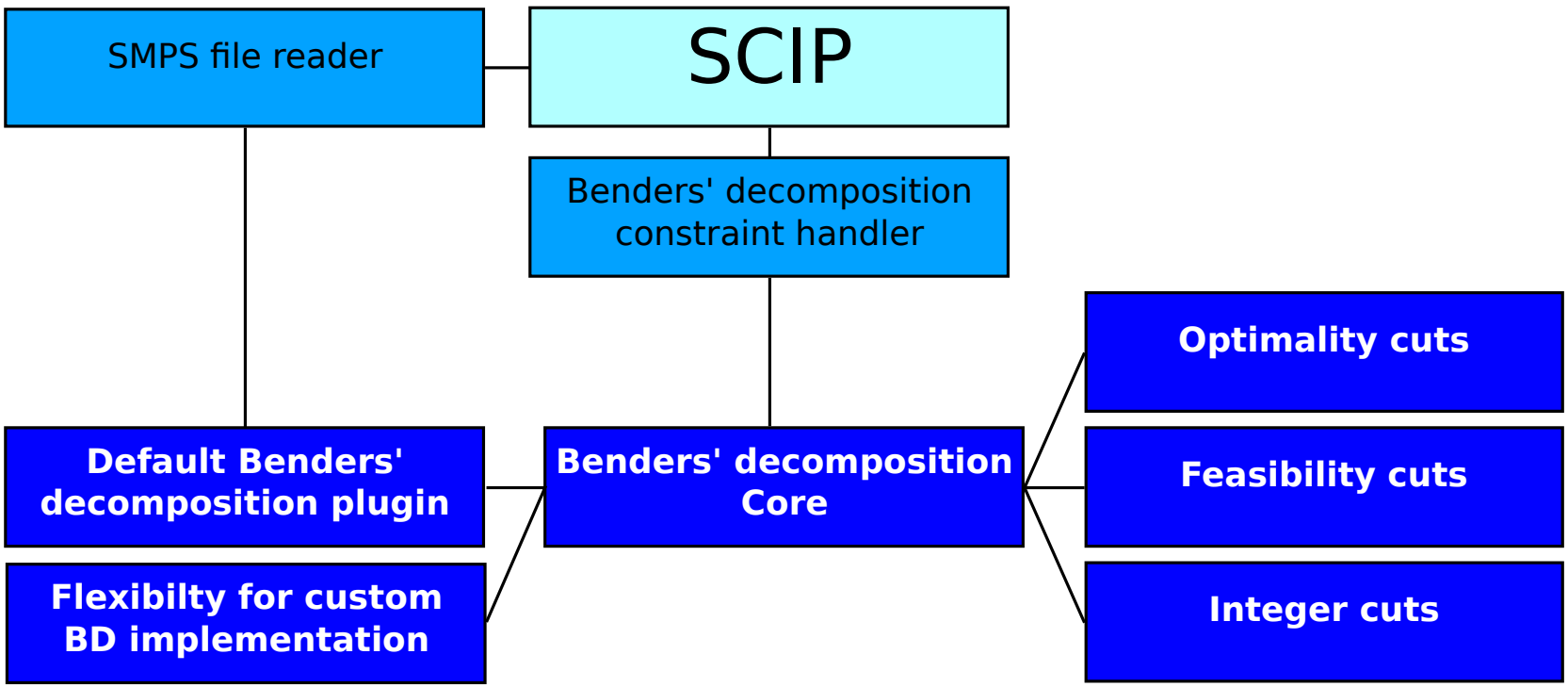
- Local branching (Rei et al. (2009)) and proximity search (Boland et al. (2015)) have demonstrated potential to enhance BD using LNS heuristics.
- Modern MIP solvers are endowed with vast array of LNS heuristics.
- LNS heuristics solve sub-MIP instances—can be solved by Benders' decomposition.

Benders' decomposition everywhere

- Local branching (Rei et al. (2009)) and proximity search (Boland et al. (2015)) have demonstrated potential to enhance BD using LNS heuristics.
- Modern MIP solvers are endowed with vast array of LNS heuristics.
- LNS heuristics solve sub-MIP instances—can be solved by Benders' decomposition.

Aim: Employ Benders' decomposition in all available LNS heuristics.

