Cutting Planes and Primal Heuristics

CO@Work Berlin

Timo Berthold and Kati Wolter

09/30/2009

Berlin Mathematical School

DFG Research Center MATHEON
Mathematics for key technologies

Cutting Planes and Primal Heuristics

CO@Work Berlin

Timo Berthold and Kati Wolter

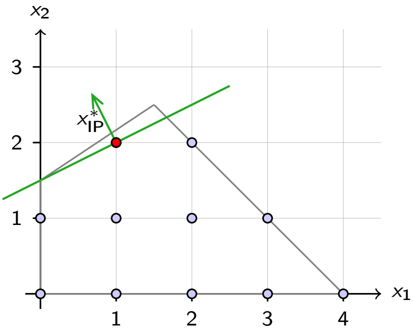
09/30/2009

Berlin Mathematical School

DFG Research Center MATHEON
Mathematics for key technologies

$$\mathcal{F}_{\text{IP}} := \{x \in \mathbb{Z}^n_+ : Ax \le b\}$$

$$\mathcal{F}_{\text{LP}} := \{x \in \mathbb{R}^n_+ : Ax \le b\}$$



$\min\{c^T x : x \in \mathcal{F}_{\text{IP}}\}$

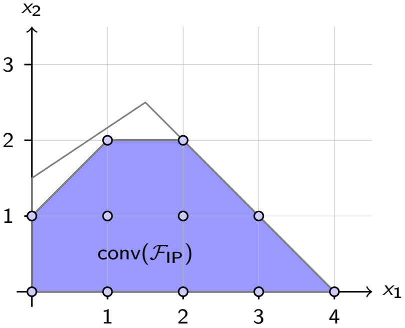
General Cutting Plane Method

Observation

- ▷ conv($\mathcal{F}_{\text{IP}}$) is a polyhedron
- ▷ IP can be formulated as LP

Problems with conv($\mathcal{F}_{\text{IP}}$):

- ▷ linear description not known
- ▷ large nr. of constraints needed



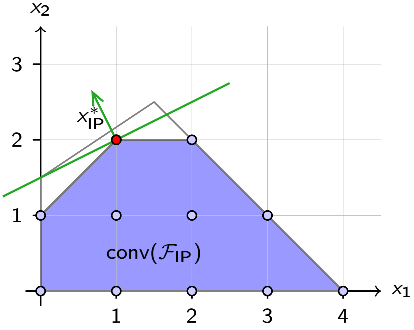
General Cutting Plane Method

Observation

- $\mathrm{conv}(\mathcal{F}_{\mathrm{IP}})$ is a polyhedron
- IP can be formulated as LP

Problems with $\mathrm{conv}(\mathcal{F}_{\mathrm{IP}})$:

- linear description not known
- large nr. of constraints needed



$$\min\{c^T x : x \in \mathrm{conv}(\mathcal{F}_{\mathrm{IP}})\}$$

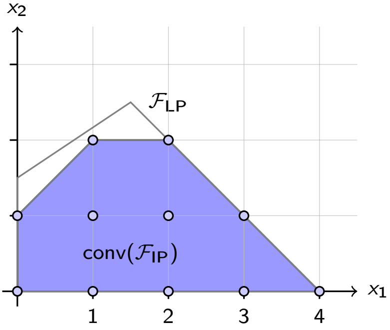
General Cutting Plane Method

Observation

- $\text{conv}(\mathcal{F}_{\text{IP}})$ is a polyhedron
- IP can be formulated as LP

Problems with $\text{conv}(\mathcal{F}_{\text{IP}})$:

- linear description not known
- large nr. of constraints needed



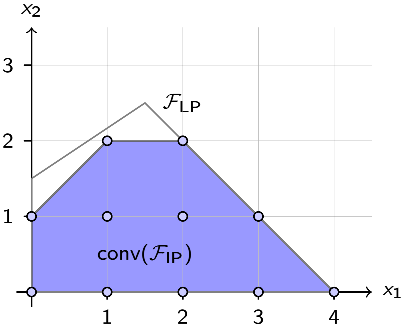
$\min\{c^T x : x \in \text{conv}(\mathcal{F}_{\text{IP}})\}$

$$\begin{array}{ccccc} \mathcal{F}_{\text{LP}} & \supseteq & \mathcal{F} & \supseteq & \text{conv}(\mathcal{F}_{\text{IP}}) \\ \min\{c^T x : x \in \mathcal{F}_{\text{LP}}\} & \leq & \min\{c^T x : x \in \mathcal{F}\} & = & \min\{c^T x : x \in \text{conv}(\mathcal{F}_{\text{IP}})\} \end{array}$$

General Cutting Plane Method

Algorithm

1. $\mathcal{F} \leftarrow \mathcal{F}_{\text{LP}}$
2. Solve
$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & x \in \mathcal{F} \end{array}$$
3. If $x^* \in \mathcal{F}_{\text{IP}}$: Stop
4. Add inequality to $\mathcal{F}$ that is ...
   - valid for $\text{conv}(\mathcal{F}_{\text{IP}})$ but
   - violated by x^*.
5. Goto 2.



$\min\{c^T x : x \in \text{conv}(\mathcal{F}_{\text{IP}})\}$

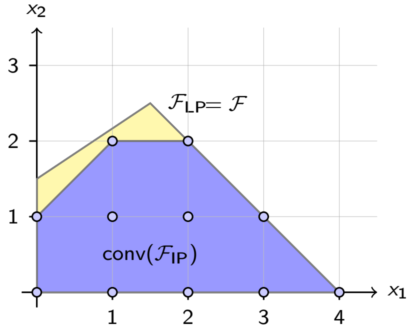
General Cutting Plane Method

Algorithm

1. $\mathcal{F} \leftarrow \mathcal{F}_{\text{LP}}$
2. Solve

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & x \in \mathcal{F} \end{array}$$

3. If $x^* \in \mathcal{F}_{\text{IP}}$: Stop
4. Add inequality to $\mathcal{F}$ that is ...
   - valid for $\text{conv}(\mathcal{F}_{\text{IP}})$ but
   - violated by x^*.
5. Goto 2.



$\min\{c^T x : x \in \text{conv}(\mathcal{F}_{\text{IP}})\}$

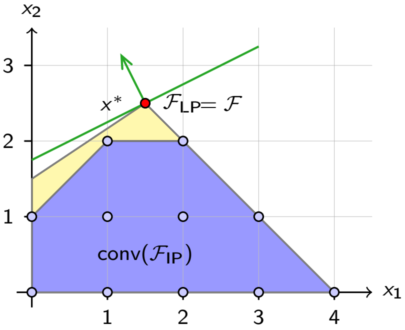
General Cutting Plane Method

Algorithm

1. $\mathcal{F} \leftarrow \mathcal{F}_{\text{LP}}$
2. Solve

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & x \in \mathcal{F} \end{array}$$

3. If $x^* \in \mathcal{F}_{\text{IP}}$: Stop
4. Add inequality to $\mathcal{F}$ that is ...
   - valid for $\text{conv}(\mathcal{F}_{\text{IP}})$ but
   - violated by x^*.
5. Goto 2.



$\min\{c^T x : x \in \text{conv}(\mathcal{F}_{\text{IP}})\}$

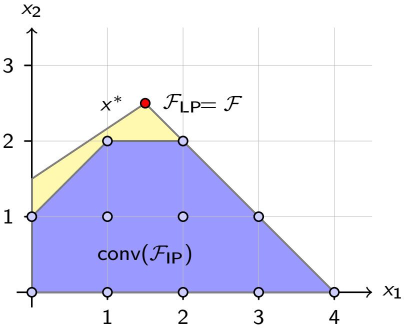
General Cutting Plane Method

Algorithm

1. $\mathcal{F} \leftarrow \mathcal{F}_{\text{LP}}$
2. Solve

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & x \in \mathcal{F} \end{array}$$

3. If $x^* \in \mathcal{F}_{\text{IP}}$: Stop
4. Add inequality to $\mathcal{F}$ that is ...
   - valid for $\text{conv}(\mathcal{F}_{\text{IP}})$ but
   - violated by x^*.
5. Goto 2.



$\min\{c^T x : x \in \text{conv}(\mathcal{F}_{\text{IP}})\}$

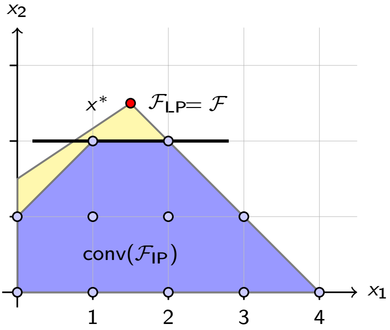
General Cutting Plane Method

Algorithm

1. $\mathcal{F} \leftarrow \mathcal{F}_{\text{LP}}$
2. Solve

$$\begin{array}{rr} \min & c^T x \\ \text{s.t.} & x \in \mathcal{F} \end{array}$$

3. If $x^* \in \mathcal{F}_{\text{IP}}$: Stop
4. Add inequality to $\mathcal{F}$ that is ...
   - valid for $\text{conv}(\mathcal{F}_{\text{IP}})$ but
   - violated by x^*.
5. Goto 2.

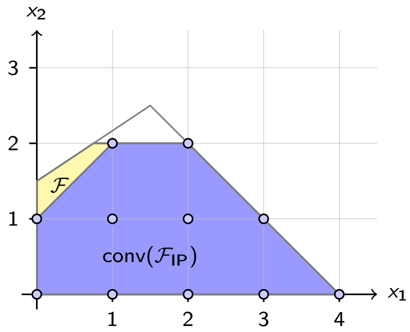


$\min\{c^T x : x \in \text{conv}(\mathcal{F}_{\text{IP}})\}$

General Cutting Plane Method

Algorithm

1. $\mathcal{F} \leftarrow \mathcal{F}_{\text{LP}}$
2. Solve
$$\begin{array}{rr} \min & c^T x \\ \text{s.t.} & x \in \mathcal{F} \end{array}$$
3. If $x^* \in \mathcal{F}_{\text{IP}}$: Stop
4. Add inequality to $\mathcal{F}$ that is ...
   - valid for $\text{conv}(\mathcal{F}_{\text{IP}})$ but
   - violated by x^*.
5. Goto 2.

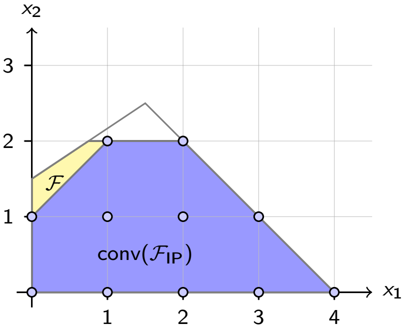


$\min\{c^T x : x \in \text{conv}(\mathcal{F}_{\text{IP}})\}$

General Cutting Plane Method

Algorithm

1. $\mathcal{F} \leftarrow \mathcal{F}_{\text{LP}}$
2. Solve
$$\begin{array}{rr} \min & c^T x \\ \text{s.t.} & x \in \mathcal{F} \end{array}$$
3. If $x^* \in \mathcal{F}_{\text{IP}}$: Stop
4. Add inequality to $\mathcal{F}$ that is ...
   - valid for $\text{conv}(\mathcal{F}_{\text{IP}})$ but
   - violated by x^*.
5. Goto 2.



$\min\{c^T x : x \in \text{conv}(\mathcal{F}_{\text{IP}})\}$

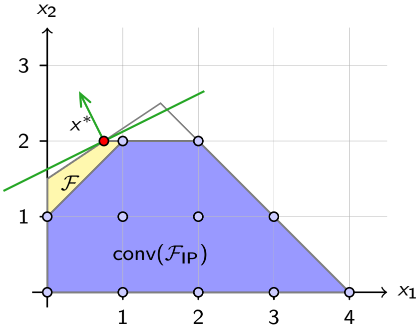
General Cutting Plane Method

Algorithm

1. $\mathcal{F} \leftarrow \mathcal{F}_{\text{LP}}$
2. Solve

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & x \in \mathcal{F} \end{array}$$

3. If $x^* \in \mathcal{F}_{\text{IP}}$: Stop
4. Add inequality to $\mathcal{F}$ that is ...
   - valid for $\text{conv}(\mathcal{F}_{\text{IP}})$ but
   - violated by x^*.
5. Goto 2.

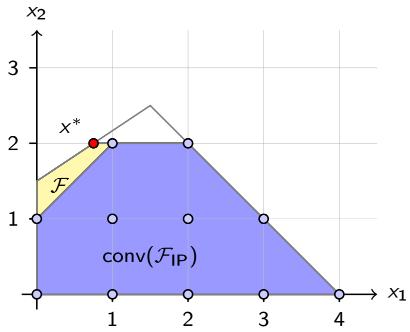


$\min\{c^T x : x \in \text{conv}(\mathcal{F}_{\text{IP}})\}$

General Cutting Plane Method

Algorithm

1. $\mathcal{F} \leftarrow \mathcal{F}_{\text{LP}}$
2. Solve
$$\begin{array}{rr} \min & c^T x \\ \text{s.t.} & x \in \mathcal{F} \end{array}$$
3. If $x^* \in \mathcal{F}_{\text{IP}}$: Stop
4. Add inequality to $\mathcal{F}$ that is ...
   - valid for $\text{conv}(\mathcal{F}_{\text{IP}})$ but
   - violated by x^*.
5. Goto 2.

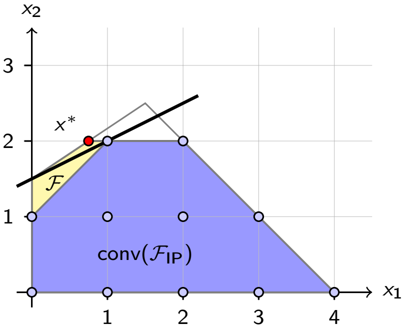


$\min\{c^T x : x \in \text{conv}(\mathcal{F}_{\text{IP}})\}$

General Cutting Plane Method

Algorithm

1. $\mathcal{F} \leftarrow \mathcal{F}_{\mathrm{LP}}$
2. Solve
$$\begin{array}{rr} \min & c^T x \\ \text{s.t.} & x \in \mathcal{F} \end{array}$$
3. If $x^* \in \mathcal{F}_{\mathrm{IP}}$: Stop
4. Add inequality to $\mathcal{F}$ that is ...
   - valid for $\mathrm{conv}(\mathcal{F}_{\mathrm{IP}})$ but
   - violated by x^*.
5. Goto 2.

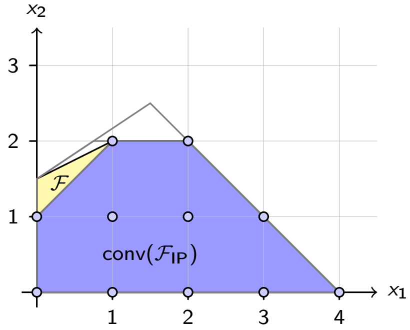


$\min\{c^T x : x \in \mathrm{conv}(\mathcal{F}_{\mathrm{IP}})\}$

General Cutting Plane Method

Algorithm

1. $\mathcal{F} \leftarrow \mathcal{F}_{\text{LP}}$
2. Solve
$$\begin{array}{rr} \min & c^T x \\ \text{s.t.} & x \in \mathcal{F} \end{array}$$
3. If $x^* \in \mathcal{F}_{\text{IP}}$: Stop
4. Add inequality to $\mathcal{F}$ that is ...
   - valid for $\text{conv}(\mathcal{F}_{\text{IP}})$ but
   - violated by x^*.
5. Goto 2.

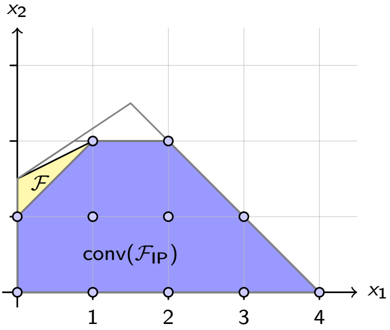


$\min\{c^T x : x \in \text{conv}(\mathcal{F}_{\text{IP}})\}$

General Cutting Plane Method

Algorithm

1. $\mathcal{F} \leftarrow \mathcal{F}_{\text{LP}}$
2. Solve
$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & x \in \mathcal{F} \end{array}$$
3. If $x^* \in \mathcal{F}_{\text{IP}}$: Stop
4. Add inequality to $\mathcal{F}$ that is ...
   - valid for $\text{conv}(\mathcal{F}_{\text{IP}})$ but
   - violated by x^*.
5. Goto 2.



$\min\{c^T x : x \in \text{conv}(\mathcal{F}_{\text{IP}})\}$

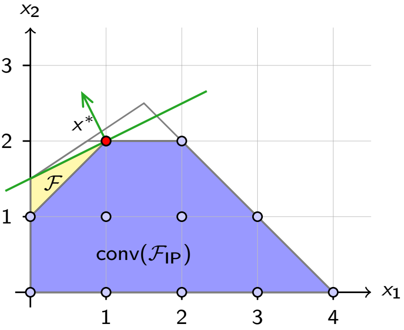
General Cutting Plane Method

Algorithm

1. $\mathcal{F} \leftarrow \mathcal{F}_{\text{LP}}$
2. Solve

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & x \in \mathcal{F} \end{array}$$

3. If $x^* \in \mathcal{F}_{\text{IP}}$: Stop
4. Add inequality to $\mathcal{F}$ that is ...
   - valid for $\text{conv}(\mathcal{F}_{\text{IP}})$ but
   - violated by x^*.
5. Goto 2.



$\min\{c^T x : x \in \text{conv}(\mathcal{F}_{\text{IP}})\}$

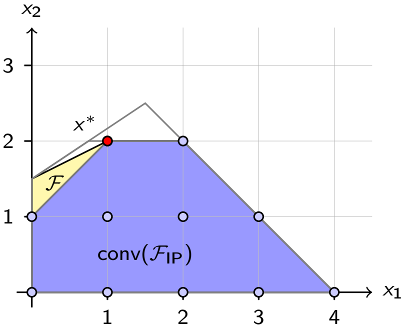
General Cutting Plane Method

Algorithm

1. $\mathcal{F} \leftarrow \mathcal{F}_{\text{LP}}$
2. Solve

$$\begin{array}{rr} \min & c^T x \\ \text{s.t.} & x \in \mathcal{F} \end{array}$$

3. If $x^* \in \mathcal{F}_{\text{IP}}$: Stop
4. Add inequality to $\mathcal{F}$ that is ...
   - valid for $\text{conv}(\mathcal{F}_{\text{IP}})$ but
   - violated by x^*.
5. Goto 2.



