Cutting Planes and Primal Heuristics
CO@Work Berlin

Timo Berthold and Kati Wolter

09/30/2009
Cutting Planes and Primal Heuristics

CO@Work Berlin

Timo Berthold and Kati Wolter

09/30/2009
\[ \mathcal{F}_{IP} := \{ x \in \mathbb{Z}_+^n : Ax \leq b \} \]

\[ \mathcal{F}_{LP} := \{ x \in \mathbb{R}_+^n : Ax \leq b \} \]
Observation

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

Problems with $\text{conv}(\mathcal{F}_{IP})$:
- linear description not known
- large nr. of constraints needed
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

\[
\min \{ c^T x : x \in \text{conv}(\mathcal{F}_{\text{IP}}) \}
\]
General Cutting Plane Method

Observation

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

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

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

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

\[
\mathcal{F}_{LP} \supseteq \mathcal{F} \supseteq \text{conv}(\mathcal{F}_{IP})
\]

\[
\min \{ c^T x : x \in \mathcal{F}_{LP} \} \leq \min \{ c^T x : x \in \mathcal{F} \} = \min \{ c^T x : x \in \text{conv}(\mathcal{F}_{IP}) \}
\]
Algorithm

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

Algorithm

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

Algorithm

1. $F \leftarrow F_{LP}$

2. Solve

\[
\min c^T x \\
\text{s.t.} \quad x \in F
\]

3. If $x^* \in F_{IP}$: Stop

4. Add inequality to $F$ that is ...
   - valid for $\text{conv}(F_{IP})$ but
   - violated by $x^*$.

5. Goto 2.
General Cutting Plane Method

Algorithm

1. $\mathcal{F} \leftarrow \mathcal{F}_{LP}$

2. Solve

   $\min \ c^T x$
   s.t. $x \in \mathcal{F}$

3. If $x^* \in \mathcal{F}_{IP}$: Stop

4. Add inequality to $\mathcal{F}$ that is ...
   - valid for $\text{conv}(\mathcal{F}_{IP})$ but
   - violated by $x^*$.

5. Goto 2.
Algorithm

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

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

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

2. Solve

   \[
   \text{min } c^T x \\
   \text{s.t. } x \in \mathcal{F}
   \]

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.
General Cutting Plane Method

Algorithm

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

1. \( \mathcal{F} \leftarrow \mathcal{F}_{\text{LP}} \)

2. Solve
   \[
   \min c^T x \\
   \text{s.t. } x \in \mathcal{F}
   \]

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.
Algorithm

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

2. Solve

$$\begin{align*}
\min & \quad c^T x \\
\text{s.t.} & \quad x \in \mathcal{F}
\end{align*}$$

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.
General Cutting Plane Method

Algorithm

1. $\mathcal{F} \leftarrow \mathcal{F}_{LP}$

2. Solve

   $$\min c^T x \quad \text{s.t.} \quad x \in \mathcal{F}$$

3. If $x^* \in \mathcal{F}_{IP}$: Stop

4. Add inequality to $\mathcal{F}$ that is ...
   - valid for $\text{conv}(\mathcal{F}_{IP})$ but
   - violated by $x^*$.

5. Goto 2.
Algorithm

1. \( \mathcal{F} \leftarrow \mathcal{F}_{\text{LP}} \)
2. Solve
   \[
   \min \ c^T x \\
   \text{s.t.} \quad x \in \mathcal{F}
   \]
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.
Algorithm

1. $\mathcal{F} \leftarrow \mathcal{F}_{LP}$

2. Solve

   $$\min \ c^T x$$

   s.t. \quad x \in \mathcal{F}

3. If $x^* \in \mathcal{F}_{IP}$: Stop

4. Add inequality to $\mathcal{F}$ that is ...
   - valid for $\text{conv}(\mathcal{F}_{IP})$ but
   - violated by $x^*$.

5. Goto 2.
General Cutting Plane Method

Algorithm

1. \( \mathcal{F} \leftarrow \mathcal{F}_{LP} \)

2. Solve

\[
\min c^T x \\
\text{s.t. } x \in \mathcal{F}
\]

3. If \( x^* \in \mathcal{F}_{IP} \): Stop

4. Add inequality to \( \mathcal{F} \) that is ... ▶ valid for \( \text{conv}(\mathcal{F}_{IP}) \) but ▶ violated by \( x^* \).