Generic Benders' Decomposition



Using the Benders' decomposition framework

- *First option:*
  - Provide an instance in SMPS (Stochastic MPS) format to SCIP.
  - The SMPS format consists of three components: core model, time file for stages and stochastic information.
- To use the Benders' decomposition algorithm, the following setting must be used

  `reading/sto/usebenders = TRUE`.

Using the Benders' decomposition framework

- *Second option:*
  - `SCIPcreateBendersDefault(master SCIP, array of subproblem SCIPs, number of subproblems)`
- Master SCIP and Subproblem SCIP instances must be created by the user.
- Variables common between the master problem and subproblems must have the same name.
- All variable mappings between master and subproblems are generated automatically.

Using the Benders' decomposition framework

- *Third option:*
  - Implement a custom Benders' decomposition plugin: `benders_xyz.c` and `benders_xyz.h`
- Required callbacks:
  - Mapping between the master and subproblem variables,
  - Method to create each subproblem.
- Various optional callbacks:
  - Pre subproblem solving, a solving method for the subproblem, post solving, freeing subproblem, and usual SCIP callbacks (init, initsol, exit, exitsol, copy, ...)

Using Benders' decomposition in LNS

Advantages:

- Solutions are guaranteed to be optimal w.r.t the subproblems.
- Generated cuts can be transferred to the original problem.
- Flexible w.r.t SCIP development.
- Simple changes of parameters can provide aggressive use of LNS for Benders' decomposition.

Test problems

Stochastic Capacitated Facility Location Problem

- CAP instances from OR-Library with 25 or 50 facilities.
- Instances with 250 and 500 scenarios.
- 48 instances.

Stochastic Network Interdiction Problem

- The same set of instances as used by Bodour et al. (2017).
- Graph with 738 nodes and 2586 arcs, 320 possible sensor locations.
- 456 scenarios.

Stochastic Multiple Knapsack Problem

- Instances collected from SIPLIB.
- First stage: 240 binary variables, 50 knapsack constraints.
- Second stage: 120 binary variables, 5 knapsack constraints.
- 30 instances, each with 20 scenarios.

Experiments

Benders - Standard implementation of branch-and-check.

- Benders' cuts are generated from the LP, Relaxation, Pseudo and feasible solutions

LNS Check - BD with large neighbourhood Benders' search.

- Employing Benders' decomposition within each LNS heuristic

Transfer cuts - Extension of large neighbourhood Benders' search.

- All cuts generated during the large neighbourhood Benders' search are transferred to the original problem

Primal Integral (Berthold (2013))

Primal Gap

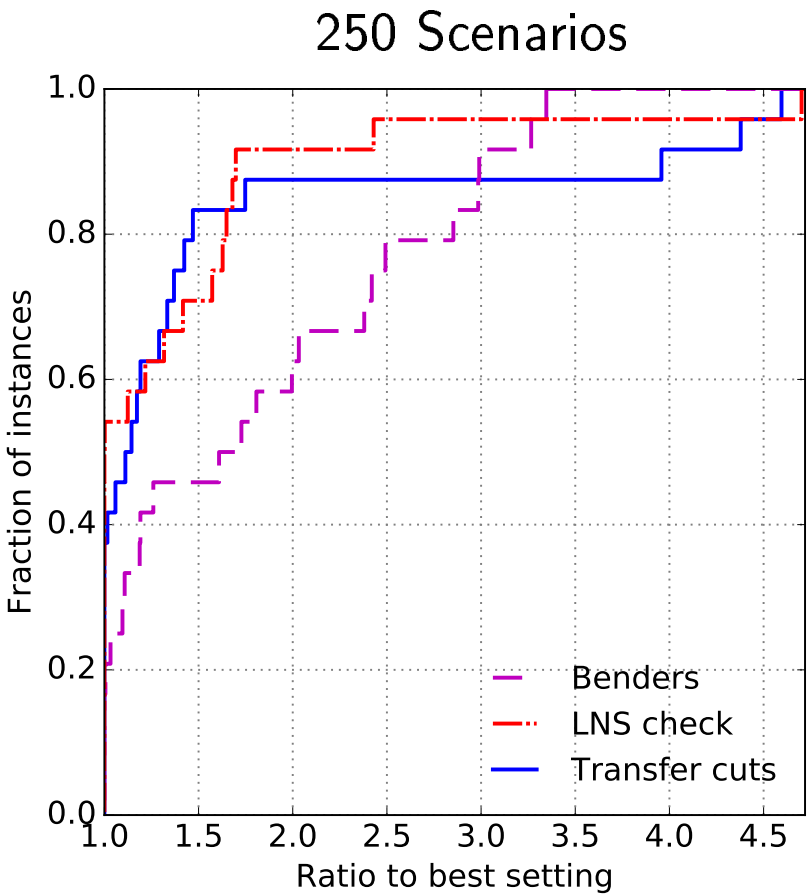
$$\gamma(Z^p, \hat{Z}^p) = \begin{cases} 0, & \text{if } |Z^p(t)| = |\hat{Z}^p| = 0, \\ 1, & \text{if } Z^p(t) \times \hat{Z}^p < 0, \\ \dfrac{|Z^p(t) - \hat{Z}^p|}{\max\{|Z^p(t)|, |\hat{Z}^p|\}}, & \text{otherwise.} \end{cases}$$