$\min\{c^T x : x \in \text{conv}(\mathcal{F}_{\text{IP}})\}$

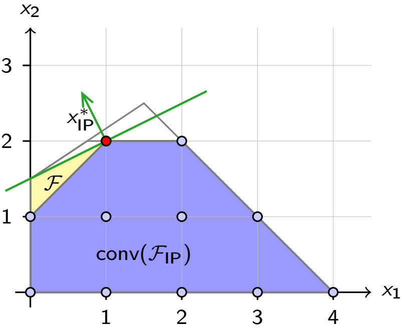
General Cutting Plane Method

Algorithm

1. $\mathcal{F} \leftarrow \mathcal{F}_{\text{LP}}$
2. Solve

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & x \in \mathcal{F} \end{array}$$

3. If $x^* \in \mathcal{F}_{\text{IP}}$: Stop
4. Add inequality to $\mathcal{F}$ that is ...
   - valid for $\text{conv}(\mathcal{F}_{\text{IP}})$ but
   - violated by x^*.
5. Goto 2.

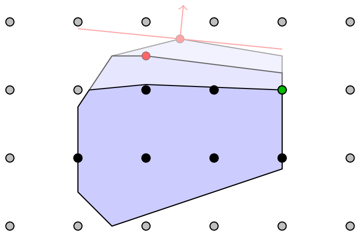


$\min\{c^T x : x \in \text{conv}(\mathcal{F}_{\text{IP}})\}$

Cutting Plane Separation in SCIP

Techniques

- General cuts:
  - complemented MIR cuts
  - Gomory mixed integer cuts
  - strong Chvátal-Gomory cuts
  - $\{0, \frac{1}{2}\}$-cuts
  - implied bound cuts
- Problem specific cuts:
  - 0-1 knapsack problem
  - stable set problem
  - 0-1 single node flow problem
  - multi-commodity-flow problem



Results

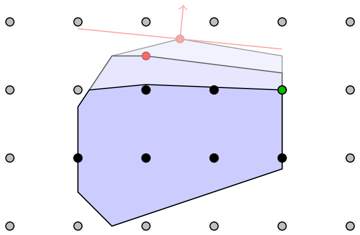
- Very important component
- In particular, c-MIR cuts
- Coordination important

Cutting Plane Separation in SCIP

Techniques

- **General cuts:**
  - complemented MIR cuts
  - Gomory mixed integer cuts
  - strong Chvátal-Gomory cuts
  - $\{0, \frac{1}{2}\}$-cuts
  - implied bound cuts

- **Problem specific cuts:**
  - 0-1 knapsack problem
  - stable set problem
  - 0-1 single node flow problem
  - multi-commodity-flow problem



Results

- Very important component
- In particular, c-MIR cuts
- Coordination important

Cuts for the 0-1 Knapsack Problem

Feasible region: $(b \in \mathbb{Z}_+,\ a_j \in \mathbb{Z}_+ \ \ \forall j \in N)$

$$X^{BK} := \{\ x \in \{0,1\}^n\ :\ \sum_{j \in N} a_j x_j \leq b\ \}$$

Minimal Cover: $C \subseteq N$

- $\sum_{j \in C} a_j > b$
- $\sum_{j \in C \setminus \{i\}} a_j \leq b \quad \forall\, i \in C$

Minimal Cover Inequality

$$\sum_{j \in C} x_j \leq |C| - 1$$

$5x_1 + 6x_2 + 2x_3 + 2x_4 \leq 8$

Minimal cover:
$C = \{2, 3, 4\}$

Minimal cover inequality:
$x_2 + x_3 + x_4 \leq 2$

Cuts for the 0-1 Knapsack Problem

Theorem:

If $C \subseteq N$ is a minimal cover for X^{BK}, then the minimal cover inequality

$$\sum_{j \in C} x_j \leq |C| - 1$$

defines a facet of

$$\text{conv}(\ X^{BK} \cap \ \{x \in \{0,1\}^n \ : \ x_j = 0 \ \text{for all} \ j \in N \setminus C\}\ ).$$

Use sequential up-lifting to strengthen minimal cover inequalities.

Sequential Up-lifting

- $X^{BK} = \{x \in \{0,1\}^4 : 5x_1 + 6x_2 + 2x_3 + 2x_4 \leq 8\}$
- $C = \{2, 3, 4\}$

$$\begin{array}{lll} & \sum_{j \in C} x_j \leq 2 & \text{valid for } X^{BK} \cap \{x \in \{0,1\}^4 : x_1 = 0\} \\ (I) & \sum_{j \in C} x_j + \alpha_1 x_1 \leq 2 & \text{valid for } X^{BK} \end{array}$$

Sequential Up-lifting

- $X^{BK} = \{x \in \{0,1\}^4 : 5x_1 + 6x_2 + 2x_3 + 2x_4 \leq 8\}$
- $C = \{2,3,4\}$

$$\begin{array}{llll} & \sum_{j \in C} x_j & \leq 2 & \text{valid for } X^{BK} \cap \{x \in \{0,1\}^4 : x_1 = 0\} \\ (I) & \sum_{j \in C} x_j + \alpha_1 x_1 & \leq 2 & \text{valid for } X^{BK} \end{array}$$

Step 1: Inequality (I) is valid for $X^{BK} \cap \{x \in \{0,1\}^4 : x_1 = 0\}$

Sequential Up-lifting

- $X^{BK} = \{x \in \{0,1\}^4 : 5x_1 + 6x_2 + 2x_3 + 2x_4 \leq 8\}$
- $C = \{2,3,4\}$

$$\begin{array}{rlll} & \sum_{j \in C} x_j & \leq 2 & \text{valid for } X^{BK} \cap \{x \in \{0,1\}^4 : x_1 = 0\} \\ (I) & \sum_{j \in C} x_j + \alpha_1 x_1 & \leq 2 & \text{valid for } X^{BK} \end{array}$$

Step 1: Inequality (I) is valid for $X^{BK} \cap \{x \in \{0,1\}^4 : x_1 = 0\}$

$$\Leftrightarrow \sum_{j \in C} x_j + \alpha_1 \cdot 0 \leq 2 \text{ is valid for } X^{BK} \cap \{x \in \{0,1\}^4 : x_1 = 0\}$$

Sequential Up-lifting

- $X^{BK} = \{x \in \{0,1\}^4 : 5x_1 + 6x_2 + 2x_3 + 2x_4 \leq 8\}$
- $C = \{2,3,4\}$

$$\begin{array}{rlll} & \sum_{j \in C} x_j & \leq 2 & \text{valid for } X^{BK} \cap \{x \in \{0,1\}^4 : x_1 = 0\} \\ (I) & \sum_{j \in C} x_j + \alpha_1 x_1 & \leq 2 & \text{valid for } X^{BK} \end{array}$$

Step 1: Inequality (I) is valid for $X^{BK} \cap \{x \in \{0,1\}^4 : x_1 = 0\}$

$$\Leftrightarrow \sum_{j \in C} x_j + \alpha_1 \cdot 0 \leq 2 \text{ is valid for } X^{BK} \cap \{x \in \{0,1\}^4 : x_1 = 0\}$$

$$\Leftrightarrow \alpha_1 \in [-\infty, \infty]$$

Sequential Up-lifting

- $X^{BK} = \{x \in \{0,1\}^4 : 5x_1 + 6x_2 + 2x_3 + 2x_4 \leq 8\}$
- $C = \{2,3,4\}$

$$\begin{array}{llll} & \sum_{j \in C} x_j & \leq 2 & \text{valid for } X^{BK} \cap \{x \in \{0,1\}^4 : x_1 = 0\} \\ (I) & \sum_{j \in C} x_j + \alpha_1 x_1 & \leq 2 & \text{valid for } X^{BK} \end{array}$$

Step 2: Inequality (I) is valid for $X^{BK} \cap \{x \in \{0,1\}^4 : x_1 = 1\}$

Sequential Up-lifting

- $X^{BK} = \{x \in \{0,1\}^4 : 5x_1 + 6x_2 + 2x_3 + 2x_4 \leq 8\}$
- $C = \{2,3,4\}$

$$\begin{array}{llll} & \sum_{j \in C} x_j & \leq 2 & \text{valid for } X^{BK} \cap \{x \in \{0,1\}^4 : x_1 = 0\} \\ (I) & \sum_{j \in C} x_j + \alpha_1 x_1 & \leq 2 & \text{valid for } X^{BK} \end{array}$$

Step 2: Inequality (I) is valid for $X^{BK} \cap \{x \in \{0,1\}^4 : x_1 = 1\}$

$$\Leftrightarrow \sum_{j \in C} x_j + \alpha_1 \cdot 1 \leq 2 \text{ is valid for } \{x \in \{0,1\}^4 : 6x_2 + 2x_3 + 2x_4 \leq 8 - 5\}$$

Sequential Up-lifting

- $X^{BK} = \{x \in \{0,1\}^4 : 5x_1 + 6x_2 + 2x_3 + 2x_4 \leq 8\}$
- $C = \{2,3,4\}$

$$\begin{array}{llll} & \sum_{j \in C} x_j & \leq 2 & \text{valid for } X^{BK} \cap \{x \in \{0,1\}^4 : x_1 = 0\} \\ (I) & \sum_{j \in C} x_j + \alpha_1 x_1 & \leq 2 & \text{valid for } X^{BK} \end{array}$$

Step 2: Inequality (I) is valid for $X^{BK} \cap \{x \in \{0,1\}^4 : x_1 = 1\}$

$$\Leftrightarrow \sum_{j \in C} x_j + \alpha_1 \cdot 1 \leq 2 \text{ is valid for } \{x \in \{0,1\}^4 : 6x_2 + 2x_3 + 2x_4 \leq 8 - 5\}$$

$$\Leftrightarrow \max\{ \sum_{j \in C} x_j \ : \ 6x_2 + 2x_3 + 2x_4 \leq 3, \ x \in \{0,1\}^4 \ \} + \alpha_1 \cdot 1 \leq 2$$

Sequential Up-lifting

- $X^{BK} = \{x \in \{0,1\}^4 : 5x_1 + 6x_2 + 2x_3 + 2x_4 \leq 8\}$
- $C = \{2,3,4\}$

$$\begin{array}{rlll} & \sum_{j \in C} x_j & \leq 2 & \text{valid for } X^{BK} \cap \{x \in \{0,1\}^4 : x_1 = 0\} \\ (I) & \sum_{j \in C} x_j + \alpha_1 x_1 & \leq 2 & \text{valid for } X^{BK} \end{array}$$

Step 2: Inequality (I) is valid for $X^{BK} \cap \{x \in \{0,1\}^4 : x_1 = 1\}$

$$\Leftrightarrow \sum_{j \in C} x_j + \alpha_1 \cdot 1 \leq 2 \text{ is valid for } \{x \in \{0,1\}^4 : 6x_2 + 2x_3 + 2x_4 \leq 8 - 5\}$$

$$\Leftrightarrow \max\{ \sum_{j \in C} x_j \; : \; 6x_2 + 2x_3 + 2x_4 \leq 3, \; x \in \{0,1\}^4 \} + \alpha_1 \cdot 1 \leq 2$$

$$\Leftrightarrow 1 + \alpha_1 \leq 2 \Leftrightarrow \alpha_1 \leq 1$$

Sequential Up-lifting

- $X^{BK} = \{x \in \{0,1\}^4 : 5x_1 + 6x_2 + 2x_3 + 2x_4 \leq 8\}$
- $C = \{2, 3, 4\}$

$$
\begin{array}{llll}
 & \sum_{j \in C} x_j & \leq 2 & \text{valid for } X^{BK} \cap \{x \in \{0,1\}^4 : x_1 = 0\} \\
(I) & \sum_{j \in C} x_j + \alpha_1 x_1 & \leq 2 & \text{valid for } X^{BK}
\end{array}
$$

Step 1: Inequ. (I) valid for $X^{BK} \cap \{x \in \{0,1\}^4 : x_1 = 0\}$ for all $\alpha_1 \in [-\infty, \infty]$

Step 2: Inequ. (I) valid for $X^{BK} \cap \{x \in \{0,1\}^4 : x_1 = 1\}$ for all $\alpha_1 \leq 1$

Result: Inequ. (I) valid for X^{BK} for all $\alpha_1 \leq 1$

Sequential Up-lifting

- $(j_1, \ldots, j_t)$ lifting sequence of the variables in $N \setminus C$
- $X^i := X^{BK} \cap \{x \in \{0,1\}^n : x_{j_{i+1}} = \ldots = x_{j_t} = 0\}$

$$
\begin{aligned}
\sum_{j \in C} x_j & \leq |C| - 1 \quad \text{valid for } X^0 \\
\sum_{j \in C} x_j + \alpha_{j_1} x_{j_1} & \leq |C| - 1 \quad \text{valid for } X^1 \\
& \vdots \\
\sum_{j \in C} x_j + \sum_{k=1}^{t} \alpha_{j_k} x_{j_k} & \leq |C| - 1 \quad \text{valid for } X^t = X^{BK}
\end{aligned}
$$

Sequential Up-lifting

- $(j_1, \dots, j_t)$ lifting sequence of the variables in $N \backslash C$
- $X^i := X^{BK} \cap \{x \in \{0,1\}^n : x_{j_{i+1}} = \dots = x_{j_t} = 0\}$

$$
\begin{aligned}
\sum_{j \in C} x_j &\le |C| - 1 \quad \text{valid for } X^0 \\
\sum_{j \in C} x_j + \alpha_{j_1} x_{j_1} &\le |C| - 1 \quad \text{valid for } X^1 \\
&\vdots \\
\sum_{j \in C} x_j + \sum_{k=1}^{t} \alpha_{j_k} x_{j_k} &\le |C| - 1 \quad \text{valid for } X^t = X^{BK}
\end{aligned}
$$

- Different lifting sequences may lead to different inequalities!
- Use sequential up- and down-lifting!

Sequential Up- and Down-lifting

Theorem:

If $C \subseteq N$ is a minimal cover for X^{BK} and (C_1, C_2) is any partition of C with $C_1 \neq \emptyset$, then inequality

$$\sum_{j \in C_1} x_j \leq |C_1| - 1$$

defines a facet of

$$\text{conv}(\ X^{BK} \cap \ \{x \in \{0,1\}^n \ : \ \begin{array}{l} x_j = 0 \ \text{ for all } j \in N \backslash C, \\ x_j = 1 \ \text{ for all } j \in C_2\} \ ). \end{array}$$

- Up-lifting: variables in $N \backslash C$
- Down-lifting: variables in C_2

Outline of the Separation Algorithm

Step 1 (Initial cover)

- ▷ Determine an initial cover C for X^{BK}

Step 2 (Minimal cover and partition)

- ▷ Make the initial cover minimal by removing vars from C
- ▷ Find a partition (C_1, C_2) of C with $C_1 \neq \emptyset$

Step 3 (Lifting)

- ▷ Determine a lifting sequence of the variables in $N \setminus C_1$
- ▷ Lift the inequality $\sum_{j \in C_1} x_j \leq |C_1| - 1$ using sequential up- and down-lifting

Cutting Plane Separation in SCIP

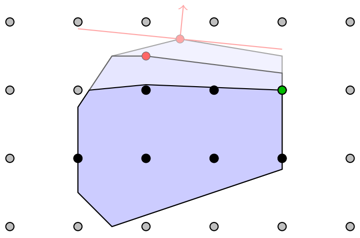
Techniques

▷ General cuts:

- complemented MIR cuts
- Gomory mixed integer cuts
- strong Chvátal-Gomory cuts
- $\{0, \frac{1}{2}\}$-cuts
- implied bound cuts

▷ Problem specific cuts:

- 0-1 knapsack problem
- stable set problem
- 0-1 single node flow problem
- multi-commodity-flow problem



Results

▷ Very important component

▷ In particular, c-MIR cuts

▷ Coordination important

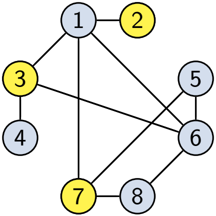
Cuts for the Stable Set Problem

Stable set polytope for graph $G = (V, E)$:

$$\text{conv}(\ \{x \in \{0,1\}^{|V|}\ :\ x_u + x_v \leq 1 \text{ for all } (u,v) \in E\ )$$

Stable Set: $S \subseteq V$

- $\forall\, u, v \in S :\ (u, v) \notin E$



Stable set

Stable set polytope for graph $G = (V, E)$:

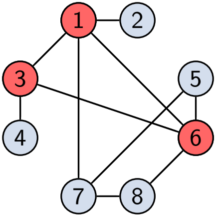
$$\text{conv}(\ \{x \in \{0,1\}^{|V|}\ :\ x_u + x_v \leq 1 \text{ for all } (u,v) \in E\ )$$

Stable Set: $S \subseteq V$

- $\forall\, u, v \in S :\ (u, v) \notin E$

Clique: $C \subseteq V$

- $\forall\, u, v \in C :\ (u, v) \in E$



Clique

Cuts for the Stable Set Problem

Stable set polytope for graph $G = (V, E)$:

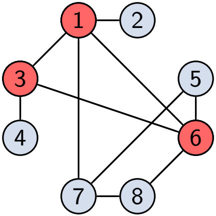
$$\text{conv}(\ \{x \in \{0,1\}^{|V|}\ :\ x_u + x_v \leq 1 \text{ for all } (u,v) \in E\ )$$

Stable Set: $S \subseteq V$

- $\forall\, u, v \in S :\ (u, v) \notin E$

Clique: $C \subseteq V$

- $\forall\, u, v \in C :\ (u, v) \in E$



$x_1 + x_3 + x_6 \leq 1$

$\rightarrow$ Clique inequality: $\sum_{j \in C} x_j \leq 1$ is valid for stable set polytope.