5. Goto 2.
Algorithm

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

2. Solve
   $$\min \ c^T x \quad \text{s.t.} \quad x \in \mathcal{F}$$

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.
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
**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: \(C \subseteq N\)

\[
\begin{align*}
\sum_{j \in C} a_j &> b \\
\sum_{j \in C \setminus \{i\}} a_j &\leq b \ \forall \ i \in C
\end{align*}
\]

Minimal Cover Inequality

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

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

Minimal cover inequality:
\(x_2 + x_3 + x_4 \leq 2\)
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

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

$C = \{2, 3, 4\}$

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

(I) $\sum_{j \in C} x_j + \alpha_1 x_1 \leq 2$ valid for $X^{BK}$
Sequential Up-lifting

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

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

$$(I) \sum_{j \in C} x_j + \alpha_1 x_1 \leq 2 \quad \text{valid for } X^{BK}$$

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

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

\[
\sum_{j \in C} x_j \leq 2 \quad \text{valid for } X^{BK} \cap \{ x \in \{0, 1\}^4 : x_1 = 0 \}
\]

\[
(I) \quad \sum_{j \in C} x_j + \alpha_1 x_1 \leq 2 \quad \text{valid for } X^{BK}
\]

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

\[
\Leftrightarrow \sum_{j \in C} x_j + \alpha_1 \cdot 0 \leq 2 \quad \text{is valid for } X^{BK} \cap \{ x \in \{0, 1\}^4 : x_1 = 0 \}
\]
Sequential Up-lifting

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

▷ $C = \{2, 3, 4\}$

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

(l) $$\sum_{j \in C} x_j + \alpha_1 x_1 \leq 2 \quad \text{valid for } X^{BK}$$

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

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

$$\Leftrightarrow \alpha_1 \in [-\infty, \infty]$$
Sequential Up-lifting

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

\[
\sum_{j \in C} x_j \leq 2 \quad \text{valid for } X^{BK} \cap \{x \in \{0, 1\}^4 : x_1 = 0\},
\]

(I) \[
\sum_{j \in C} x_j + \alpha_1 x_1 \leq 2 \quad \text{valid for } X^{BK}
\]

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

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

$C = \{2, 3, 4\}$

\[
\sum_{j \in C} x_j \leq 2 \quad \text{valid for } X^{BK} \cap \{ x \in \{0, 1\}^4 : x_1 = 0 \}
\]

(I) \[
\sum_{j \in C} x_j + \alpha_1 x_1 \leq 2 \quad \text{valid for } X^{BK}
\]

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

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

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

$\setminus C = \{2, 3, 4\}$

$$\sum_{j \in C} x_j \leq 2 \quad \text{valid for } X^B \cap \{x \in \{0, 1\}^4 : x_1 = 0\}$$

$$(l) \quad \sum_{j \in C} x_j + \alpha_1 x_1 \leq 2 \quad \text{valid for } X^B$$

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

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

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

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

\[ C = \{2, 3, 4\} \]

\[ \sum_{j \in C} x_j \leq 2 \quad \text{valid for } X^{BK} \cap \{ x \in \{0, 1\}^4 : x_1 = 0 \} \]

\( (I) \quad \sum_{j \in C} x_j + \alpha_1 x_1 \leq 2 \quad \text{valid for } X^{BK} \)

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

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

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

\[ \Rightarrow 1 + \alpha_1 \leq 2 \Rightarrow \alpha_1 \leq 1 \]
Sequential Up-lifting

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

\[\sum_{j \in C} x_j \leq 2 \quad \text{valid for } X^{BK} \cap \{x \in \{0, 1\}^4 : x_1 = 0\}\]

(l) \[\sum_{j \in C} x_j + \alpha_1 x_1 \leq 2 \quad \text{valid for } X^{BK}\]

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

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

**Result:** Inequ. (l) valid for $X^{BK}$ for all $\alpha_1 \leq 1$
Sequential Up-lifting

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

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

Different lifting sequences may lead to different inequalities!