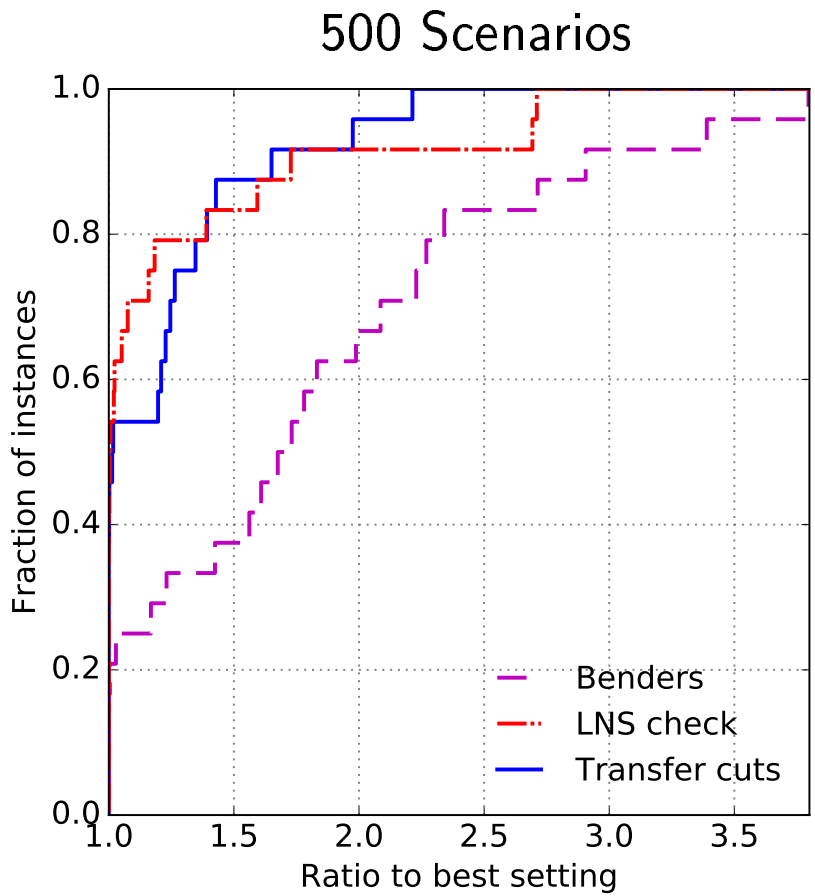
Integral

$$P(T) = \int_{t=0}^{T} \gamma(Z^p, \hat{Z}^p) dt$$

SCFLP

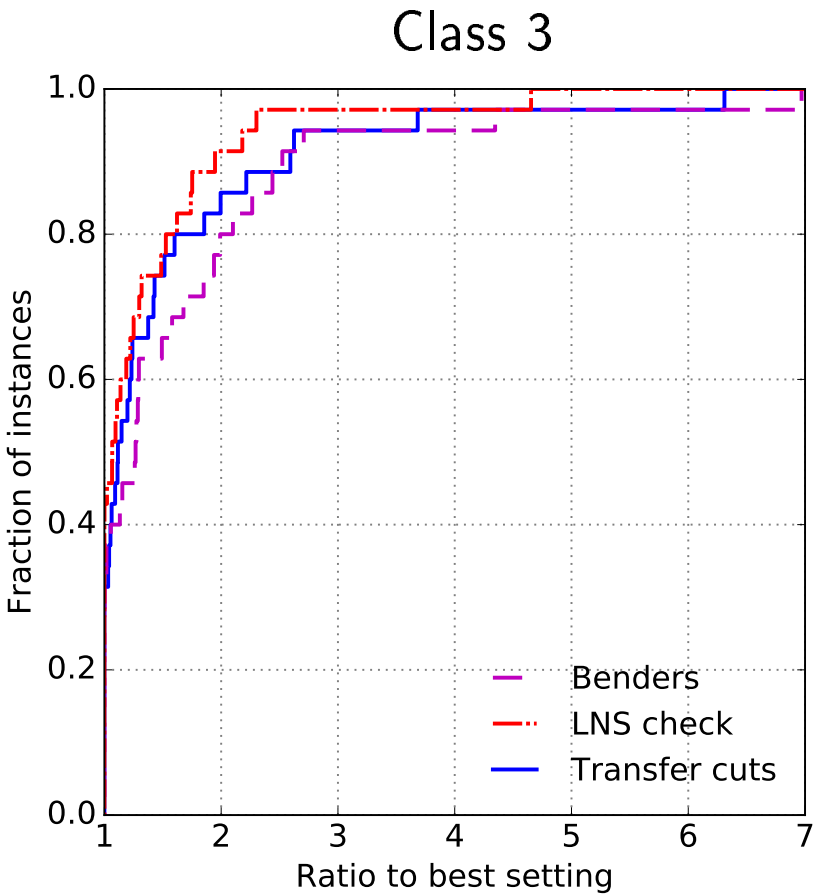
Performance profile of the primal integral