Stable Set Relaxation of a MIP

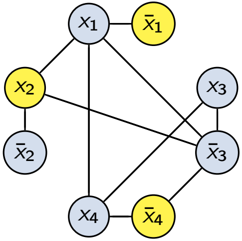
Conflict Graph: $G = (V, E)$

V: node for every binary variable x_j and for its complement $\bar{x}_j := 1 - x_j$

E: $(x_i, x_j) \in E \;\Leftrightarrow\;$ In any feasible MIP solution,
x_i and x_j cannot be one at the same time

$\longrightarrow$ Feasible MIP solution corresponds to stable set in conflict graph

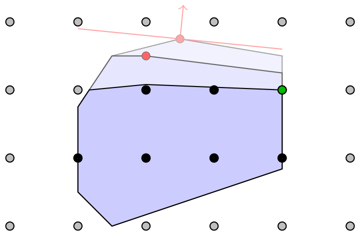
$\longrightarrow$ Stable set polytope on conflict graph is relaxation of MIP's feasible region



Cutting Plane Separation in SCIP

Techniques

- **General cuts:**
  - complemented MIR cuts
  - Gomory mixed integer cuts
  - strong Chvátal-Gomory cuts
  - $\{0, \frac{1}{2}\}$-cuts
  - implied bound cuts
- **Problem specific cuts:**
  - 0-1 knapsack problem
  - stable set problem
  - 0-1 single node flow problem
  - multi-commodity-flow problem



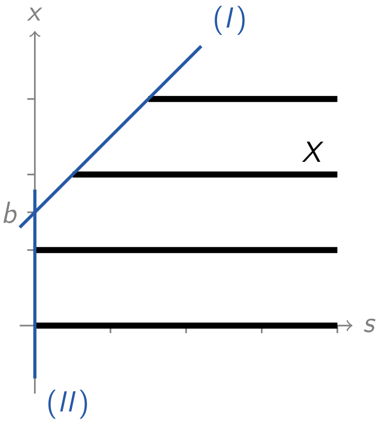
Results

- Very important component
- In particular, c-MIR cuts
- Coordination important

Mixed Integer Rounding (MIR) Cut

Elementary mixed integer set:

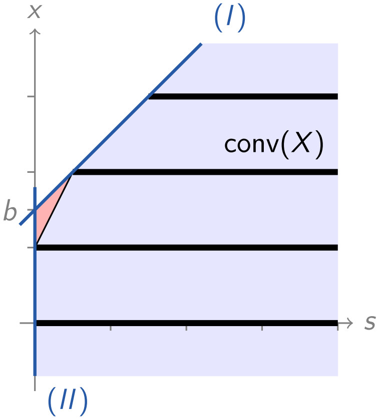
$$X := \{\ (x, s) \in \mathbb{Z} \times \mathbb{R} : \quad x \le b + s \quad (I) \qquad s \ge 0 \quad (II)\ \}$$



Mixed Integer Rounding (MIR) Cut

Elementary mixed integer set:

$$X := \{ (x, s) \in \mathbb{Z} \times \mathbb{R} : \begin{array}{rcll} x & \leq & b + s & (I) \\ s & \geq & 0 & (II) \} \end{array}$$

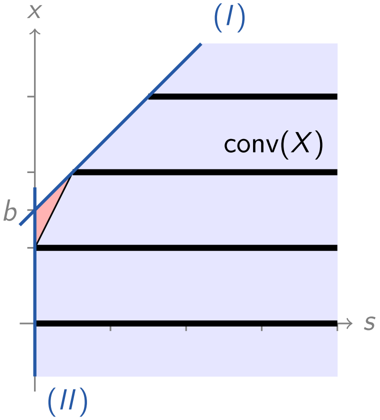


Inequalities (I) and (II) do not suffice to describe $\text{conv}(X)$.

Mixed Integer Rounding (MIR) Cut

Disjunctive argument:

- If an inequality is valid for X^1 and for X^2 it is also valid for $X^1 \cup X^2$.



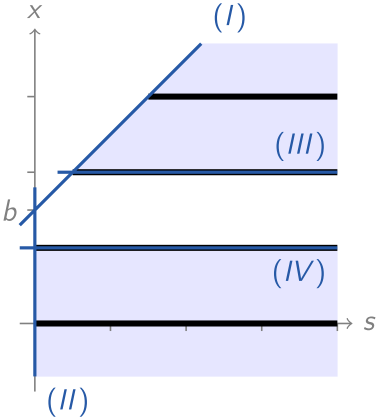
Mixed Integer Rounding (MIR) Cut

Disjunctive argument:

- If an inequality is valid for X^1 and for X^2 it is also valid for $X^1 \cup X^2$.

Here:

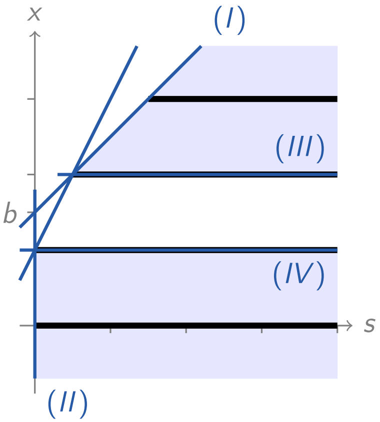
- X^1: Add $x \geq \lceil b \rceil$ (III)
- X^2: Add $x \leq \lfloor b \rfloor$ (IV)



Mixed Integer Rounding (MIR) Cut

Inequality valid for X^1 and for X^2:

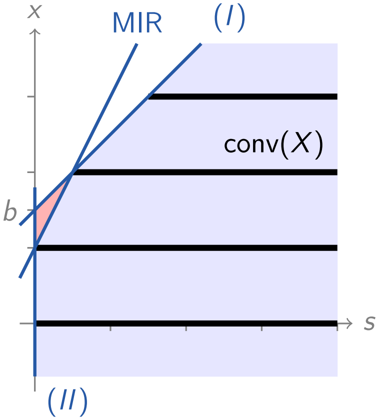
$$\underbrace{x \leq \lfloor b \rfloor + \frac{s}{1 - f_b}}_{(I)+f_b(III) \text{ and } (II)+(1-f_b)(IV)}$$



Mixed Integer Rounding (MIR) Cut

Inequality valid for $X^1 \cup X^2 = X$:

$$\underbrace{x \le \lfloor b \rfloor + \frac{s}{1 - f_b}}_{\text{MIR inequality}}$$



Complemented MIR (C-MIR) Cut

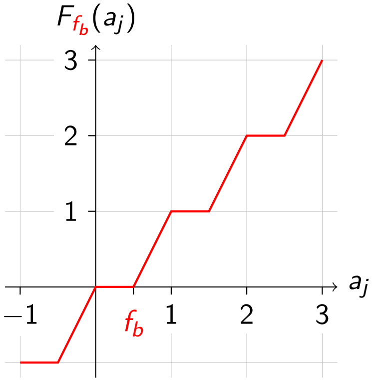
Mixed knapsack set:

$$\{\, (x,s) \in \mathbb{Z}^n_+ \times \mathbb{R}_+ : \quad \sum_{j\in N} a_j x_j \le b + s, \quad x_j \le u_j \quad j \in N \,\}$$

Complemented MIR (C-MIR) Cut

MIR inequality:

$$\sum_{j \in N} F_{f_b}(a_j) x_j \leq \lfloor b \rfloor + \frac{s}{1 - f_b}$$



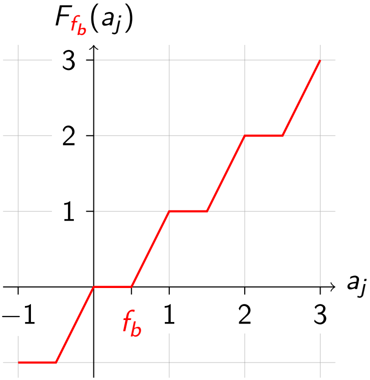
Complemented MIR (C-MIR) Cut

MIR inequality:

$$\sum_{j \in N} F_{f_b}(a_j) x_j \leq \lfloor b \rfloor + \frac{s}{1 - f_b}$$

C-MIR inequality:

- Divide by $\delta \in \mathbb{Q}_+ \setminus \{0\}$
- Complement some integer vars $(x_j = u_j - \bar{x}_j)$
- MIR inequality



Example

$$\sum_{j \in N} a_j x_j \leq b + s \quad \rightsquigarrow \quad \sum_{j \in N} F_{f_b}(a_j) x_j \leq \lfloor b \rfloor + \frac{s}{1 - f_b}$$

Example

$$\sum_{j\in N} a_j x_j \leq b + s \quad \rightsquigarrow \quad \sum_{j\in N} F_{f_b}(a_j) x_j \leq \lfloor b \rfloor + \frac{s}{1 - f_b}$$

$$1x_1 + 4x_2 \leq \tfrac{11}{2} + s$$

Bounds: $x_1, x_2 \leq 2$

Example

$$\sum_{j \in N} a_j x_j \leq b + s$$

$\rightsquigarrow$

$$\sum_{j \in N} F_{f_b}(a_j) x_j \leq \lfloor b \rfloor + \frac{s}{1 - f_b}$$

$$1x_1 + 4x_2 \leq \tfrac{11}{2} + s$$

Bounds: $x_1, x_2 \leq 2$

For $\delta = 1$:

Example

$$\sum_{j\in N} a_j x_j \leq b + s \quad \rightsquigarrow \quad \sum_{j\in N} F_{f_b}(a_j)x_j \leq \lfloor b \rfloor + \frac{s}{1-f_b}$$

$$1x_1 + 4x_2 \leq \tfrac{11}{2} + s$$

Bounds: $x_1, x_2 \leq 2$

For $\delta = 1$:

- $f_{\frac{11}{2}} = \frac{11}{2} - \left\lfloor \frac{11}{2} \right\rfloor = \frac{1}{2}$

Example

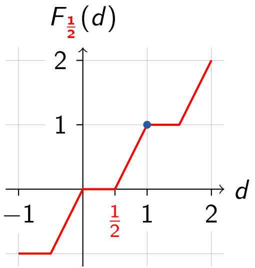
$$\sum_{j \in N} a_j x_j \leq b + s \quad \rightsquigarrow \quad \sum_{j \in N} F_{f_b}(a_j) x_j \leq \lfloor b \rfloor + \frac{s}{1 - f_b}$$

$$1x_1 + 4x_2 \leq \tfrac{11}{2} + s$$

Bounds: $x_1, x_2 \leq 2$

For $\delta = 1$:

- $f_{\frac{11}{2}} = \frac{11}{2} - \left\lfloor \frac{11}{2} \right\rfloor = \frac{1}{2}$



Example

$$\sum_{j \in N} a_j x_j \leq b + s \quad \rightsquigarrow \quad \sum_{j \in N} F_{f_b}(a_j) x_j \leq \lfloor b \rfloor + \frac{s}{1 - f_b}$$

$$1x_1 + 4x_2 \leq \frac{11}{2} + s$$

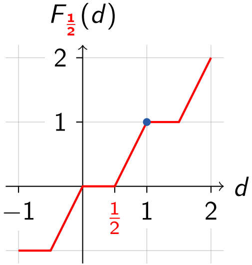
Bounds: $x_1, x_2 \leq 2$

$\rightsquigarrow$

$$1\,x_1 + 4\,x_2 \leq 5 + 2\,s$$

For $\delta = 1$:

▷ $f_{\frac{11}{2}} = \frac{11}{2} - \left\lfloor \frac{11}{2} \right\rfloor = \frac{1}{2}$



Example

$$\sum_{j \in N} a_j x_j \leq b + s \quad \rightsquigarrow \quad \sum_{j \in N} F_{f_b}(a_j) x_j \leq \lfloor b \rfloor + \frac{s}{1 - f_b}$$

$$1x_1 + 4x_2 \leq \tfrac{11}{2} + s$$

Bounds: $x_1, x_2 \leq 2$

$\rightsquigarrow$

$$1\,x_1 + 4\,x_2 \leq 5 + 2\,s$$

For $\delta = 4$, $x_1 = 2 - \bar{x}_1$:

Example

$$\sum_{j \in N} a_j x_j \leq b + s \quad \rightsquigarrow \quad \sum_{j \in N} F_{f_b}(a_j) x_j \leq \lfloor b \rfloor + \frac{s}{1 - f_b}$$

$$1x_1 + 4x_2 \leq \tfrac{11}{2} + s$$

Bounds: $x_1, x_2 \leq 2$

$$\rightsquigarrow \quad 1\,x_1 + 4\,x_2 \leq 5 + 2\,s$$

For $\delta = 4$, $x_1 = 2 - \bar{x}_1$:

- $-\frac{1}{4}\,\bar{x}_1 + x_2 \leq \frac{7}{8} + \frac{1}{4}\,s$

Example

$$\sum_{j \in N} a_j x_j \leq b + s \quad \rightsquigarrow \quad \sum_{j \in N} F_{f_b}(a_j) x_j \leq \lfloor b \rfloor + \frac{s}{1 - f_b}$$

$$1x_1 + 4x_2 \leq \tfrac{11}{2} + s$$

Bounds: $x_1, x_2 \leq 2$

$\rightsquigarrow$

$$1\,x_1 + 4\,x_2 \leq 5 + 2\,s$$

For $\delta = 4$, $x_1 = 2 - \bar{x}_1$:

- $-\frac{1}{4}\,\bar{x}_1 + x_2 \leq \frac{7}{8} + \frac{1}{4}\,s$
- $f_{\frac{7}{8}} = \frac{7}{8} - \left\lfloor \frac{7}{8} \right\rfloor = \frac{7}{8}$

Example

$$\sum_{j \in N} a_j x_j \leq b + s \quad \rightsquigarrow \quad \sum_{j \in N} F_{f_b}(a_j) x_j \leq \lfloor b \rfloor + \frac{s}{1 - f_b}$$

$$1x_1 + 4x_2 \leq \tfrac{11}{2} + s$$

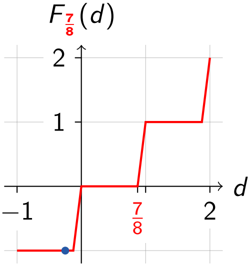
Bounds: $x_1, x_2 \leq 2$

$\rightsquigarrow$

$$1\,x_1 + 4\,x_2 \leq 5 + 2\,s$$

For $\delta = 4$, $x_1 = 2 - \bar{x}_1$:

- $-\frac{1}{4}\,\bar{x}_1 + x_2 \leq \frac{7}{8} + \frac{1}{4}\,s$
- $f_{\frac{7}{8}} = \frac{7}{8} - \left\lfloor \frac{7}{8} \right\rfloor = \frac{7}{8}$



Example

$$\sum_{j \in N} a_j x_j \leq b + s \quad \rightsquigarrow \quad \sum_{j \in N} F_{f_b}(a_j) x_j \leq \lfloor b \rfloor + \frac{s}{1 - f_b}$$

$$1x_1 + 4x_2 \leq \tfrac{11}{2} + s$$

Bounds: $x_1, x_2 \leq 2$

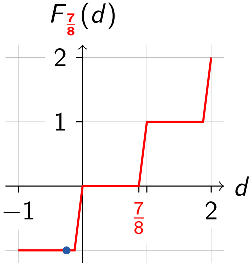
$\rightsquigarrow$

$$1\,x_1 + 4\,x_2 \leq 5 + 2\,s$$

$$-1\,\bar{x}_1 + 1\,x_2 \leq 0 + 2\,s$$

For $\delta = 4$, $x_1 = 2 - \bar{x}_1$:

- $-\frac{1}{4}\,\bar{x}_1 + x_2 \leq \frac{7}{8} + \frac{1}{4}\,s$
- $f_{\frac{7}{8}} = \frac{7}{8} - \left\lfloor \frac{7}{8} \right\rfloor = \frac{7}{8}$



Example

$$\sum_{j \in N} a_j x_j \le b + s \quad \rightsquigarrow \quad \sum_{j \in N} F_{f_b}(a_j) x_j \le \lfloor b \rfloor + \frac{s}{1 - f_b}$$

$$1x_1 + 4x_2 \le \tfrac{11}{2} + s$$

Bounds: $x_1, x_2 \le 2$

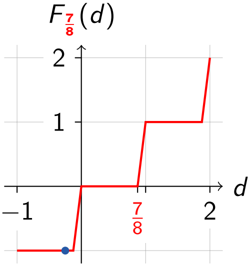
$\rightsquigarrow$

$$1\,x_1 + 4\,x_2 \le 5 + 2\,s$$

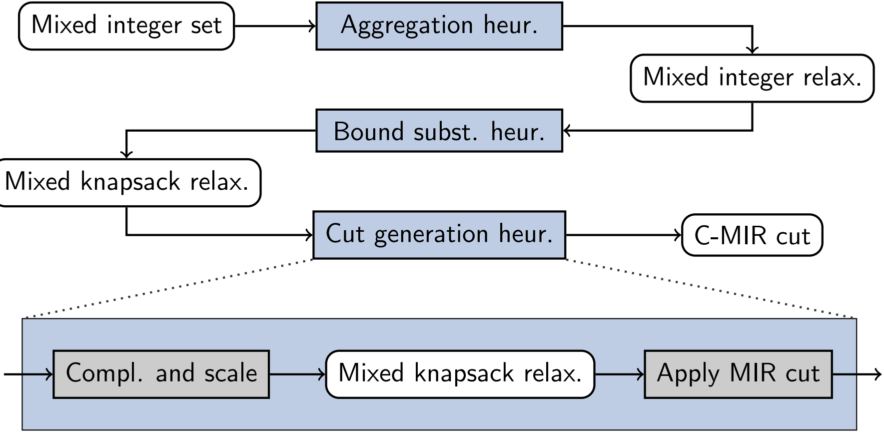
$$1\,x_1 + 1\,x_2 \le 2 + 2\,s$$

For $\delta = 4$, $x_1 = 2 - \bar{x}_1$:

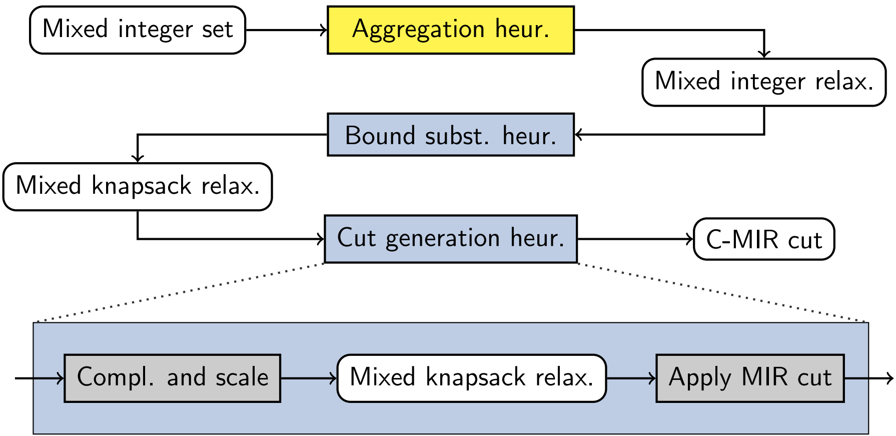
- $-\frac{1}{4}\,\bar{x}_1 + x_2 \le \frac{7}{8} + \frac{1}{4}\,s$
- $f_{\frac{7}{8}} = \frac{7}{8} - \left\lfloor \frac{7}{8} \right\rfloor = \frac{7}{8}$