Use sequential up- and down-lifting!
Sequential Up-lifting

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

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

- 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

$$\text{conv}( X^{BK} \cap \{x \in \{0,1\}^n : x_j = 0 \text{ for all } j \in N \setminus C, \\
\hspace{3cm} x_j = 1 \text{ for all } j \in C_2\} ).$$

- **Up-lifting:** variables in $N \setminus C$
- **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 $G = (V, E)$:

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

Stable Set: $S \subseteq V$

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

Stable set
Cuts for the Stable Set Problem

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

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

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

1, 2
3, 4
5, 6
7, 8
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: $C \subseteq V$**

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

$\Rightarrow$ Clique inequality: $\sum_{j \in C} x_j \leq 1$ is valid for stable set polytope.
Conflict Graph: $G = (V, E)$

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

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

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

$\rightarrow$ 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:

\[ X := \{ (x, s) \in \mathbb{Z} \times \mathbb{R} : x \leq b + s \quad (I) \]
\[ s \geq 0 \quad (II) \} \]
Elementary mixed integer set:

\[ X := \{ (x, s) \in \mathbb{Z} \times \mathbb{R} : x \leq b + s \ (I) \ s \geq 0 \ (II) \} \]

Inequalities (I) and (II) do not suffice to describe \( \text{conv}(X) \).
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)
Inequality valid for $X^1$ and for $X^2$:

\[ x \leq \lfloor b \rfloor + \frac{s}{1-f_b} \]

(\text{I}) + f_b(\text{III}) \quad \text{and} \quad (\text{II}) + (1-f_b)(\text{IV})
Mixed Integer Rounding (MIR) Cut

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

$$x \leq \lfloor b \rfloor + \frac{s}{1 - f_b}$$

MIR inequality

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

$$x \leq \lfloor b \rfloor + \frac{s}{1 - f_b}$$

MIR inequality
Complemented MIR (C-MIR) Cut

Mixed knapsack set:

\[
\{ (x, s) \in \mathbb{Z}_+^n \times \mathbb{R}_+ : \\
\sum_{j \in N} a_j x_j \leq b + s \\
x_j \leq u_j \quad j \in N \}
\]
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

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

\[\Rightarrow\]

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

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

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

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

Bounds: \(x_1, x_2 \leq 2\)
Example

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

\[ \sum_{j \in N} b_j x_j \leq \lfloor b \rfloor + \frac{s}{1 - f_b} \]

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

Bounds: \( x_1, x_2 \leq 2 \)

For \( \delta = 1 \):
Example

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

$$\Rightarrow$$

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

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

Bounds: $$x_1, x_2 \leq 2$$

For $$\delta = 1$$:

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

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

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

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

Bounds: \(x_1, x_2 \leq 2\)

For \(\delta = 1\):

\[ f_{\frac{11}{2}} = \frac{11}{2} - \lfloor \frac{11}{2} \rfloor = \frac{1}{2} \]
Example

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

\[ \sum_{j \in N} F_{f_b}(a_j) x_j \leq \lceil b \rceil + \frac{s}{1 - f_b} \]

1x₁ + 4x₂ ≤ \frac{11}{2} + s

Bounds: x₁, x₂ ≤ 2

For δ = 1:

\[ f_{\frac{11}{2}} = \frac{11}{2} - \left\lfloor \frac{11}{2} \right\rfloor = \frac{1}{2} \]
Example

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

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

Bounds: \(x_1, x_2 \leq 2\)

\[
1x_1 + 4x_2 \leq 5 + 2s
\]

For \(\delta = 4\), \(x_1 = 2 - \bar{x}_1\):
Example

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

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

Bounds: \(x_1, x_2 \leq 2\)

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

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

\[ \Rightarrow \]

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

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

Bounds: \(x_1, x_2 \leq 2\)

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

\(\triangleright\) \(- \frac{1}{4} \bar{x}_1 + x_2 \leq \frac{7}{8} + \frac{1}{4} s\)

\(\triangleright\) \(f_{\frac{7}{8}} = \frac{7}{8} - \left\lfloor \frac{7}{8} \right\rfloor = \frac{7}{8}\)
Example

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

$$1x_1 + 4x_2 \leq \frac{11}{2} + s \quad \sim \rightarrow \quad 1x_1 + 4x_2 \leq 5 + 2s$$