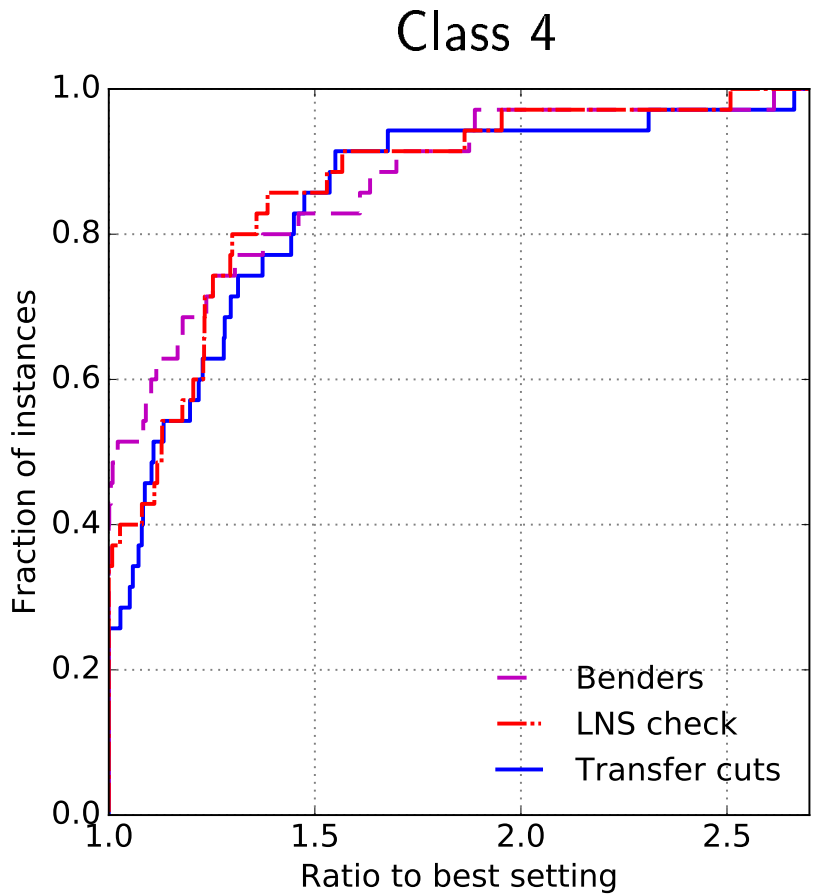




SNIP

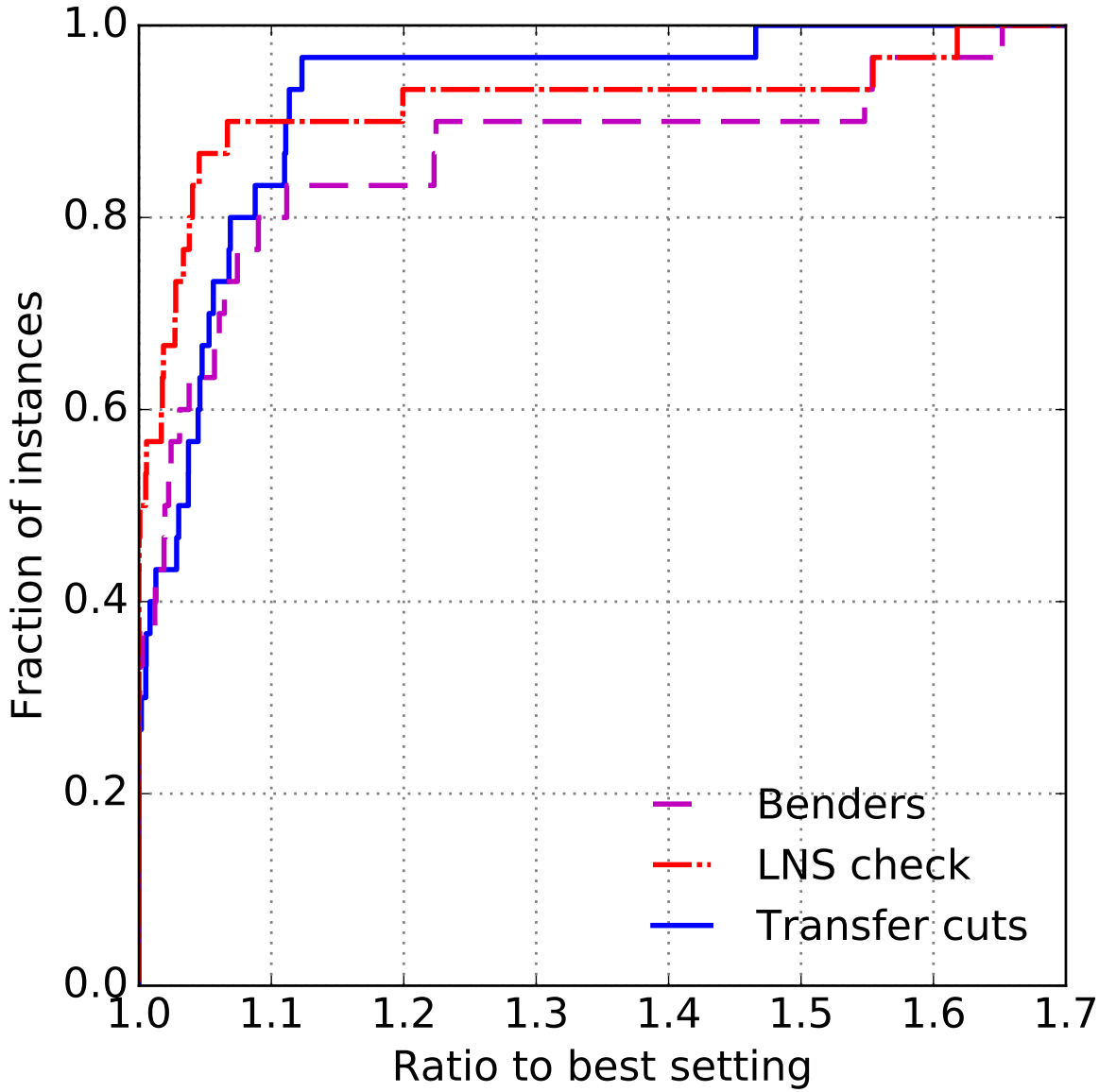
Performance profile of the primal integral





SMKP

Performance profile of the primal integral



Key points

- SCIP has been extended with a generic implementation of Benders' decomposition.
- Benders' decomposition has been implemented as a branch-and-check algorithm.
- Functionality is available to use Benders' decomposition within LNS heuristics.
- Integrating Benders' decomposition and LNS heuristics can significantly enhance the primal bound improvement.