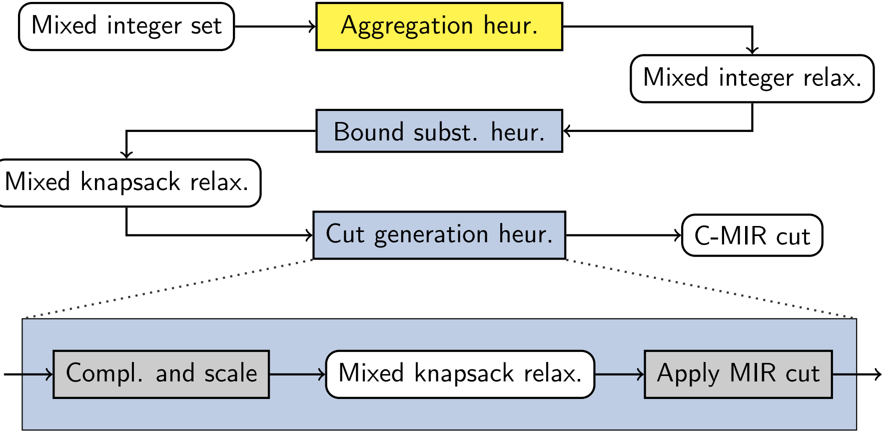
Outline of the Separation Algorithm



Outline of the Separation Algorithm

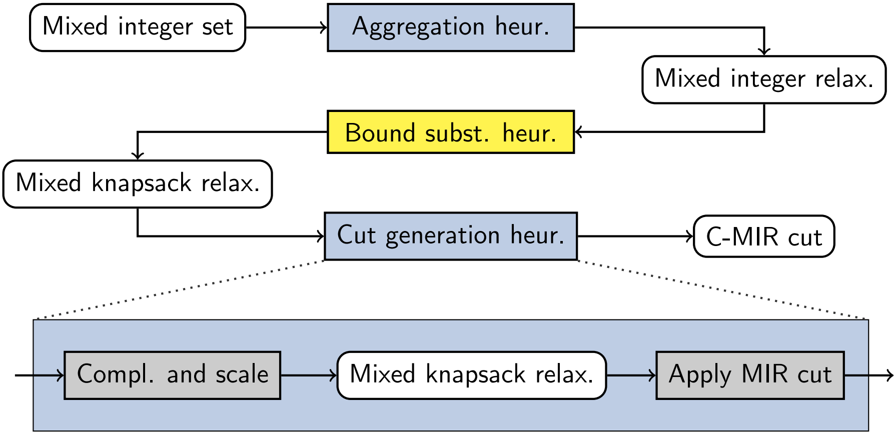


Outline of the Separation Algorithm

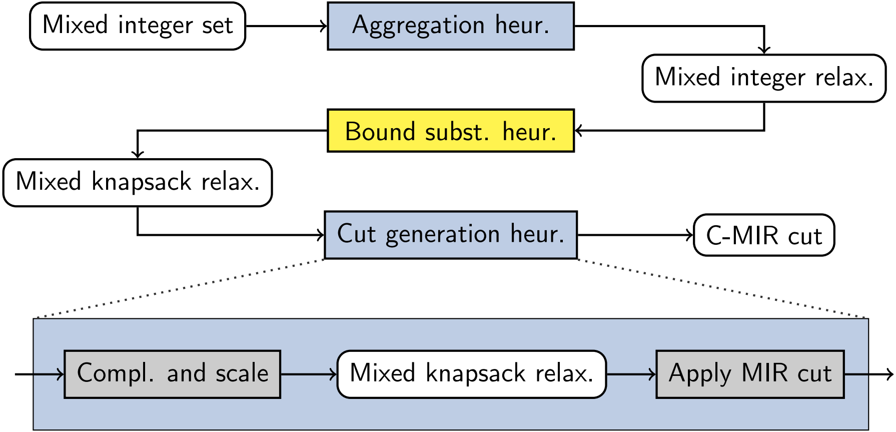


- Linear combination of constraints defining the mixed integer set

Outline of the Separation Algorithm

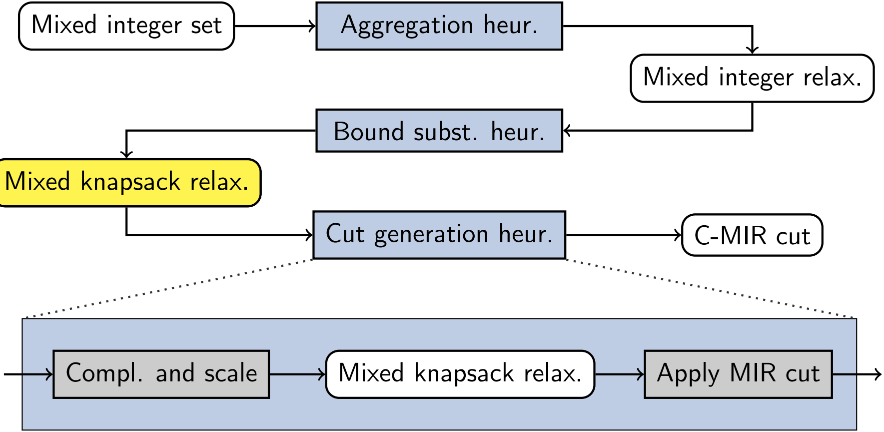


Outline of the Separation Algorithm

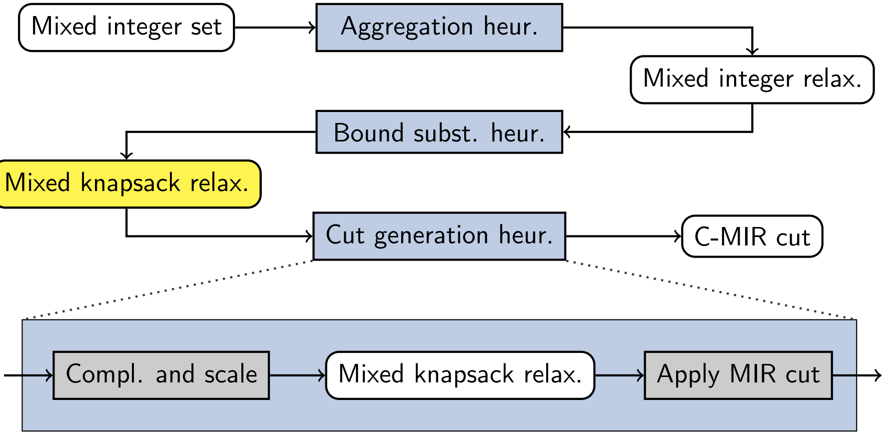


- Perform bound substitution for all real vars (e.g., $y_j = l_j + \bar{y}_j$)
- Relax corresponding set to a mixed knapsack set

Outline of the Separation Algorithm

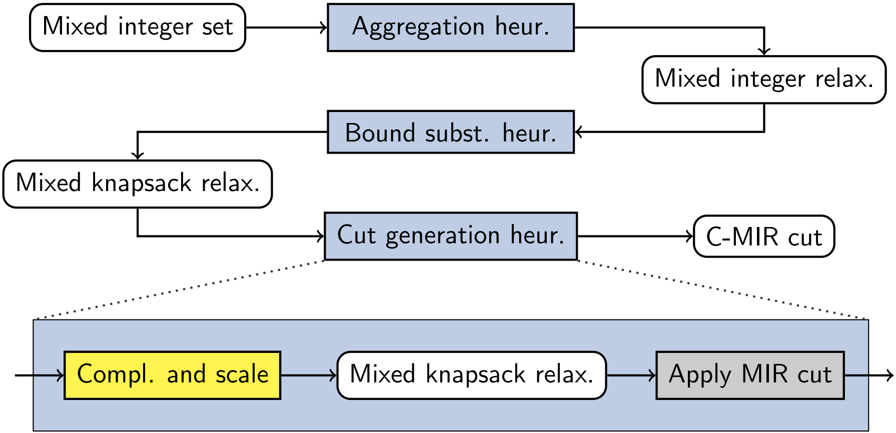


Outline of the Separation Algorithm

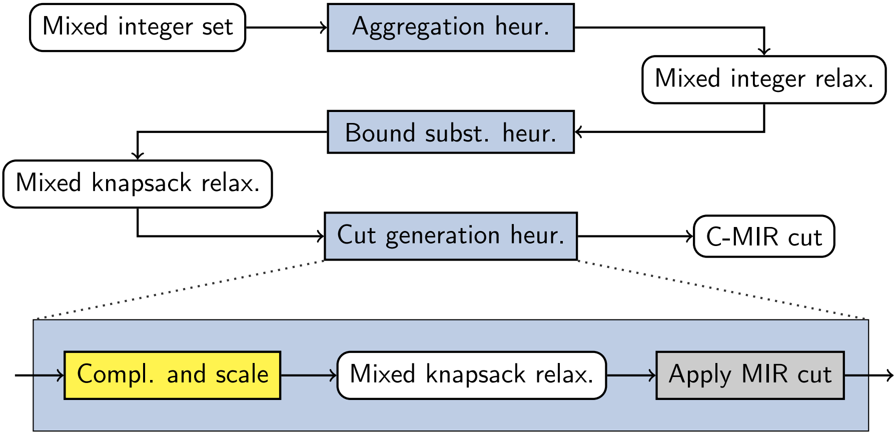


$$\left\{ (x,s) \in \mathbb{Z}^n_+ \times \mathbb{R}_+ : \begin{array}{rcll} \sum_{j\in N} a_j x_j & \le & b + s & \\ x_j & \le & u_j & \quad j \in N \end{array} \right\}$$

Outline of the Separation Algorithm

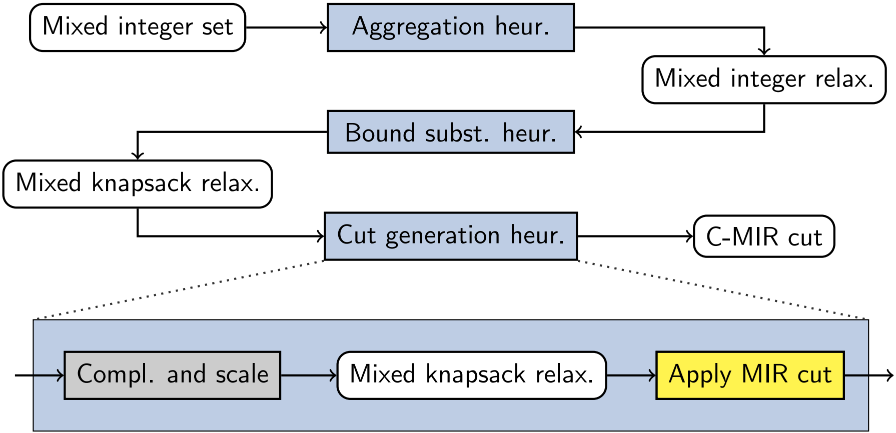


Outline of the Separation Algorithm

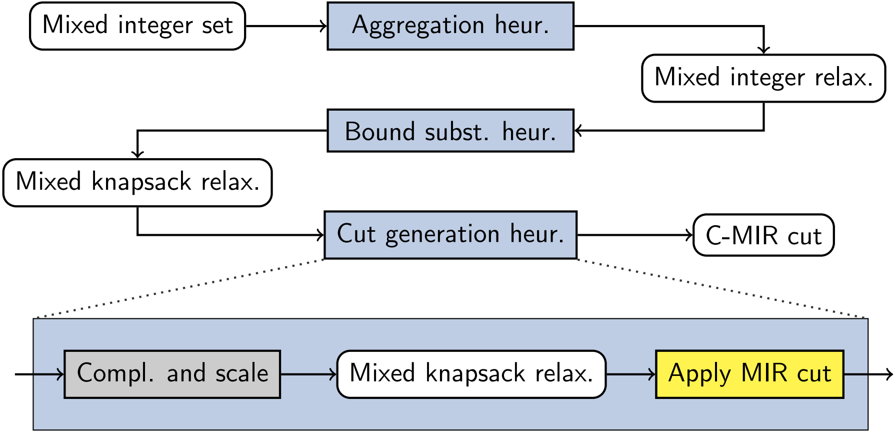


- Complement some integer vars
- Divide constraint by $\delta \in \mathbb{Q}_+ \setminus \{0\}$

Outline of the Separation Algorithm

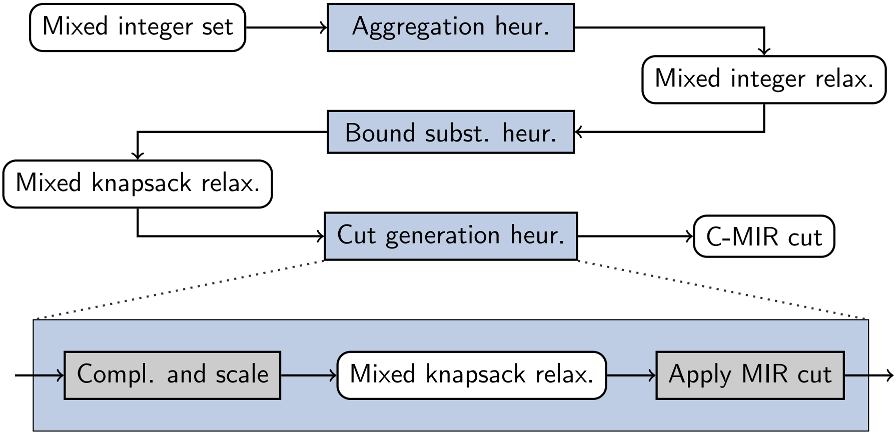


Outline of the Separation Algorithm



$$\sum_{j \in N} F_{f_b}(a_j) x_j \leq \lfloor b \rfloor + \frac{s}{1 - f_b}$$

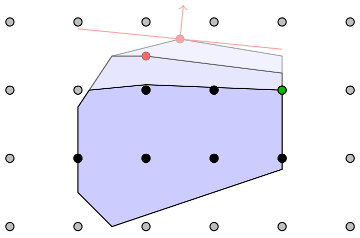
Outline of the Separation Algorithm



Cutting Plane Separation in SCIP

Techniques

- General cuts:
  - complemented MIR cuts
  - Gomory mixed integer cuts
  - strong Chvátal-Gomory cuts
  - $\{0, \frac{1}{2}\}$-cuts
  - implied bound cuts
- Problem specific cuts:
  - 0-1 knapsack problem
  - stable set problem
  - 0-1 single node flow problem
  - multi-commodity-flow problem



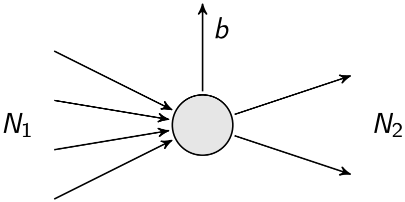
Results

- Very important component
- In particular, c-MIR cuts
- Coordination important

0-1 Single Node Flow Set

Single node with ...

- external demand of b
- inflow arcs $j \in N_1$
- outflow arcs $j \in N_2$



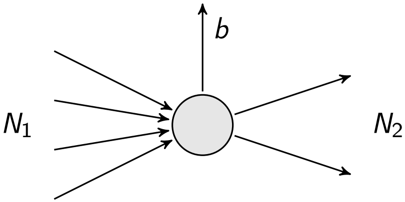
0-1 Single Node Flow Set

Single node with ...

- external demand of b
- inflow arcs $j \in N_1$
- outflow arcs $j \in N_2$

Flow has to satisfy ...

- flow conservation constraint
- capacities on open arcs



$$\begin{aligned} \sum_{j \in N_1} y_j - \sum_{j \in N_2} y_j &\le b \\ 0 \le y_j &\le u_j x_j \\ x_j &\in \{0, 1\} \end{aligned}$$

0-1 Single Node Flow Set

$$\Big\{\, (x,y) \in \{0,1\}^n \times \mathbb{R}^n \;:\; \sum_{j \in N_1} y_j - \sum_{j \in N_2} y_j \;\leq\; b, \quad 0 \;\leq\; y_j \;\leq\; u_j x_j \text{ for all } j \in N \,\Big\}$$

- (N_1, N_2) partition of $N = \{1, \ldots, n\}$
- $b \in \mathbb{Q}$ and $u \in \mathbb{Q}^n_+$

Basic Structure

Set of arcs s.t. flow conservation constraint is violated, if ...

- only these arcs are open
- for each open arc: flow equals capacity

Basic Structure

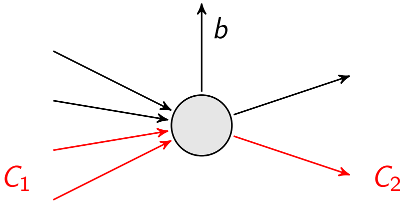
Set of arcs s.t. flow conservation constraint is violated, if ...

- only these arcs are open
- for each open arc: flow equals capacity

Flow cover (C_1, C_2):

- $C_1 \subseteq N_1$ and $C_2 \subseteq N_2$
- $$\sum_{j \in C_1} u_j - \sum_{j \in C_2} u_j = b + \lambda,$$

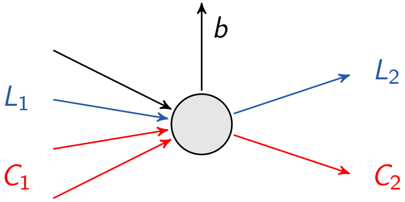
  where $\lambda > 0$.