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

\begin{itemize}
  \item $-\frac{1}{4} \bar{x}_1 + x_2 \leq \frac{7}{8} + \frac{1}{4} s$
  \item $f_{\frac{7}{8}} = \frac{7}{8} - \lfloor \frac{7}{8} \rfloor = \frac{7}{8}$
\end{itemize}
Example

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

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

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

Bounds: \(x_1, x_2 \leq 2\)

\[ 1x_1 + 4x_2 \leq 5 + 2s \]

\[ -1\bar{x}_1 + 1x_2 \leq 0 + 2s \]

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

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

\[ f_{\frac{7}{8}} = \frac{7}{8} - \lfloor \frac{7}{8} \rfloor = \frac{7}{8} \]
Example

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

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

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

Bounds: \( x_1, x_2 \leq 2 \)

\[ 1x_1 + 4x_2 \leq 5 + 2s \]

\[ 1x_1 + 1x_2 \leq 2 + 2s \]

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

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

\[ f_{\frac{7}{8}} = \frac{7}{8} - \lfloor \frac{7}{8} \rfloor = \frac{7}{8} \]
Outline of the Separation Algorithm


Compl. and scale → Mixed knapsack relax. → Apply MIR cut
Outline of the Separation Algorithm


Compl. and scale → Mixed knapsack relax. → Apply MIR cut
Linear combination of constraints defining the mixed integer set
Outline of the Separation Algorithm

Mixed integer set → Aggregation heur. → Mixed integer relax.


C-MIR cut

Compl. and scale → Mixed knapsack relax. → Apply MIR cut
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

1. Mixed integer set
2. Aggregation heur.
4. Bound subst. heur.
5. Cut generation heur.
6. C-MIR cut
7. Mixed knapsack relax.
8. Compl. and scale
10. Apply MIR cut
Outline of the Separation Algorithm

Mixed integer set → Aggregation heur. → Mixed integer relax.

Bound subst. heur. → Cut generation heur. → C-MIR cut

Mixed knapsack relax. → Mixed knapsack relax. → Compl. and scale → Apply MIR cut

\{ (x, s) \in \mathbb{Z}_+^n \times \mathbb{R}_+ : \sum_{j \in N} a_j x_j \leq b + s \}

\quad x_j \leq u_j \quad j \in N \}
Outline of the Separation Algorithm

- Mixed integer set
  - Aggregation heur.
    - Bound subst. heur.
      - Mixed integer relax.
  - Mixed knapsack relax.
    - Cut generation heur.
      - C-MIR cut

- Compl. and scale
  - Mixed knapsack relax.
  - Apply MIR cut

T. Berthold and K. Wolter (ZIB)
Outline of the Separation Algorithm

Mixed integer set → Aggregation heur. → Mixed integer relax.

Mixed integer relax. → Bound subst. heur. → C-MIR cut

Mixed knapsack relax. → Cut generation heur. → Mixed knapsack relax. → Apply MIR cut

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

1. Mixed integer set
2. Aggregation heur.
4. Bound subst. heur.
5. Cut generation heur.
6. C-MIR cut
7. Compl. and scale
8. Mixed knapsack relax.
9. Apply MIR cut
Outline of the Separation Algorithm


Compl. and scale → Mixed knapsack relax. → Apply MIR cut

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

1. Mixed integer set
2. Aggregation heur.
5. Mixed knapsack relax.
6. Cut generation heur.
7. C-MIR cut
8. Compl. and scale
10. Apply MIR cut
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
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

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

- \((N_1, N_2)\) 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 ($\bar{u} = \infty$, $L_1 = \emptyset$):

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

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

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

\[ y_1 \leq 4x_1 \]
\[ y_2 \leq 4x_2 \]
\[ y_3 \leq 2x_3 \]
\[ y_4 \leq 2x_4 \]
\[ y_5 \leq 6x_5 \]
\[ y_6 \leq 6x_6 \]
\[ y_7 \leq 2x_7 \]
Example

Flow cover:
\[ C_1 = \{1, 2, 3\}, \quad C_2 = \{5\}, \quad \lambda = 2 \]
Example

\begin{align*}
y_1 & \leq 4x_1 \\
y_2 & \leq 4x_2 \\
y_3 & \leq 2x_3 \\
y_4 & \leq 2x_4 \\
y_5 & \leq 6x_5 \\
y_6 & \leq 6x_6 \\
y_7 & \leq 2x_7
\end{align*}