Generalized Flow Cover Inequality (GFCI)

Basis:

- ▷ Flow cover (C_1, C_2)
- ▷ Sets $L_i \subseteq N_i \setminus C_i$ for $i = 1, 2$
- ▷ Constant $\bar{u} \geq \max\{\lambda, \max_{j \in C_1} u_j\}$



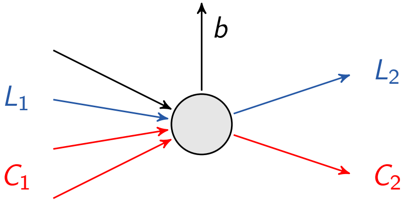
Generalized Flow Cover Inequality (GFCI)

Basis:

- Flow cover (C_1, C_2)
- Sets $L_i \subseteq N_i \setminus C_i$ for $i = 1, 2$
- Constant $\bar{u} \geq \max\{\lambda, \max_{j \in C_1} u_j\}$

Emphasized special cases:

- SGFCI: $\bar{u} = \infty,\ L_1 = \emptyset$
- EGFCI: $\bar{u} = \max_{j \in C_1} u_j > \lambda$



Generalized Flow Cover Inequality (GFCI)

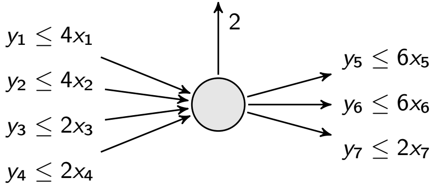
SGFCI ($\bar{u} = \infty,\ L_1 = \emptyset$):

$$
\begin{aligned}
& \sum_{j \in C_1} y_j + (u_j - \lambda)^+ (1 - x_j) \\
- & \sum_{j \in C_2} u_j \\
\\
- & \sum_{j \in L_2} \min\{u_j, \lambda\}\, x_j \\
- & \sum_{j \in N_2 \setminus (C_2 \cup L_2)} y_j \\
\leq \ & b
\end{aligned}
$$

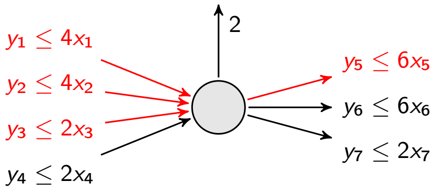
EGFCI ($\bar{u} = \max_{j \in C_1} u_j > \lambda$):

$$
\begin{aligned}
& \sum_{j \in C_1} y_j + (u_j - \lambda)^+ (1 - x_j) \\
- & \sum_{j \in C_2} u_j - \min\{\lambda, (u_j - \bar{u} + \lambda)^+\}\, (1 - x_j) \\
+ & \sum_{j \in L_1} y_j - (\max\{\bar{u}, u_j\} - \lambda)\, x_j \\
- & \sum_{j \in L_2} \min\{u_j, \max\{u_j - \bar{u} + \lambda, \lambda\}\}\, x_j \\
- & \sum_{j \in N_2 \setminus (C_2 \cup L_2)} y_j \\
\leq \ & b
\end{aligned}
$$

Example


$y_1 \le 4x_1$
$y_2 \le 4x_2$
$y_3 \le 2x_3$
$y_4 \le 2x_4$
2
$y_5 \le 6x_5$
$y_6 \le 6x_6$
$y_7 \le 2x_7$

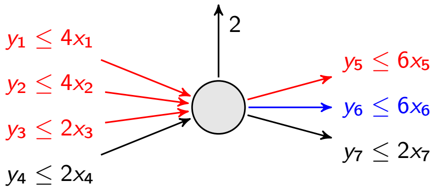
Example



Flow cover:

$C_1 = \{1,2,3\},\ C_2 = \{5\},\ \lambda = 2$

Example

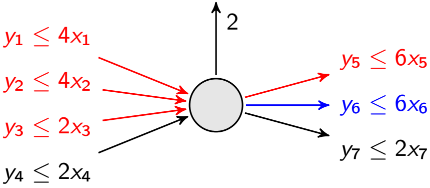


Flow cover:

$C_1 = \{1, 2, 3\},\ C_2 = \{5\},\ \lambda = 2$

Sets: $L_1 = \emptyset,\ L_2 = \{6\}$

Example



Flow cover:

$C_1 = \{1, 2, 3\},\ C_2 = \{5\},\ \lambda = 2$

Sets: $L_1 = \emptyset,\ L_2 = \{6\}$

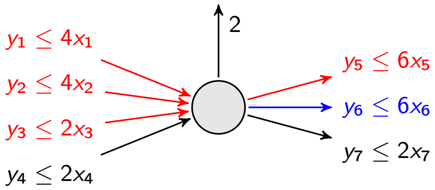
SGFCI ($\bar{u} = \infty,\ L_1 = \emptyset$):

$$
\begin{aligned}
& \textstyle\sum_{j \in C_1} y_j + (u_j - \lambda)^+ (1 - x_j) \\
- & \textstyle\sum_{j \in C_2} u_j \\
- & \textstyle\sum_{j \in L_2} \min\{u_j, \lambda\} x_j \\
- & \textstyle\sum_{j \in N_2 \setminus (C_2 \cup L_2)} y_j \\
\leq & \ b
\end{aligned}
$$

EGFCI ($\bar{u} = \max_{j \in C_1} u_j > \lambda$):

$$
\begin{aligned}
& \textstyle\sum_{j \in C_1} y_j + (u_j - \lambda)^+ (1 - x_j) \\
- & \textstyle\sum_{j \in C_2} u_j - \min\{\lambda, (u_j - \bar{u} + \lambda)^+\} (1 - x_j) \\
+ & \textstyle\sum_{j \in L_1} y_j - (\max\{\bar{u}, u_j\} - \lambda) x_j \\
- & \textstyle\sum_{j \in L_2} \min\{u_j, \max\{u_j - \bar{u} + \lambda, \lambda\}\} x_j \\
- & \textstyle\sum_{j \in N_2 \setminus (C_2 \cup L_2)} y_j \\
\leq & \ b
\end{aligned}
$$

Example



Flow cover:

$C_1 = \{1,2,3\},\ C_2 = \{5\},\ \lambda = 2$

Sets: $L_1 = \emptyset,\ L_2 = \{6\}$

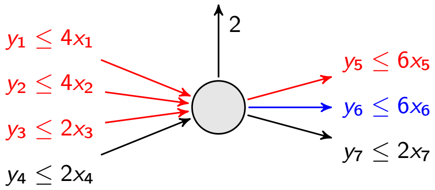
SGFCI ($\bar{u} = \infty,\ L_1 = \emptyset$):

$$
\begin{aligned}
& \textstyle\sum_{j\in C_1} y_j + (u_j - \lambda)^+ (1 - x_j) \\
- & \textstyle\sum_{j\in C_2} u_j \\
- & \textstyle\sum_{j\in L_2} \min\{u_j, \lambda\}\, x_j \\
- & \textstyle\sum_{j\in N_2\setminus(C_2\cup L_2)} y_j \\
\le & \ b
\end{aligned}
$$

EGFCI ($\bar{u} = 4 > 2$):

$$
\begin{aligned}
& \textstyle\sum_{j\in C_1} y_j + (u_j - \lambda)^+ (1 - x_j) \\
- & \textstyle\sum_{j\in C_2} u_j - \min\{\lambda, (u_j - \bar{u} + \lambda)^+\}\,(1 - x_j) \\
+ & \textstyle\sum_{j\in L_1} y_j - (\max\{\bar{u}, u_j\} - \lambda)\, x_j \\
- & \textstyle\sum_{j\in L_2} \min\{u_j, \max\{u_j - \bar{u} + \lambda, \lambda\}\}\, x_j \\
- & \textstyle\sum_{j\in N_2\setminus(C_2\cup L_2)} y_j \\
\le & \ b
\end{aligned}
$$

Example



Flow cover:

$C_1 = \{1,2,3\},\ C_2 = \{5\},\ \lambda = 2$

Sets: $L_1 = \emptyset,\ L_2 = \{6\}$

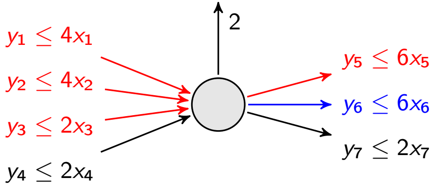
SGFCI ($\bar{u} = \infty,\ L_1 = \emptyset$):

$$
\begin{aligned}
& y_1 + 2\,(1 - x_1) + \\
& y_2 + 2\,(1 - x_2) + y_3 \\
- & \textstyle\sum_{j \in C_2} u_j \\
- & \textstyle\sum_{j \in L_2} \min\{u_j, \lambda\}\, x_j \\
- & \textstyle\sum_{j \in N_2 \setminus (C_2 \cup L_2)} y_j \\
\leq & \ b
\end{aligned}
$$

EGFCI ($\bar{u} = 4 > 2$):

$$
\begin{aligned}
& y_1 + 2\,(1 - x_1) + \\
& y_2 + 2\,(1 - x_2) + y_3 \\
- & \textstyle\sum_{j \in C_2} u_j - \min\{\lambda, (u_j - \bar{u} + \lambda)^+\}\,(1 - x_j) \\
+ & \textstyle\sum_{j \in L_1} y_j - (\max\{\bar{u}, u_j\} - \lambda)\, x_j \\
- & \textstyle\sum_{j \in L_2} \min\{u_j, \max\{u_j - \bar{u} + \lambda, \lambda\}\}\, x_j \\
- & \textstyle\sum_{j \in N_2 \setminus (C_2 \cup L_2)} y_j \\
\leq & \ b
\end{aligned}
$$

Example



Flow cover:

$C_1 = \{1,2,3\},\ C_2 = \{5\},\ \lambda = 2$

Sets: $L_1 = \emptyset,\ L_2 = \{6\}$

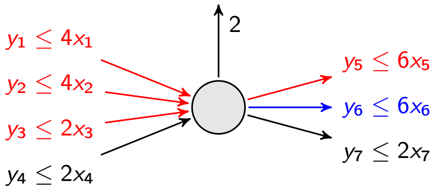
SGFCI ($\bar{u} = \infty,\ L_1 = \emptyset$):

$$\begin{aligned}
& y_1 + 2\,(1 - x_1) + \\
& y_2 + 2\,(1 - x_2) + y_3 \\
- \ & 6 \\
- \ & \textstyle\sum_{j \in L_2} \min\{u_j, \lambda\}\, x_j \\
- \ & \textstyle\sum_{j \in N_2 \setminus (C_2 \cup L_2)} y_j \\
\leq \ & b
\end{aligned}$$

EGFCI ($\bar{u} = 4 > 2$):

$$\begin{aligned}
& y_1 + 2\,(1 - x_1) + \\
& y_2 + 2\,(1 - x_2) + y_3 \\
- \ & 6 \ + 2\,(1 - x_5) \\
+ \ & \textstyle\sum_{j \in L_1} y_j - (\max\{\bar{u}, u_j\} - \lambda)\, x_j \\
- \ & \textstyle\sum_{j \in L_2} \min\{u_j, \max\{u_j - \bar{u} + \lambda, \lambda\}\}\, x_j \\
- \ & \textstyle\sum_{j \in N_2 \setminus (C_2 \cup L_2)} y_j \\
\leq \ & b
\end{aligned}$$

Example



Flow cover:

$C_1 = \{1,2,3\},\ C_2 = \{5\},\ \lambda = 2$

Sets: $L_1 = \emptyset,\ L_2 = \{6\}$

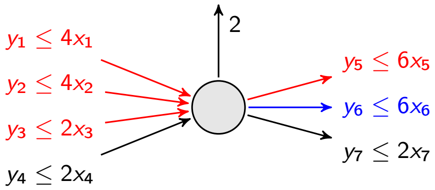
SGFCI ($\bar{u} = \infty,\ L_1 = \emptyset$):

$$
\begin{aligned}
& y_1 + 2\,(1 - x_1) + \\
& y_2 + 2\,(1 - x_2) + y_3 \\
- \; & 6 \\
- \; & \textstyle\sum_{j \in L_2} \min\{u_j, \lambda\}\, x_j \\
- \; & \textstyle\sum_{j \in N_2 \setminus (C_2 \cup L_2)} y_j \\
\leq \; & b
\end{aligned}
$$

EGFCI ($\bar{u} = 4 > 2$):

$$
\begin{aligned}
& y_1 + 2\,(1 - x_1) + \\
& y_2 + 2\,(1 - x_2) + y_3 \\
- \; & 6 \;\; + 2\,(1 - x_5) \\
- \; & \textstyle\sum_{j \in L_2} \min\{u_j, \max\{u_j - \bar{u} + \lambda, \lambda\}\}\, x_j \\
- \; & \textstyle\sum_{j \in N_2 \setminus (C_2 \cup L_2)} y_j \\
\leq \; & b
\end{aligned}
$$

Example



Flow cover:

$C_1 = \{1,2,3\},\ C_2 = \{5\},\ \lambda = 2$

Sets: $L_1 = \emptyset,\ L_2 = \{6\}$

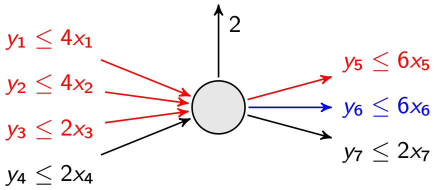
SGFCI ($\bar{u} = \infty,\ L_1 = \emptyset$):

$$\begin{aligned} & y_1 + 2\,(1 - x_1) + \\ & y_2 + 2\,(1 - x_2) + y_3 \\ - \ & 6 \\ - \ & 2x_6 \\ - \ & \textstyle\sum_{j \in N_2 \setminus (C_2 \cup L_2)} y_j \\ \leq \ & b \end{aligned}$$

EGFCI ($\bar{u} = 4 > 2$):

$$\begin{aligned} & y_1 + 2\,(1 - x_1) + \\ & y_2 + 2\,(1 - x_2) + y_3 \\ - \ & 6 \ + 2\,(1 - x_5) \\ - \ & 4x_6 \\ - \ & \textstyle\sum_{j \in N_2 \setminus (C_2 \cup L_2)} y_j \\ \leq \ & b \end{aligned}$$

Example



Flow cover:

$C_1 = \{1, 2, 3\},\ C_2 = \{5\},\ \lambda = 2$

Sets: $L_1 = \emptyset,\ L_2 = \{6\}$

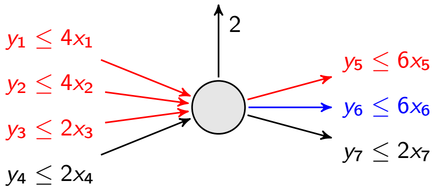
SGFCI ($\bar{u} = \infty,\ L_1 = \emptyset$):

$$
\begin{aligned}
& y_1 + 2\,(1 - x_1) + \\
& y_2 + 2\,(1 - x_2) + y_3 \\
- \ & 6 \\
- \ & 2x_6 \\
- \ & y_7 \\
\leq \ & b
\end{aligned}
$$

EGFCI ($\bar{u} = 4 > 2$):

$$
\begin{aligned}
& y_1 + 2\,(1 - x_1) + \\
& y_2 + 2\,(1 - x_2) + y_3 \\
- \ & 6 \ + 2\,(1 - x_5) \\
- \ & 4x_6 \\
- \ & y_7 \\
\leq \ & b
\end{aligned}
$$

Example



Flow cover:

$C_1 = \{1, 2, 3\},\ C_2 = \{5\},\ \lambda = 2$

Sets: $L_1 = \emptyset,\ L_2 = \{6\}$

SGFCI ($\bar{u} = \infty,\ L_1 = \emptyset$):

$$
\begin{aligned}
&y_1 + 2(1 - x_1) + \\
&y_2 + 2(1 - x_2) + y_3 \\
-\ &6 \\
-\ &2x_6 \\
-\ &y_7 \\
\le\ &2
\end{aligned}
$$

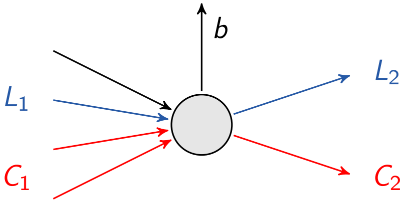
EGFCI ($\bar{u} = 4 > 2$):

$$
\begin{aligned}
&y_1 + 2(1 - x_1) + \\
&y_2 + 2(1 - x_2) + y_3 \\
-\ &6 \ + 2(1 - x_5) \\
-\ &4x_6 \\
-\ &y_7 \\
\le\ &2
\end{aligned}
$$

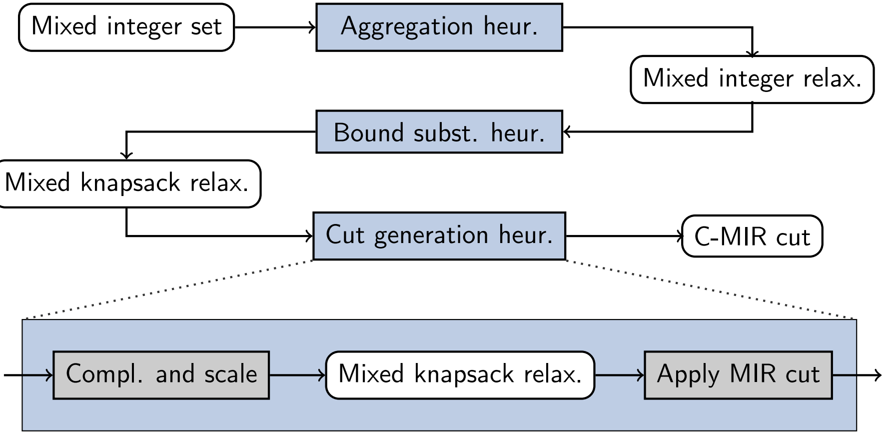
C-MIR Flow Cover Inequality (C-MIRFCI)

Basis:

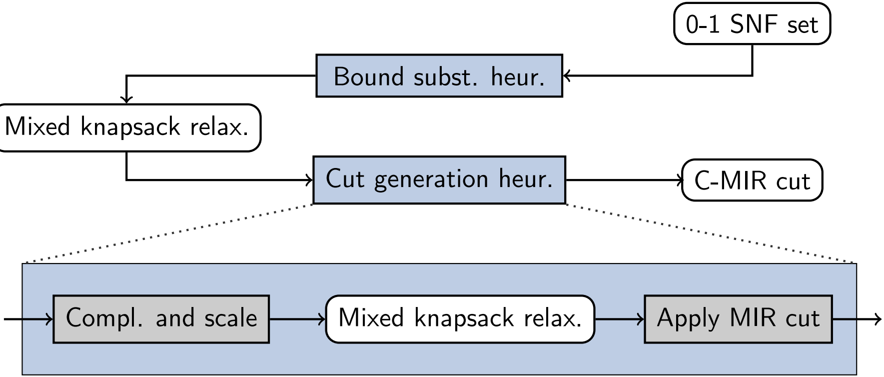
- Flow cover (C_1, C_2)
- Sets $L_i \subseteq N_i \backslash C_i$ for $i = 1, 2$
- Constant $\bar{u} > \lambda$



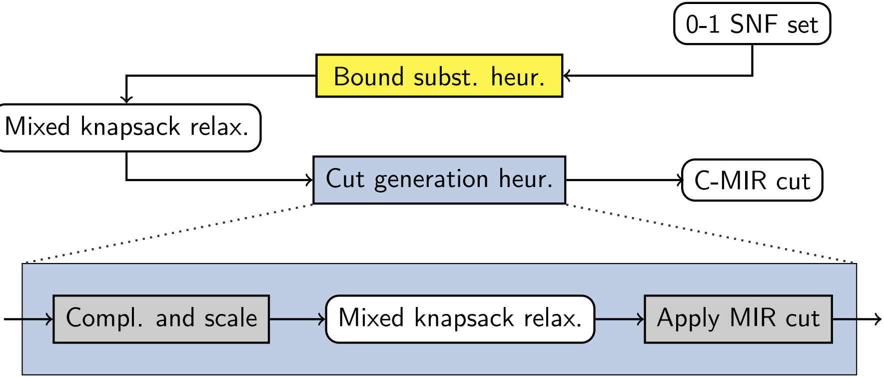
C-MIR Flow Cover Inequality (C-MIRFCI)



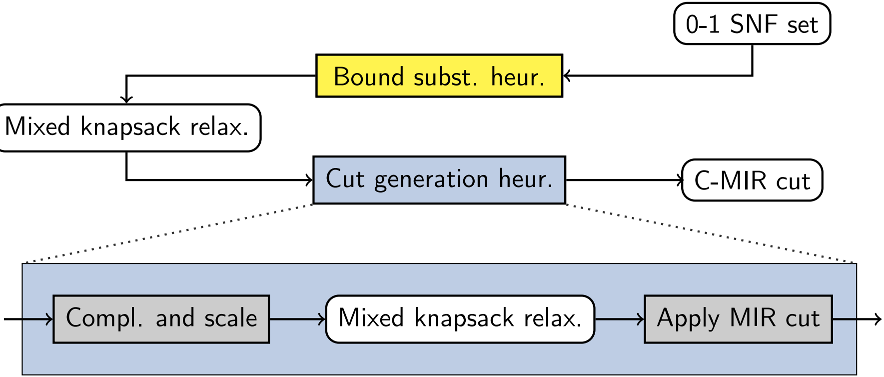
C-MIR Flow Cover Inequality (C-MIRFCI)



C-MIR Flow Cover Inequality (C-MIRFCI)

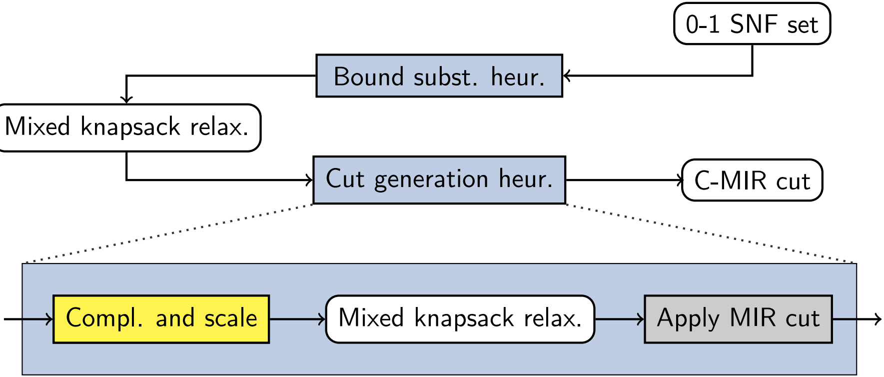


C-MIR Flow Cover Inequality (C-MIRFCI)

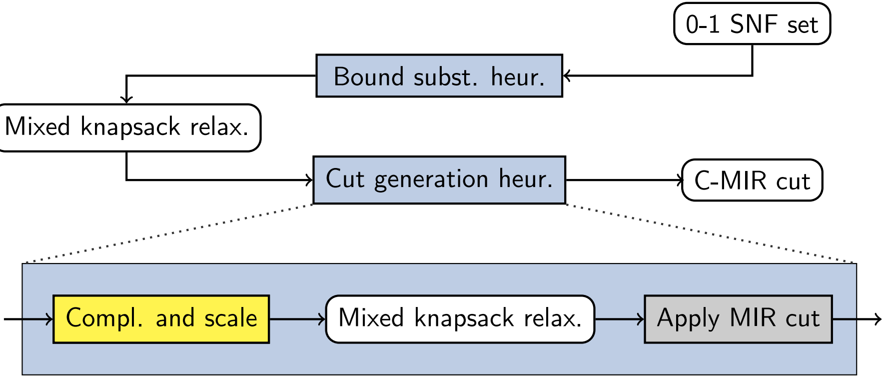


▷ Substitute $y_j = \begin{cases} u_j x_j - \bar{y}_j & : \quad j \in C_1 \cup C_2 \cup L_1 \cup L_2 \\ 0 + \bar{y}_j & : \quad \text{otherwise} \end{cases}$

C-MIR Flow Cover Inequality (C-MIRFCI)

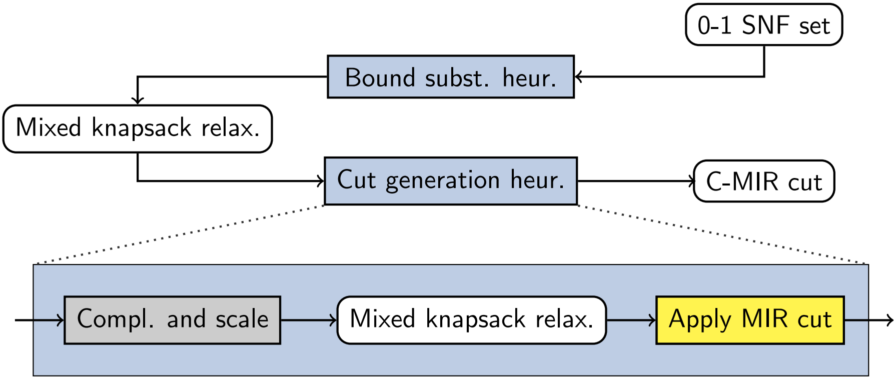


C-MIR Flow Cover Inequality (C-MIRFCI)

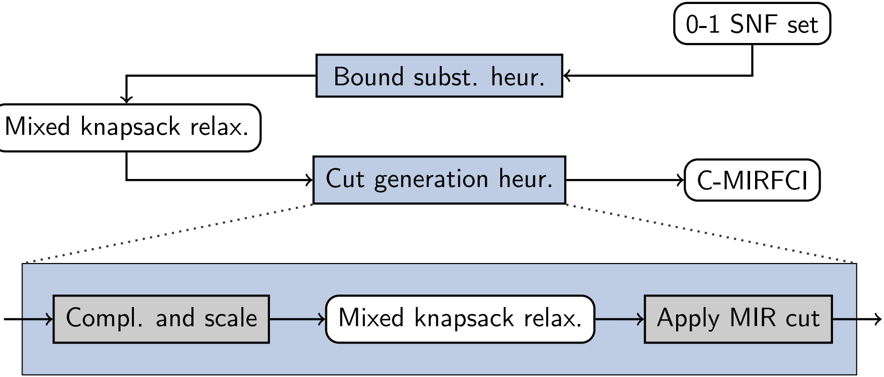


- Complement all integer vars in $U = C_1 \cup C_2$
- Divide constraint by $\delta = \bar{u}$

C-MIR Flow Cover Inequality (C-MIRFCI)



C-MIR Flow Cover Inequality (C-MIRFCI)



Dominance Relations

$$
\begin{tikzcd}
\text{Lifted SGFCI} \arrow[r] & \begin{array}{c}\text{SGFCI}\\ (\bar{u} = \infty,\ L_1 = \emptyset)\end{array} & \begin{array}{c}\text{EGFCI}\\ (\bar{u} = \max\limits_{j \in C_1} u_j > \lambda)\end{array} & \text{Lifted FCI} \arrow[l] \arrow[dl, leftrightarrow] \\
& \begin{array}{c}\text{C-MIRFCI}\\ (\bar{u} = \max\limits_{j \in C_1 \cup L_2} u_j > \lambda)\end{array} \arrow[u, "L_1 = \emptyset"'] & \begin{array}{c}\text{C-MIRFCI}\\ (\bar{u} = \max\limits_{j \in C_1} u_j > \lambda)\end{array} \arrow[u] &
\end{tikzcd}
$$

Outline of the Separation Algorithm

For each MIP row:

1. Construct 0-1 SNF relaxation

- Similar to the procedure of Van Roy and Wolsey 1986
- Considers SCIP specific variable bounds

Outline of the Separation Algorithm

For each MIP row:

1. Construct 0-1 SNF relaxation
2. Determine flow cover (C_1, C_2)

Upper bound on violation of weakened SGFCIs:

$$\max\{ \sum_{j \in N_1} (x_j^* - 1) z_j + \sum_{j \in N_2} x_j^* z_j : \sum_{j \in N_1} u_j z_j - \sum_{j \in N_2} u_j z_j > b, \; z_j \in \{0,1\} \text{ for all } j \in N\}$$

Outline of the Separation Algorithm

For each MIP row:

1. Construct 0-1 SNF relaxation
2. Determine flow cover (C_1, C_2)

Default:

- Exact algorithm (after scaling)

Alternatives:

- Heuristic
- Select algo depending on rhs of scaled cons

Extension:

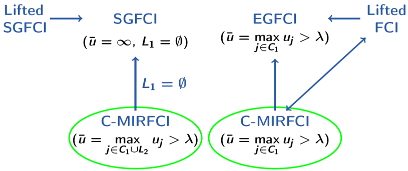
- Fixing strategy

Outline of the Separation Algorithm

For each MIP row:

1. Construct 0-1 SNF relaxation
2. Determine flow cover (C_1, C_2)
3. For different values of $\bar{u}$:

Default:



Extended candidate set:

- $u_j > \lambda$ for all $j \in N$
- $\lambda + 1$
- $\max_{j \in N} u_j + 1 > \lambda$

Outline of the Separation Algorithm

For each MIP row:

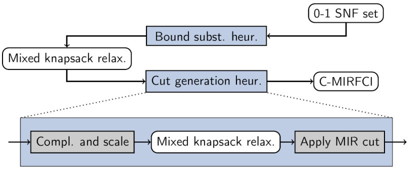
1. Construct 0-1 SNF relaxation
2. Determine flow cover (C_1, C_2)
3. For different values of $\bar{u}$:
   - Determine sets $L_i \subseteq N_i \setminus C_i$ for $i = 1, 2$

▷ Chosen by comparison.

Outline of the Separation Algorithm

For each MIP row:

1. Construct 0-1 SNF relaxation
2. Determine flow cover (C_1, C_2)
3. For different values of $\bar{u}$:
   - Determine sets $L_i \subseteq N_i \setminus C_i$ for $i = 1, 2$
   - Derive c-MIRFCI



Outline of the Separation Algorithm

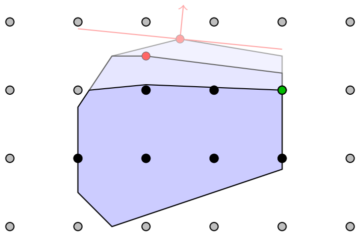
For each MIP row:

1. Construct 0-1 SNF relaxation
2. Determine flow cover (C_1, C_2)
3. For different values of $\bar{u}$:
   - Determine sets $L_i \subseteq N_i \setminus C_i$ for $i = 1, 2$
   - Derive c-MIRFCI
4. Select most violated cut

Cutting Plane Separation in SCIP

Techniques

- General cuts:
  - complemented MIR cuts
  - Gomory mixed integer cuts
  - strong Chvátal-Gomory cuts
  - $\{0, \frac{1}{2}\}$-cuts
  - implied bound cuts
- Problem specific cuts:
  - 0-1 knapsack problem
  - stable set problem
  - 0-1 single node flow problem
  - multi-commodity-flow problem



Results

- Very important component
- In particular, c-MIR cuts
- Coordination important

Cut Selection Strategy

- **Efficacy**, i.e., distance of the hyperplane to the LP solution $\boxed{e_r}$
- **Orthogonality** with respect to the other cuts $\boxed{o_r}$
- **Parallelism** with respect to the objective function $\boxed{p_r}$

$\Rightarrow$ **Select** cuts with largest value of $\boxed{e_r} + \boxed{w_o * o_r} + \boxed{w_p * p_r}$

Cut Selection Strategy

- ▷ **Efficacy**, i.e., distance of the hyperplane to the LP solution $\boxed{e_r}$
- ▷ **Orthogonality** with respect to the other cuts $\boxed{o_r}$
- ▷ **Parallelism** with respect to the objective function $\boxed{p_r}$

$\Rightarrow$ **Select** cuts with largest value of $\boxed{e_r} + \boxed{w_o * o_r} + \boxed{w_p * p_r}$

Consequence

- ▷ Cut as deep as possible into the current LP polyhedron
- ▷ Select cuts that are pairwise almost orthogonal
- ▷ Prefer cuts that are close to being parallel to the objective function

Cut Selection Strategy

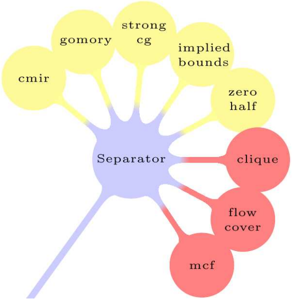
- **Efficacy**, i.e., distance of the hyperplane to the LP solution $\boxed{e_r}$
- **Orthogonality** with respect to the other cuts $\boxed{o_r}$
- **Parallelism** with respect to the objective function $\boxed{p_r}$

⇒ **Select** cuts with largest value of $\boxed{e_r} + \boxed{w_o * o_r} + \boxed{w_p * p_r}$

Weights of the criteria can be adjusted

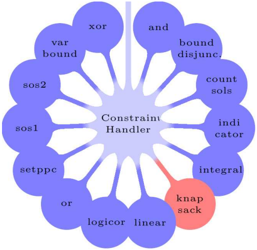
- `ORTHOFAC = 1.0`
- `OBJPARALFAC = 0.0001`

Implementation in SCIP



Separators

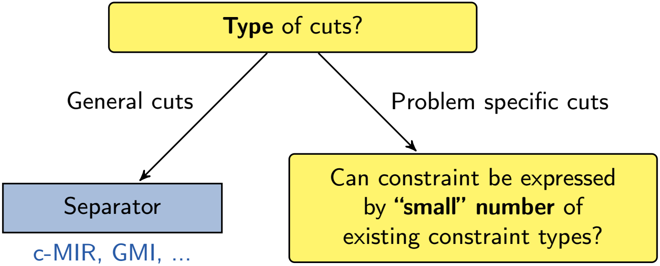
- ▷ provide general cuts
- ▷ provide problem specific cuts



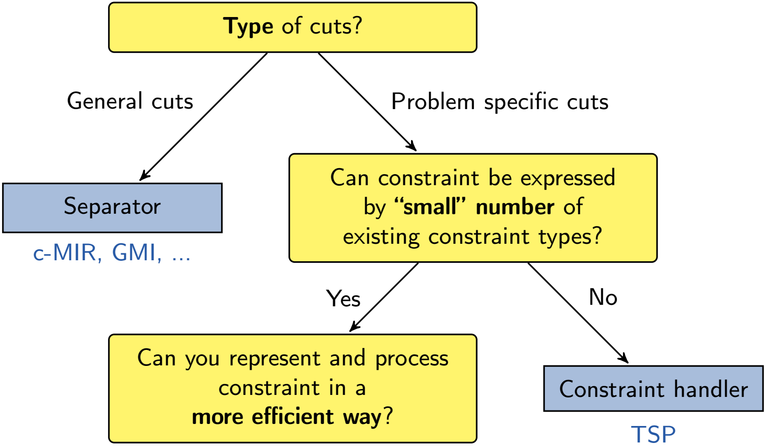
Constraint handlers

- ▷ feasibility check for given solution
- ▷ provide linear relaxation
  (in advance or on the fly)
- ▷ additional problem specific cuts

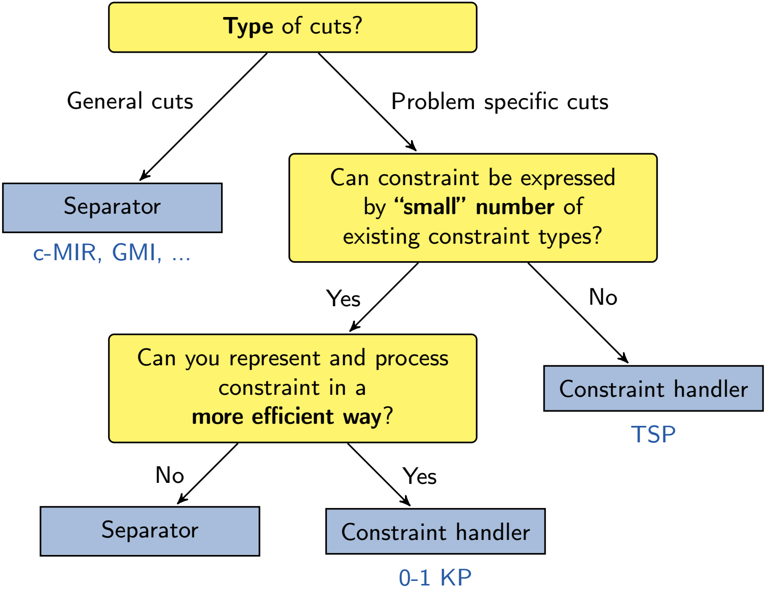
Separator or Constraint Handler?



Separator or Constraint Handler?



Separator or Constraint Handler?



Cutting Planes and Primal Heuristics

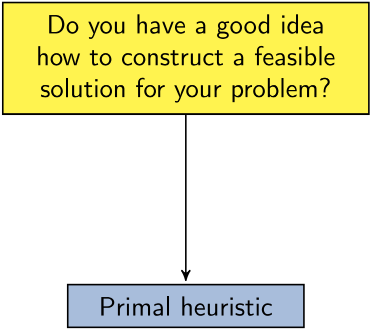
CO@Work Berlin

Timo Berthold and Kati Wolter

09/30/2009

Berlin Mathematical School

DFG Research Center MATHEON
Mathematics for key technologies


Do you have a good idea how to construct a feasible solution for your problem?
Primal heuristic

Do you have a good idea how to construct a feasible solution for your problem?

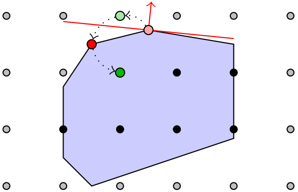
↓

Primal heuristic

Primal heuristics...

- are incomplete methods which
- often find good solutions
- within a reasonable time
- without any warranty!

$\rightsquigarrow$ Integrate into exact solver



Primal Heuristics

Why use primal heuristics inside an exact solver?

- ▷ Able to prove feasibility of the model
- ▷ Often nearly optimal solutions suffice in practice
- ▷ Feasible solutions guide remaining search process

Characteristics

Primal Heuristics

Why use primal heuristics inside an exact solver?