Flow cover:
\[ C_1 = \{1, 2, 3\}, \quad C_2 = \{5\}, \quad \lambda = 2 \]

Sets: \( L_1 = \emptyset, \quad L_2 = \{6\} \)
Example

Flow cover:
\[ C_1 = \{1, 2, 3\}, \quad C_2 = \{5\}, \quad \lambda = 2 \]

Sets: \( L_1 = \emptyset, \quad L_2 = \{6\} \)

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

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

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

\[
\begin{align*}
\sum_{j \in C_1} y_j + (u_j - \lambda)^+ (1 - x_j) \\
- \sum_{j \in C_2} u_j - \min\{\lambda, (u_j - \bar{u} + \lambda)^+\} (1 - x_j) \\
+ \sum_{j \in L_1} y_j - (\max\{\bar{u}, u_j\} - \lambda) x_j \\
- \sum_{j \in L_2} \min\{u_j, \max\{u_j - \bar{u} + \lambda, \lambda\}\} x_j \\
- \sum_{j \in N_2 \setminus (C_2 \cup L_2)} y_j \\
\leq b
\end{align*}
\]
\begin{align*}
y_1 &\leq 4x_1 \\
y_2 &\leq 4x_2 \\
y_3 &\leq 2x_3 \\
y_4 &\leq 2x_4 \\
y_5 &\leq 6x_5 \\
y_6 &\leq 6x_6 \\
y_7 &\leq 2x_7
\end{align*}

\text{Flow cover:}
\begin{align*}
C_1 &= \{1, 2, 3\}, \quad C_2 = \{5\}, \quad \lambda = 2 \\
\text{Sets:} \quad L_1 = \emptyset, \quad L_2 = \{6\}
\end{align*}

\text{SGFCI (} \bar{u} = \infty, \ L_1 = \emptyset \text{):}
\begin{align*}
\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{align*}

\text{EGFCI (} \bar{u} = 4 > 2 \text{):}
\begin{align*}
\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{align*}
Flow cover:
\( C_1 = \{1, 2, 3\}, \ C_2 = \{5\}, \ \lambda = 2 \)

Sets: \( L_1 = \emptyset, \ L_2 = \{6\} \)

SGFCI \((\bar{u} = \infty, \ L_1 = \emptyset)\):
\[
\begin{align*}
y_1 + 2 (1 - x_1) + \\
y_2 + 2 (1 - x_2) + y_3 \\
- \sum_{j \in C_2} u_j \\
- \sum_{j \in L_2} \min\{u_j, \lambda\} x_j \\
- \sum_{j \in N_2 \setminus (C_2 \cup L_2)} y_j \\
\leq b
\end{align*}
\]

EGFCI \((\bar{u} = 4 > 2)\):
\[
\begin{align*}
y_1 + 2 (1 - x_1) + \\
y_2 + 2 (1 - x_2) + y_3 \\
- \sum_{j \in C_2} u_j - \min\{\lambda, (u_j - \bar{u} + \lambda)^+\} (1 - x_j) \\
+ \sum_{j \in L_1} y_j - (\max\{\bar{u}, u_j\} - \lambda) x_j \\
- \sum_{j \in L_2} \min\{u_j, \max\{u_j - \bar{u} + \lambda, \lambda\}\} x_j \\
- \sum_{j \in N_2 \setminus (C_2 \cup L_2)} y_j \\
\leq b
\end{align*}
\]
Example

Flow cover:
\[ C_1 = \{1, 2, 3\}, \ C_2 = \{5\}, \ \lambda = 2 \]

Sets: \( L_1 = \emptyset, \ L_2 = \{6\} \)

SGFCI (\( \bar{u} = \infty, \ L_1 = \emptyset \)): \[
y_1 + 2 (1 - x_1) + \]
\[
y_2 + 2 (1 - x_2) + y_3\]
\[
- 6\]
\[
- \sum_{j \in L_2} \min\{u_j, \lambda\} x_j\]
\[
- \sum_{j \in N_2 \setminus (C_2 \cup L_2)} y_j\]
\[
\leq b\]

EGFCI (\( \bar{u} = 4 > 2 \)): \[
y_1 + 2 (1 - x_1) + \]
\[
y_2 + 2 (1 - x_2) + y_3\]
\[
- 6 + 2 (1 - x_5)\]
\[
+ \sum_{j \in L_1} y_j - (\max\{\bar{u}, u_j\} - \lambda) x_j\]
\[
- \sum_{j \in L_2} \min\{u_j, \max\{u_j - \bar{u} + \lambda, \lambda\}\} x_j\]
\[
- \sum_{j \in N_2 \setminus (C_2 \cup L_2)} y_j\]
\[
\leq b\]

T. Berthold and K. Wolter (ZIB)
Example

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

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

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

\[
y_1 + 2(1 - x_1) + \\
y_2 + 2(1 - x_2) + y_3 \\
- 6 \\
- \sum_{j \in L_2} \min\{u_j, \lambda\} \ x_j \\
- \sum_{j \in N_2 \setminus (C_2 \cup L_2)} y_j \\
\leq b
\]

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

\[
y_1 + 2(1 - x_1) + \\
y_2 + 2(1 - x_2) + y_3 \\
- 6 + 2(1 - x_5) \\
- \sum_{j \in L_2} \min\{u_j, \max\{u_j - \bar{u} + \lambda, \lambda\}\} \ x_j \\
- \sum_{j \in N_2 \setminus (C_2 \cup L_2)} y_j \\
\leq b
\]
Example

Flow cover:
\[ C_1 = \{1, 2, 3\}, \quad C_2 = \{5\}, \quad \lambda = 2 \]

Sets: \( L_1 = \emptyset, \quad L_2 = \{6\} \)

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

\[
y_1 + 2(1 - x_1) + \\
y_2 + 2(1 - x_2) + y_3 \\
- 6 \\
- 2x_6 \\
- \sum_{j \in N_2 \setminus (C_2 \cup L_2)} y_j \\
\leq b
\]

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

\[
y_1 + 2(1 - x_1) + \\
y_2 + 2(1 - x_2) + y_3 \\
- 6 + 2(1 - x_5) \\
- 4x_6 \\
- \sum_{j \in N_2 \setminus (C_2 \cup L_2)} y_j \\
\leq b
\]
Example

Flow cover:
$C_1 = \{1, 2, 3\}, \quad C_2 = \{5\}, \quad \lambda = 2$

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

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

\[
y_1 + 2 (1 - x_1) + y_2 + 2 (1 - x_2) + y_3 - 6 - 2x_6 - y_7 \leq b
\]

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

\[
y_1 + 2 (1 - x_1) + y_2 + 2 (1 - x_2) + y_3 - 6 + 2 (1 - x_5) - 4x_6 - y_7 \leq b
\]
Example

Flow cover:
\[ C_1 = \{1, 2, 3\}, \ C_2 = \{5\}, \ \lambda = 2 \]

Sets: \( L_1 = \emptyset, \ L_2 = \{6\} \)

\[ \text{SGFCI} (\bar{u} = \infty, \ L_1 = \emptyset): \]
\[ y_1 + 2(1 - x_1) + \]
\[ y_2 + 2(1 - x_2) + y_3 \]
\[ - 6 \]
\[ - 2x_6 \]
\[ - y_7 \]
\[ \leq 2 \]

\[ \text{EGFCI} (\bar{u} = 4 > 2): \]
\[ y_1 + 2(1 - x_1) + \]
\[ y_2 + 2(1 - x_2) + y_3 \]
\[ - 6 + 2(1 - x_5) \]
\[ - 4x_6 \]
\[ - y_7 \]
\[ \leq 2 \]
C-MIR Flow Cover Inequality (C-MIRFCI)

Basis:
- Flow cover \((C_1, C_2)\)
- Sets \(L_i \subseteq N_i \setminus C_i\) for \(i = 1, 2\)
- Constant \(\bar{u} > \lambda\)
C-MIR Flow Cover Inequality (C-MIRFCI)

1. Mixed integer set → Aggregation heur.
4. Cut generation heur. → C-MIR cut
5. Mixed knapsack relax. → Compl. and scale
6. Apply MIR cut