- Able to prove feasibility of the model
- Often nearly optimal solutions suffice in practice
- Feasible solutions guide remaining search process

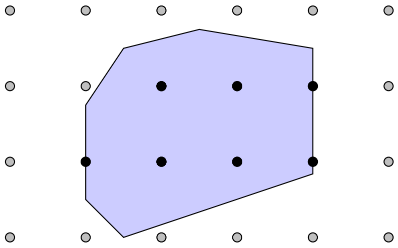
Characteristics

- Main goal: feasible solutions
- Keep control of effort!
- Use as much information as you can get

Useful Information

Statistics & points

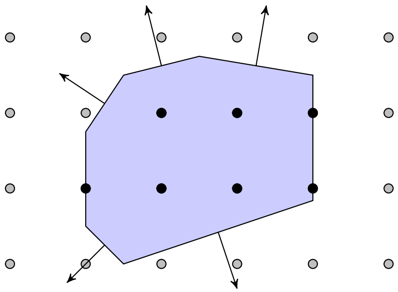
- **Variables' locking numbers:**
  Potentially violated rows
- **Variables' pseudocosts:**
  Average objective change
- **Special points:**
  - LP optimum at root node
  - Current LP solution
  - Current best solution
  - Other known solutions



Useful Information

Statistics & points

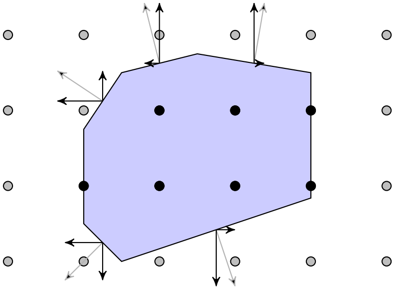
- Variables' locking numbers:
  Potentially violated rows
- Variables' pseudocosts:
  Average objective change
- Special points:
  - LP optimum at root node
  - Current LP solution
  - Current best solution
  - Other known solutions



Useful Information

Statistics & points

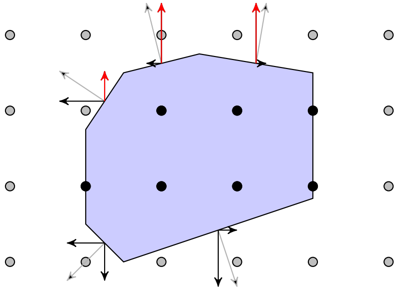
- Variables' locking numbers:
  Potentially violated rows
- Variables' pseudocosts:
  Average objective change
- Special points:
  - LP optimum at root node
  - Current LP solution
  - Current best solution
  - Other known solutions



Useful Information

Statistics & points

- Variables' locking numbers:
  Potentially violated rows
- Variables' pseudocosts:
  Average objective change
- Special points:
  - LP optimum at root node
  - Current LP solution
  - Current best solution
  - Other known solutions

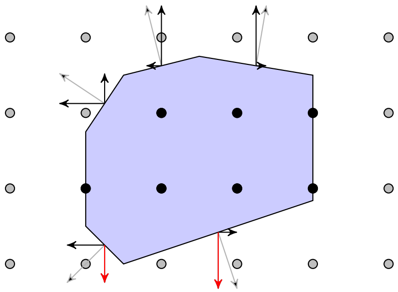


```
int nlocks = SCIPvarGetNLocksUp(var);
```

Useful Information

Statistics & points

- Variables' locking numbers:
  Potentially violated rows
- Variables' pseudocosts:
  Average objective change
- Special points:
  - LP optimum at root node
  - Current LP solution
  - Current best solution
  - Other known solutions

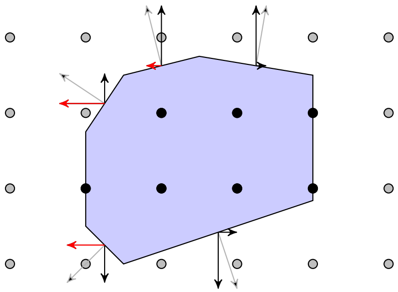


```
int nlocks = SCIPvarGetNLocksDown(var);
```

Useful Information

Statistics & points

- Variables' locking numbers:
  Potentially violated rows
- Variables' pseudocosts:
  Average objective change
- Special points:
  - LP optimum at root node
  - Current LP solution
  - Current best solution
  - Other known solutions

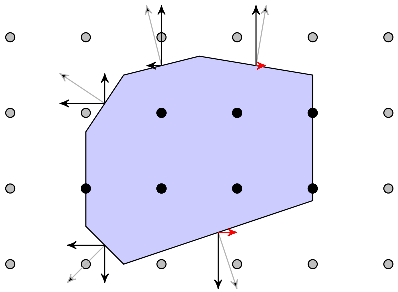


```
int nlocks = SCIPvarGetNLocksDown(var);
```

Useful Information

Statistics & points

- Variables' locking numbers:
  Potentially violated rows
- Variables' pseudocosts:
  Average objective change
- Special points:
  - LP optimum at root node
  - Current LP solution
  - Current best solution
  - Other known solutions

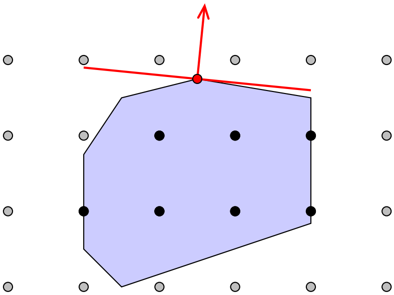


```
int nlocks = SCIPvarGetNLocksUp(var);
```

Useful Information

Statistics & points

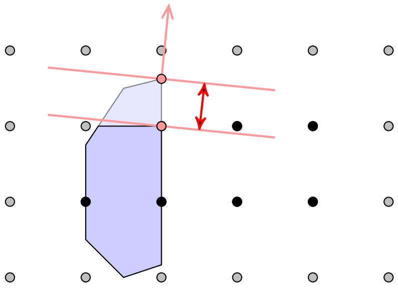
- Variables' locking numbers:
  Potentially violated rows
- Variables' pseudocosts:
  Average objective change
- Special points:
  - LP optimum at root node
  - Current LP solution
  - Current best solution
  - Other known solutions



Useful Information

Statistics & points

- Variables' locking numbers:
  Potentially violated rows
- Variables' pseudocosts:
  Average objective change
- Special points:
  - LP optimum at root node
  - Current LP solution
  - Current best solution
  - Other known solutions

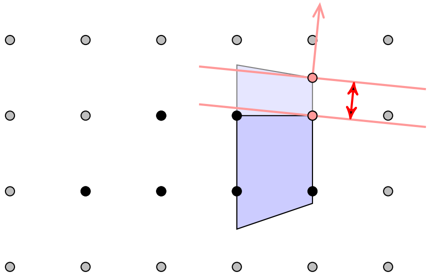


```
SCIP_Real pscost = SCIPgetVarPseudocost(scip, var, delta);
```

Useful Information

Statistics & points

- Variables' locking numbers:
  Potentially violated rows
- Variables' pseudocosts:
  Average objective change
- Special points:
  - LP optimum at root node
  - Current LP solution
  - Current best solution
  - Other known solutions

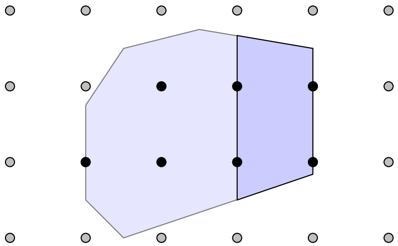


```
SCIP_Real pscost = SCIPgetVarPseudocost(scip, var, delta);
```

Useful Information

Statistics & points

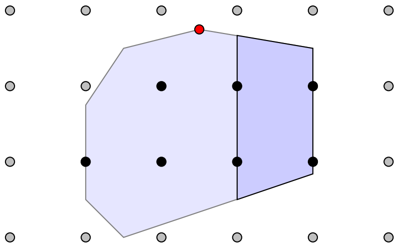
- Variables' locking numbers:
  Potentially violated rows
- Variables' pseudocosts:
  Average objective change
- Special points:
  - LP optimum at root node
  - Current LP solution
  - Current best solution
  - Other known solutions



Useful Information

Statistics & points

- Variables' locking numbers:
  Potentially violated rows
- Variables' pseudocosts:
  Average objective change
- Special points:
  - LP optimum at root node
  - Current LP solution
  - Current best solution
  - Other known solutions

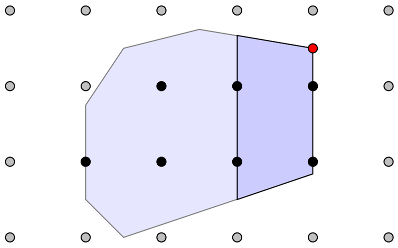


```
SCIP_Real rootsolval = SCIPvarGetRootSol(var);
```

Useful Information

Statistics & points

- Variables' locking numbers:
  Potentially violated rows
- Variables' pseudocosts:
  Average objective change
- Special points:
  - LP optimum at root node
  - Current LP solution
  - Current best solution
  - Other known solutions

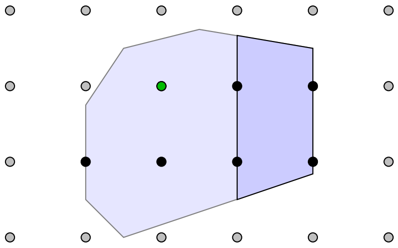


```
SCIP_Real solval = SCIPgetSolVal(scip, NULL, var);
```

Useful Information

Statistics & points

- Variables' locking numbers:
  Potentially violated rows
- Variables' pseudocosts:
  Average objective change
- Special points:
  - LP optimum at root node
  - Current LP solution
  - Current best solution
  - Other known solutions

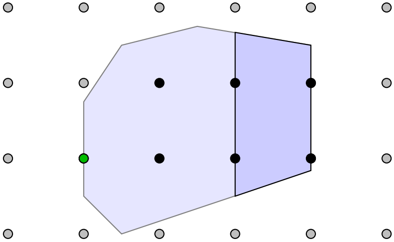


```
SCIP_Sol* bestsol = SCIPgetBestSol(scip);
SCIP_Real solval = SCIPgetSolVal(scip, bestsol, var);
```

Useful Information

Statistics & points

- Variables' locking numbers:
  Potentially violated rows
- Variables' pseudocosts:
  Average objective change
- Special points:
  - LP optimum at root node
  - Current LP solution
  - Current best solution
  - Other known solutions



```
SCIP_Sol** sols = SCIPgetSols(scip);
SCIP_Real solval = SCIPgetSolVal(scip, sols[i], var);
```

Categories

Two main categories

- Start heuristics
  - Applied early in the search process
    - often at root node
  - Mostly start from LP optimum
- Improvement heuristics
  - Require feasible solution
  - Normally at most once for each incumbent

Categories

Two main categories

- Start heuristics
  - Applied early in the search process
    - often at root node
  - Mostly start from LP optimum
- Improvement heuristics
  - Require feasible solution
  - Normally at most once for each incumbent

```
#define HEUR_FREQOFS 0
```

Categories

Two main categories

- Start heuristics
  - Applied early in the search process
    - often at root node
  - Mostly start from LP optimum
- Improvement heuristics
  - Require feasible solution
  - Normally at most once for each incumbent

```
if( SCIPgetLPSolstat(scip) != SCIP_LPSOLSTAT_OPTIMAL )
    return SCIP_OKAY;
```

Categories

Two main categories

- Start heuristics
  - Applied early in the search process
    - often at root node
  - Mostly start from LP optimum
- Improvement heuristics
  - Require feasible solution
  - Normally at most once for each incumbent

```
if( SCIPgetNSols(scip) <= 0 )
   return SCIP_OKAY;
```

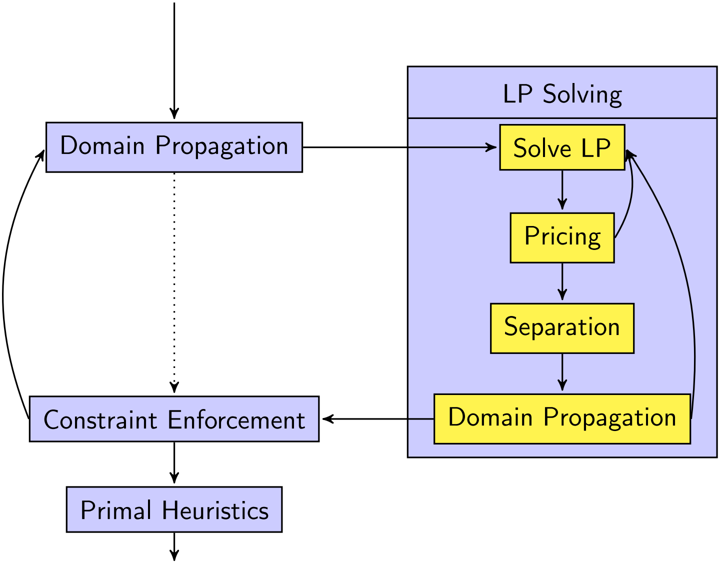
Categories

Two main categories

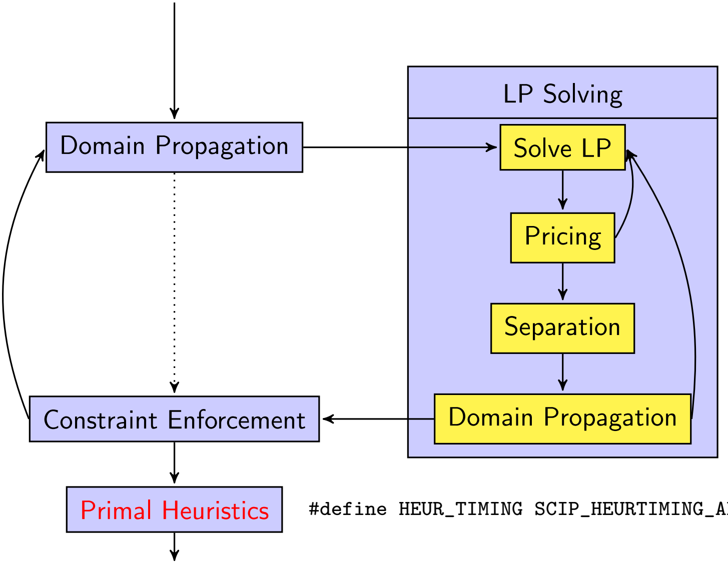
- Start heuristics
  - Applied early in the search process
    - often at root node
  - Mostly start from LP optimum
- Improvement heuristics
  - Require feasible solution
  - Normally at most once for each incumbent

```
struct SCIP_HeurData
{
   SCIP_SOL* lastsol;
}
```

Heuristic Timings

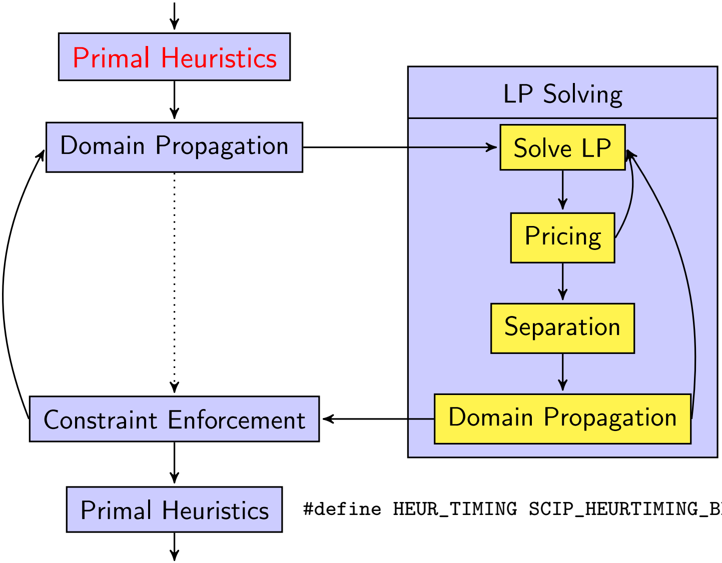


Heuristic Timings



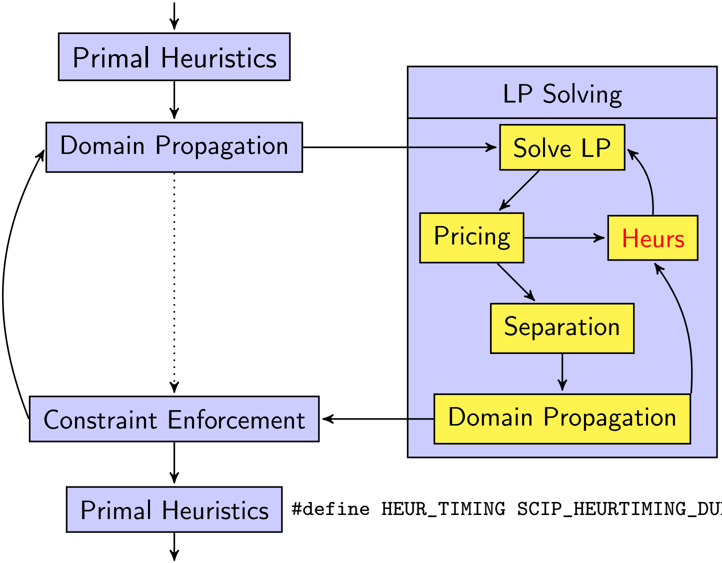

```
#define HEUR_TIMING SCIP_HEURTIMING_AFTERNODE
```

Heuristic Timings




```
#define HEUR_TIMING SCIP_HEURTIMING_BEFORENODE
```

Heuristic Timings



`#define HEUR_TIMING SCIP_HEURTIMING_DURINGLPLOOP`

Primal Heuristics

Approaches

- Rounding: Change fractional to integral values
- Diving: simulate DFS in the branch-and-bound tree using some special branching rule
- Objective diving: manipulate objective function (instead of bounds)
- Large Neighborhood Search: solve some sub-MIP
- Pivoting: manipulate simplex algorithm
- Combinatorial: use special polyhedral properties

Primal Heuristics

Approaches

- ▷ Rounding: Change fractional to integral values
- ▷ Diving: simulate DFS in the branch-and-bound tree using some special branching rule
- ▷ Objective diving: manipulate objective function (instead of bounds)
- ▷ Large Neighborhood Search: solve some sub-MIP
- ▷ Pivoting: manipulate simplex algorithm
- ▷ Combinatorial: use special polyhedral properties

Rounding Heuristics

Guideline: Stay feasible!

Features

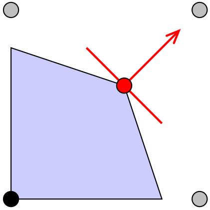
- ▷ Simple Rounding always stays feasible,
- ▷ Rounding may violate constraints,
- ▷ Shifting may unfix integers,
- ▷ Integer Shifting finally solves an LP.

Rounding Heuristics

Guideline: Stay feasible!

Features

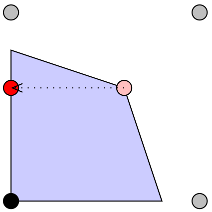
- Simple Rounding always stays feasible,
- Rounding may violate constraints,
- Shifting may unfix integers,
- Integer Shifting finally solves an LP.



Guideline: Stay feasible!

Features

- ▷ Simple Rounding always stays feasible,
- ▷ Rounding may violate constraints,
- ▷ Shifting may unfix integers,
- ▷ Integer Shifting finally solves an LP.

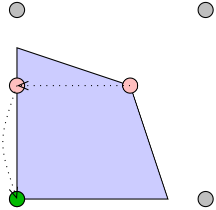


Rounding Heuristics

Guideline: Stay feasible!

Features

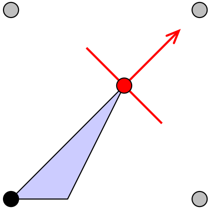
- Simple Rounding always stays feasible,
- Rounding may violate constraints,
- Shifting may unfix integers,
- Integer Shifting finally solves an LP.



Guideline: Stay feasible!

Features

- Simple Rounding always stays feasible,
- Rounding may violate constraints,
- Shifting may unfix integers,
- Integer Shifting finally solves an LP.

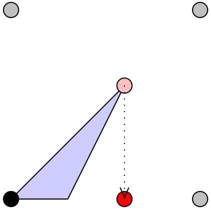


Rounding Heuristics

Guideline: Stay feasible!