T. Berthold and K. Wolter (ZIB)
C-MIR Flow Cover Inequality (C-MIRFCI)


0-1 SNF set

Compl. and scale → Mixed knapsack relax. → Apply MIR cut

T. Berthold and K. Wolter (ZIB)
C-MIR Flow Cover Inequality (C-MIRFCI)


0-1 SNF set

Compl. and scale → Mixed knapsack relax. → Apply MIR cut
Substitute \( y_j = \begin{cases} u_j x_j - \bar{y}_j & : j \in C_1 \cup C_2 \cup L_1 \cup L_2 \\ 0 + \bar{y}_j & : \text{otherwise} \end{cases} \)
Complement all integer vars in $U = C_1 \cup C_2$

Divide constraint by $\delta = \bar{u}$
C-MIR Flow Cover Inequality (C-MIRFCI)


0-1 SNF set

Compl. and scale → Mixed knapsack relax. → Apply MIR cut
C-MIR Flow Cover Inequality (C-MIRFCI)

- Mixed knapsack relax.
- Bound subst. heur.
- Cut generation heur.
- C-MIRFCI

- 0-1 SNF set

- Compl. and scale
- Mixed knapsack relax.
- Apply MIR cut

T. Berthold and K. Wolter (ZIB)
Dominance Relations

Lifted SGFCI → SGFCI

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

\[ L_1 = \emptyset \]

C-MIRFCI

\( (\bar{u} = \max_{j \in C_1 \cup L_2} u_j > \lambda) \)

Lifted FCi → EGFCI

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

C-MIRFCI

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

\[
\max \left\{ \sum_{j \in N_1} (x_j^* - 1)z_j + \sum_{j \in N_2} x_j^* z_j : \right. \\
\left. \sum_{j \in N_1} u_j z_j - \sum_{j \in N_2} u_j z_j > b, \right. \\
\left. z_j \in \{0, 1\} \text{ for all } j \in N \right\}
\]
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:
     - Lifted SGFCI
     - SGFCI \((\bar{u} = \infty, L_1 = \emptyset)\)
     - \(L_1 = \emptyset\)
     - EGFCI \((\bar{u} = \max_{j \in C_1} u_j > \lambda)\)
   - 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
  
- **Orthogonality** with respect to the other cuts
  
- **Parallelism** with respect to the objective function

⇒ **Select** cuts with largest value of

\[
\begin{align*}
& e_r + w_o \cdot o_r + w_p \cdot p_r
\end{align*}
\]
Cut Selection Strategy

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

\[ \Rightarrow \text{Select cuts with largest value of } e_r + w_o o_r + w_p p_r \]

**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
- **Orthogonality** with respect to the other cuts
- **Parallelism** with respect to the objective function

⇒ **Select** cuts with largest value of $e_r + w_o o_r + w_p p_r$

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?

Type of cuts?

- General cuts
  - Separator
    - c-MIR, GMI, ...

- Problem specific cuts
  - Can constraint be expressed by "small" number of existing constraint types?

Can you represent and process constraint in a more efficient way?
Separator or Constraint Handler?

Type of cuts?

- General cuts
  - c-MIR, GMI, ...
- Problem specific cuts
  - Can constraint be expressed by "small" number of existing constraint types?
    - Yes
      - Can you represent and process constraint in a more efficient way?
        - Yes
        - Constraint handler: TSP
    - No
      - Constraint handler: TSP
**Separator or Constraint Handler?**

**Type of cuts?**

- General cuts
  - Separator
    - c-MIR, GMI, ...
- Problem specific cuts
  - Can constraint be expressed by "small" number of existing constraint types?
    - Yes
      - Can you represent and process constraint in a more efficient way?
        - Yes
          - Constraint handler
        - No
          - Separator
    - No
      - Constraint handler

- TSP

- 0-1 KP

T. Berthold and K. Wolter (ZIB) Cuts and Heuristics 09/30/2009 34 / 57
Cutting Planes and **Primal Heuristics**

CO@Work Berlin

Timo Berthold and Kati Wolter

09/30/2009
What are primal heuristics?

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

Primal heuristic
What are primal heuristics?

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!