Features

- ▷ Simple Rounding always stays feasible,
- ▷ Rounding may violate constraints,
- ▷ Shifting may unfix integers,
- ▷ Integer Shifting finally solves an LP.

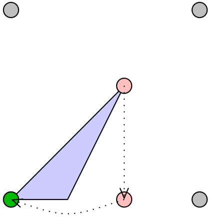


Rounding Heuristics

Guideline: Stay feasible!

Features

- ▷ Simple Rounding always stays feasible,
- ▷ Rounding may violate constraints,
- ▷ Shifting may unfix integers,
- ▷ Integer Shifting finally solves an LP.

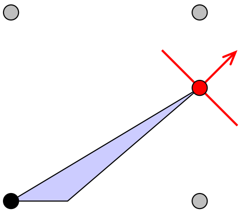


Rounding Heuristics

Guideline: Stay feasible!

Features

- ▷ Simple Rounding always stays feasible,
- ▷ Rounding may violate constraints,
- ▷ Shifting may unfix integers,
- ▷ Integer Shifting finally solves an LP.

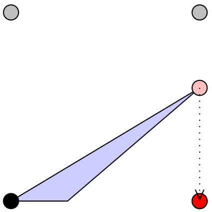


Rounding Heuristics

Guideline: Stay feasible!

Features

- ▷ Simple Rounding always stays feasible,
- ▷ Rounding may violate constraints,
- ▷ Shifting may unfix integers,
- ▷ Integer Shifting finally solves an LP.

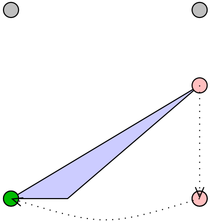


Rounding Heuristics

Guideline: Stay feasible!

Features

- ▷ Simple Rounding always stays feasible,
- ▷ Rounding may violate constraints,
- ▷ Shifting may unfix integers,
- ▷ Integer Shifting finally solves an LP.

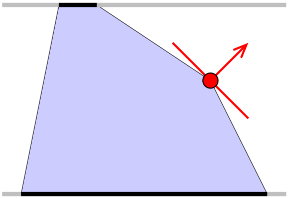


Rounding Heuristics

Guideline: Stay feasible!

Features

- ▷ Simple Rounding always stays feasible,
- ▷ Rounding may violate constraints,
- ▷ Shifting may unfix integers,
- ▷ Integer Shifting finally solves an LP.

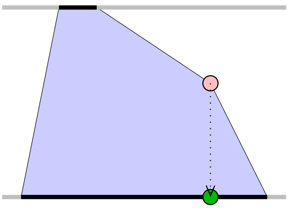


Rounding Heuristics

Guideline: Stay feasible!

Features

- ▷ Simple Rounding always stays feasible,
- ▷ Rounding may violate constraints,
- ▷ Shifting may unfix integers,
- ▷ Integer Shifting finally solves an LP.

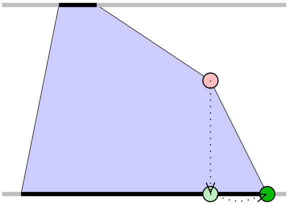


Rounding Heuristics

Guideline: Stay feasible!

Features

- ▷ Simple Rounding always stays feasible,
- ▷ Rounding may violate constraints,
- ▷ Shifting may unfix integers,
- ▷ Integer Shifting finally solves an LP.



Primal Heuristics

Approaches

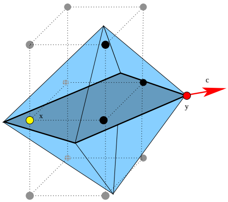
- Rounding: Change fractional to integral values
- Diving: simulate DFS in the branch-and-bound tree using some special branching rule
- Objective diving: manipulate objective function (instead of bounds)
- Large Neighborhood Search: solve some sub-MIP
- Pivoting: manipulate simplex algorithm
- Combinatorial: use special polyhedral properties

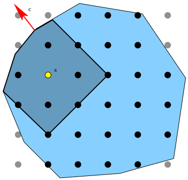
Large Neighborhood Search

LNS improvement heuristics

- ✓ RINS
- ✓ Local Branching
- ✓ Mutation
- ✓ Crossover

Today: LNS Start Heuristic





RENS (Relaxation Enforced Neighborhood Search)

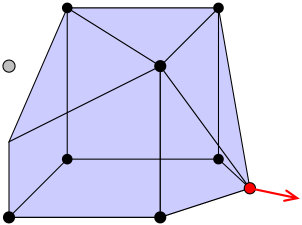
Algorithm

1. $\bar{x} \leftarrow$ LP optimum;
2. Fix all integral variables:
   $x_i := \bar{x}_i$ for all $i : \bar{x}_i \in \mathbb{Z}$;
3. Reduce domain of fractional variables:
   $x_i \in \{\lfloor \bar{x}_i \rfloor; \lceil \bar{x}_i \rceil\}$;
4. Solve the resulting sub-MIP

RENS (Relaxation Enforced Neighborhood Search)

Algorithm

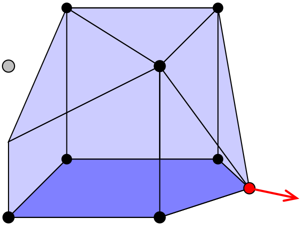
1. $\bar{x} \leftarrow$ LP optimum;
2. Fix all integral variables:
   $x_i := \bar{x}_i$ for all $i : \bar{x}_i \in \mathbb{Z}$;
3. Reduce domain of fractional variables:
   $x_i \in \{\lfloor \bar{x}_i \rfloor ; \lceil \bar{x}_i \rceil\}$;
4. Solve the resulting sub-MIP



RENS (Relaxation Enforced Neighborhood Search)

Algorithm

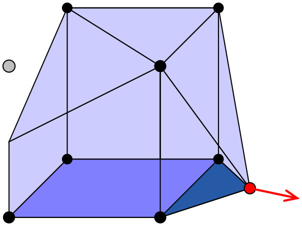
1. $\bar{x} \leftarrow$ LP optimum;
2. Fix all integral variables:
   $x_i := \bar{x}_i \quad$ for all $i : \bar{x}_i \in \mathbb{Z}$;
3. Reduce domain of fractional variables:
   $x_i \in \{\lfloor \bar{x}_i \rfloor; \lceil \bar{x}_i \rceil\}$;
4. Solve the resulting sub-MIP



RENS (Relaxation Enforced Neighborhood Search)

Algorithm

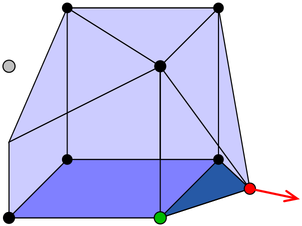
1. $\bar{x} \leftarrow$ LP optimum;
2. Fix all integral variables:
   $x_i := \bar{x}_i \quad$ for all $i : \bar{x}_i \in \mathbb{Z}$;
3. Reduce domain of fractional variables:
   $x_i \in \{\lfloor \bar{x}_i \rfloor ; \lceil \bar{x}_i \rceil\}$;
4. Solve the resulting sub-MIP



RENS (Relaxation Enforced Neighborhood Search)

Algorithm

1. $\bar{x} \leftarrow$ LP optimum;
2. Fix all integral variables:
   $x_i := \bar{x}_i \quad$ for all $i : \bar{x}_i \in \mathbb{Z}$;
3. Reduce domain of fractional variables:
   $x_i \in \{\lfloor \bar{x}_i \rfloor ; \lceil \bar{x}_i \rceil\}$;
4. Solve the resulting sub-MIP



RENS (Relaxation Enforced Neighborhood Search)

Observations

- Solutions found by RENS are roundings of $\bar{x}$
- Yields best possible rounding
- Yields certificate, if no rounding exists

Results

RENS (Relaxation Enforced Neighborhood Search)

Observations

- Solutions found by RENS are roundings of $\bar{x}$
- Yields best possible rounding
- Yields certificate, if no rounding exists

Results

- 82 of 129 test instances are roundable
- RENS finds a global optimum for 23 instances!
- Dominates all other rounding heuristics

Primal Heuristics

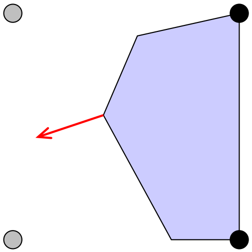
Approaches

- ▷ Rounding: Change fractional to integral values
- ▷ Diving: simulate DFS in the branch-and-bound tree using some special branching rule
- ▷ Objective diving: manipulate objective function (instead of bounds)
- ▷ Large Neighborhood Search: solve some sub-MIP
- ▷ Pivoting: manipulate simplex algorithm
- ▷ Combinatorial: use special polyhedral properties

The Feasibility Pump (Fischetti, Lodi et al.)

Algorithm

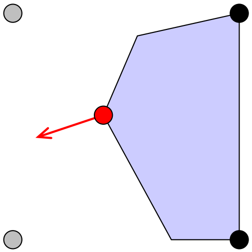
1. Solve LP;
2. Round LP optimum;
3. If feasible:
4. Stop!
5. Else:
6. Change objective;
7. Go to 1;



The Feasibility Pump (Fischetti, Lodi et al.)

Algorithm

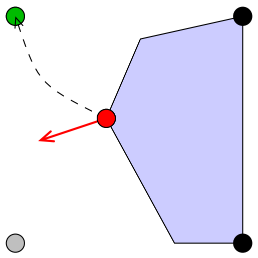
1. Solve LP;
2. Round LP optimum;
3. If feasible:
4. Stop!
5. Else:
6. Change objective;
7. Go to 1;



The Feasibility Pump (Fischetti, Lodi et al.)

Algorithm

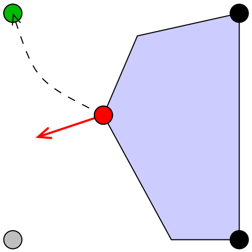
1. Solve LP;
2. Round LP optimum;
3. If feasible:
4. Stop!
5. Else:
6. Change objective;
7. Go to 1;



The Feasibility Pump (Fischetti, Lodi et al.)

Algorithm

1. Solve LP;
2. Round LP optimum;
3. If feasible:
4.   Stop!
5. Else:
6.   Change objective;
7.   Go to 1;

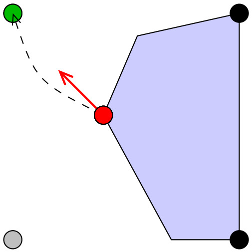


The Feasibility Pump (Fischetti, Lodi et al.)

Algorithm

1. Solve LP;
2. Round LP optimum;
3. If feasible:
4. Stop!
5. Else:
6. Change objective;
7. Go to 1;

$\Delta(x,\tilde{x}) = \sum |x_j - \tilde{x}_j|$

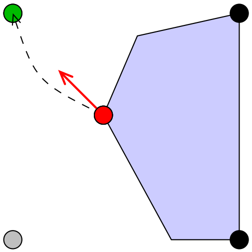


The Feasibility Pump (Fischetti, Lodi et al.)

Algorithm

1. Solve LP;
2. Round LP optimum;
3. If feasible:
4.   Stop!
5. Else:
6.   Change objective;
7.   Go to 1;

$\Delta(x,\tilde{x}) = \sum |x_j - \tilde{x}_j|$

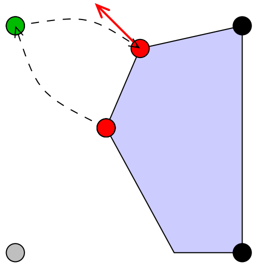


The Feasibility Pump (Fischetti, Lodi et al.)

Algorithm

1. Solve LP;
2. Round LP optimum;
3. If feasible:
4. Stop!
5. Else:
6. Change objective;
7. Go to 1;

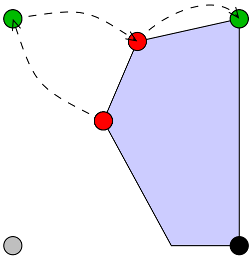
$\Delta(x,\tilde{x}) = \sum |x_j - \tilde{x}_j|$



The Feasibility Pump (Fischetti, Lodi et al.)

Algorithm

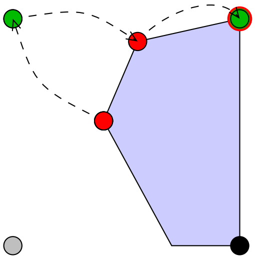
1. Solve LP;
2. Round LP optimum;
3. If feasible:
4. Stop!
5. Else:
6. Change objective;
7. Go to 1;



The Feasibility Pump (Fischetti, Lodi et al.)

Algorithm

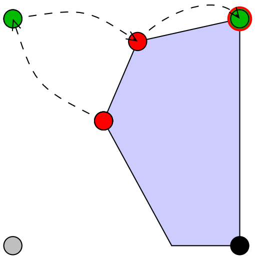
1. Solve LP;
2. Round LP optimum;
3. If feasible:
4. Stop!
5. Else:
6. Change objective;
7. Go to 1;



The Feasibility Pump (Fischetti, Lodi et al.)

Algorithm

1. Solve LP;
2. Round LP optimum;
3. If feasible:
4. Stop!
5. Else:
6. Change objective;
7. Go to 1;



Objective Feasibility Pump (Achterberg & B.)

Improvements

- Objective $c^T x$ regarded at each step:
  $\tilde{\Delta} := (1-\alpha)\Delta(x) + \alpha c^T x$, with $\alpha \in [0,1]$
- Algorithm able to recover from cycling
- Quality of solutions much better

Results

Objective Feasibility Pump (Achterberg & B.)

Improvements

- Objective $c^T x$ regarded at each step:
  $\tilde{\Delta} := (1-\alpha)\Delta(x) + \alpha c^T x$, with $\alpha \in [0,1]$
- Algorithm able to recover from cycling
- Quality of solutions much better

Results

- Finds a solution for 74% of the test instances
- $\approx$ 20% more running time
- Optimality gap decreased from 107% to 38%

(Objective) Feasibility Pump 2.0 (Fischetti & Salvagnin)

Feasibility Pump 2.0

- applies propagation after each rounding
- uses specific propagators for special linear constraints
- fewer rounding steps, "more feasible"

Results

(Objective) Feasibility Pump 2.0 (Fischetti & Salvagnin)

Feasibility Pump 2.0

- applies propagation after each rounding
- uses specific propagators for special linear constraints
- fewer rounding steps, "more feasible"

Results

- fewer ($\approx 50\%$) iterations, hence
- better quality, higher success rate
- benefits combine with Objective FP improvements

SCIP primal heuristics

SCIP primal heuristics

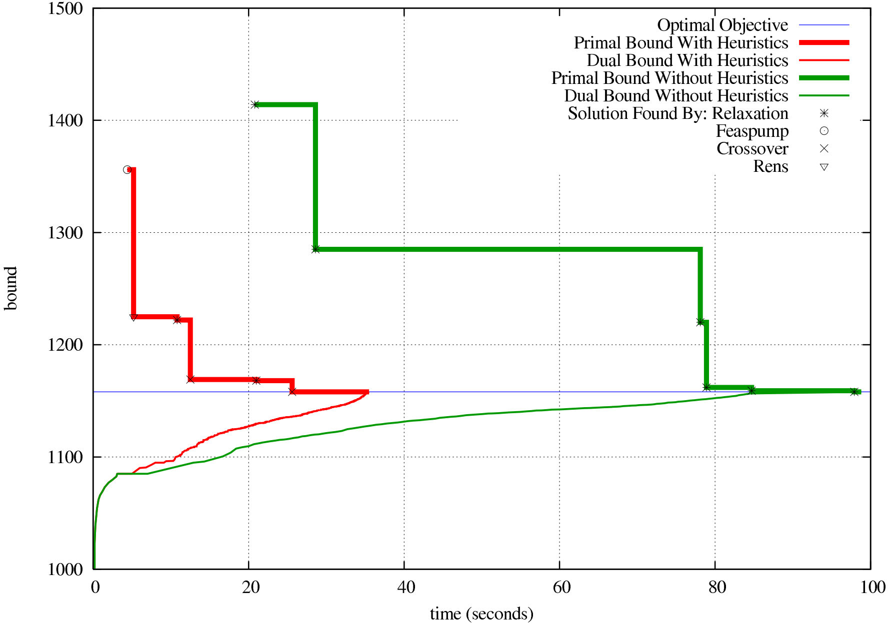
By approach

- 8 Diving heuristics
- 6 LNS heuristics
- 4 Rounding heuristics
- 4 Combinatorial / Others
- 3 Objective divers
- 2 Problem specific

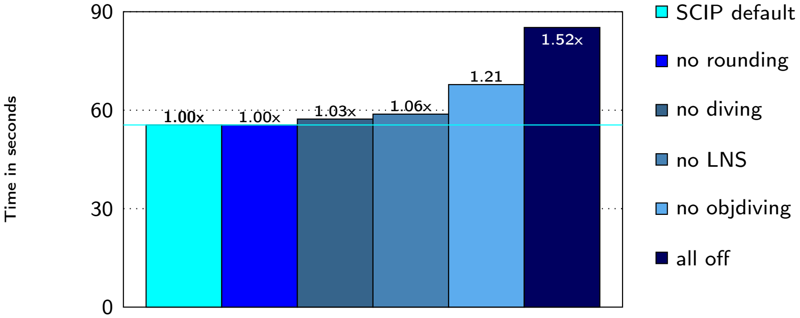
By category

- 25 start heuristics
- 6 improvement heuristics
- 2 repair heuristics

An Example


Optimal Objective
Primal Bound With Heuristics
Dual Bound With Heuristics
Primal Bound Without Heuristics
Dual Bound Without Heuristics
Solution Found By: Relaxation
Feaspump
Crossover
Rens
bound
time (seconds)
1000
1100
1200
1300
1400
1500
0
20
40
60
80
100

Impact of Different Heuristics



- Results for SCIP version 1.1
- Default settings $\leftrightarrow$ disabling classes of heuristics
- 35 Instances from MIPLIB2003 which SCIP solves within 1 hour

Results & Conclusion

Single Heuristics

- ▷ Deactivating a single heuristic yields 1%-6% degradation
- ▷ No heuristic clearly dominating (best one: objective feaspump)
- ▷ Coordination important

Overall Effect

- ▷ Better pruning, earlier fixing
- ▷ 7% less instances without any solution
- ▷ 5% more instances solved within one hour
- ▷ Only half of the branch-and-bound-nodes
- ▷ Only 70% of the solving time

Cutting Planes and Primal Heuristics

CO@Work Berlin

Timo Berthold and Kati Wolter

09/30/2009

Berlin Mathematical School

DFG Research Center MATHEON
Mathematics for key technologies

Once again,. . .

- Heuristically build groups of 3 people (e.g., nearest-neighbor)
  - laptop owner
  - C expert
  - IP expert
- Open your mouth, if you get stuck. ;-)