⇝ Integrate into exact solver
### 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

```c
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

```c
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

```c
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

```c
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

---

[Diagram of a convex hull with points marked.]
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

```c
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

```c
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

<table>
<thead>
<tr>
<th>Two main categories</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ Start heuristics</td>
</tr>
<tr>
<td>▶ Applied early in the search process</td>
</tr>
<tr>
<td>▶ Often at root node</td>
</tr>
<tr>
<td>▶ Mostly start from LP optimum</td>
</tr>
<tr>
<td>▶ Improvement heuristics</td>
</tr>
<tr>
<td>▶ Require feasible solution</td>
</tr>
<tr>
<td>▶ Normally at most once for each incumbent</td>
</tr>
</tbody>
</table>

#define HEUR_FREQQOS 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

```c
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

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

```c
struct SCIP_HeurData
{
    SCIP_SOL* lastsol;
}
```
Heuristic Timings

Domain Propagation

Constraint Enforcement

Primal Heuristics

LP Solving

Solve LP

Pricing

Separation

Domain Propagation
Heuristic Timings

```
#define HEUR_TIMING SCIP_HEURTIMING_AFTERNODE
```

T. Berthold and K. Wolter (ZIB)
Heuristic Timings

Primal Heuristics

Domain Propagation

LP Solving

Solve LP

Pricing

Separation

Domain Propagation

Constraint Enforcement

#define HEUR_TIMING SCIP_HEURTIMING_BeforeNode
Heuristic Timings

Primal Heuristics

Domain Propagation

LP Solving

Solve LP

Pricing

Heurs

Separation

Domain Propagation

Constraint Enforcement

Primal Heuristics

#define HEUR_TIMING SCIP_HEURTIMING_DURINGLPOOP

T. Berthold and K. Wolter (ZIB) Cuts and Heuristics 09/30/2009 40 / 57
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
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.
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.
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.
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.
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.
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.
**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
LNS improvement heuristics

✓ RINS
✓ Local Branching
✓ Mutation
✓ Crossover

Today: LNS Start Heuristic
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
Algorithm

1. \( \bar{x} \leftarrow \) LP optimum;
2. Fix all integral variables:
   \[ x_i := \bar{x}_i \quad \text{for all } i : \bar{x}_i \in \mathbb{Z}; \]
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
Algorithm

1. $\bar{x} \leftarrow \text{LP optimum}$;
2. Fix all integral variables:
   $$x_i := \bar{x}_i \quad \text{for all } i : \bar{x}_i \in \mathbb{Z};$$
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
Algorithm

1. \( \bar{x} \leftarrow \) LP optimum;
2. Fix all integral variables:
   \[ x_i := \bar{x}_i \quad \text{for all } i : \bar{x}_i \in \mathbb{Z}; \]
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
Algorithm

1. $\bar{x} \leftarrow$ LP optimum;
2. Fix all integral variables:
   \[ x_i := \bar{x}_i \quad \text{for all } i : \bar{x}_i \in \mathbb{Z}; \]
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
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
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;
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;
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;

\[ \Delta(x, \tilde{x}) = \sum |x_j - \tilde{x}_j| \]
## Algorithm

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

![Diagram](image-url)
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;

\[ \Delta(x, \tilde{x}) = \sum |x_j - \tilde{x}_j| \]
Objective Feasibility Pump (Achterberg & B.)

Improvements

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

Results
Improvements

- Objective $c^T x$ regarded at each step:
  \[ \tilde{\Delta} := (1 - \alpha)\Delta(x) + \alpha c^T x, \text{ with } \alpha \in [0, 1] \]
- 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%
Feasibility Pump 2.0

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

Results

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

The graph shows the performance of different bound calculations over time. The x-axis represents time in seconds, and the y-axis represents bound values. Various lines and markers indicate different bound calculations:
- **Optimal Objective**: Blue line
- **Primal Bound With Heuristics**: Green line
- **Dual Bound With Heuristics**: Red line
- **Primal Bound Without Heuristics**: Green line
- **Dual Bound Without Heuristics**: Red line

Markers indicate the solution found by:
- **Relaxation**
- **Feaspump**
- **Crossover**
- **Rens**

The graph demonstrates how these bounds converge over time, with the optimal objective reaching a lower bound as time increases.
Impact of Different Heuristics

Results for SCIP version 1.1
Default settings ↔ 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
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. ;-)