University of Paderborn
Faculty of Business and Economics
Department of Information Systems
Decision Support & Operations Research Lab

Working Papers

**WP0704**

# Separation of Mixing Inequalities in a Mixed Integer Programming Solver

P. M. Christophel

Paderborn, Germany
May 2008

# Separation of Mixing Inequalities in a Mixed Integer Programming Solver

Philipp M. Christophel*

DS&OR Lab Workingpaper 0704
revised version as of May 27, 2008

**Abstract**

This paper shows how valid inequalities based on the mixing set can be used in a mixed integer programming (MIP) solver. It discusses the relation of mixing inequalities to flow path and mixed integer rounding inequalities and how currently used separation algorithms already generate cuts implicitly that can be seen as mixing cuts. Further on, it describes two new separation algorithm that generate mixing cuts from mixed integer paths explicitly. A section with computational results discusses the importance of mixing cuts based on paths for solving MIP problems and reports results for the described separation algorithms.

## 1 Introduction

Mixed integer programming (MIP) solvers are software products that try to find proven optimal solutions to MIP problems of the form

$$\begin{aligned} \max\, & cx + hy \\ & Ax + Gy \leq b \\ & x \geq 0, y \geq 0 \text{ and integer.} \end{aligned}$$

They use a collection of algorithmic techniques for modifying the original problem and for finding strong bounds on the optimal solution. The core of all current MIP solvers is an LP relaxation-based branch-and-cut algorithm. An integral part of branch-and-cut algorithms are cutting plane separation algorithms.

The performance of modern MIP solvers heavily depends on these cutting plane separation algorithms (see [4]). State-of-the-art MIP solvers nowadays typically use (lifted) cover cuts, (lifted) clique cuts, implication cuts, Gomory mixed-integer cuts, (lifted) flow cover cuts, aggregated c-MIR cuts and flow path cuts. Recently ILOG CPLEX [10] added $\{0, \frac{1}{2}\}$-cuts [5] and XPress-MP [7] added lift-and-project cuts [2].

*The author spent four months at CORE, Université catholique de Louvain supported by a fellowship from ADONET, a European network in Algorithmic Discrete Optimization, contract nr. MRTN-CT-2003-504438.

As shown by Bixby and Rothberg in [4], aggregated complemented mixed integer rounding (c-MIR) cuts introduced by Marchand and Wolsey in [12] are among the most successful cutting plane separation algorithms used in modern MIP solvers. The approach that leads to mixed integer rounding cuts is to study the simple mixed integer rounding (MIR) set (see [23], [12], [18])

$$X^{MIR} = \{(x, y) \in \mathbb{R}_+ \times \mathbb{Z} : x + y \geq b\}. \tag{1}$$

It is easy to show that for $X^{MIR}$ the *MIR inequality*

$$x \geq f(\lceil b \rceil - y) \tag{2}$$

where $f = b - \lfloor b \rfloor$ is a valid inequality. To obtain valid inequalities for the mixed integer rows of the constraint matrix of MIP problems the MIR approach is to relax these rows to the structure $X^{MIR}$ and then generate MIR inequalities from these relaxations. The difficulty for an aggregated c-MIR cut separation algorithm is to decide on how to construct row relaxations in a way that somehow good cuts are generated. It does this by aggregating rows over paths, substituting bounds, complementing variables and scaling. The exact way in which these steps are performed is of crucial importance for the performance of this algorithm.

In [9] Pochet and Günlük studied an extension of the simple MIR set and called it the mixing set. Section 2 in this paper repeats the results about the mixing set and outlines an approach similar to the approach used for the aggregated c-MIR cut separation algorithm. Sections 3 and 4 show that flow path and aggregated c-MIR cut separation algorithms already implicitly generate cuts that can also be seen as mixing cuts. In section 5 two algorithms are described that can be used to generate mixing cuts explicitly. The paper concludes with computational results in section 6 and final remarks in section 7.

# 2 The Path Mixing Approach

This section repeats the results about the mixing set and introduces path mixing inequalities. The mixing set was initially studied by Günlük and Pochet in [9]. The definition of the mixing set in the notation of [18] is:

**Definition 1** (from [18]). *The mixing set $X_K^{MIX}$ is defined as*

$$X_K^{MIX} = \{(s, y) \in \mathbb{R}_+^1 \times \mathbb{Z}^K : s + y_k \geq b_k \text{ for } 1 \leq k \leq K\}.$$

Günlük and Pochet [9] identified the valid inequalities (3) and (4) for the mixing set, the notation is again taken from [18]. Note that $f_i = b_i - \lfloor b_i \rfloor$.

**Proposition 1** (from [18]). *Let $T \subseteq \{1, \ldots, K\}$ with $|T| = t$, and suppose that $i_1, \ldots, i_t$ is an ordering of $T$ such that $0 = f_{i_0} \leq f_{i_1} \leq \cdots \leq f_{i_t} < 1$. Then the mixing inequalities*

$$s \geq \sum_{\tau=1}^{t} (f_{i_\tau} - f_{i_{\tau-1}})(\lfloor b_{i_\tau} \rfloor + 1 - y_{i_\tau}) \tag{3}$$

*and*

$$s \geq \sum_{\tau=1}^{t} (f_{i_\tau} - f_{i_{\tau-1}})(\lfloor b_{i_\tau} \rfloor + 1 - y_{i_\tau}) + (1 - f_{i_t})(\lfloor b_{i_1} \rfloor - y_{i_1}) \tag{4}$$

*are valid for* $X_K^{MIX}$.

The most violated cut based on inequality (3) or (4) for $X_K^{MIX}$ can be separated efficiently using the following algorithm described in [18]. Reorder the rows $k = 1, \ldots, K$ such that $f_1 \leq \cdots \leq f_K$. For each row define $\beta_i = \lfloor b_j \rfloor + 1 - y_j^*$. Starting with the row which has the largest $\beta_i = \bar{\beta}$, iteratively add the next row $i$ in the ordering as long as $\beta_i > (\bar{\beta} - 1)$ and $\beta_i > 0$. If $\bar{\beta} > 1$, use inequality (4), else use inequality (3). Note that in the context of an MIP solver the most violated cut is not necessarily the best cut. Also note that theorem 8.5 in [18] states, that $s + y_k \geq b_k$ for $1 \leq k \leq K$, $s \geq 0$ and the mixing inequalities (3) and (4) completely describe the convex hull of $X_K^{MIX}$.

Studying the mixing set is very interesting because the mixing inequalities can be used to generalize a number of problem specific valid inequalities for various problem types. These are for example uncapacitated and constant capacity lot sizing, capacitated facility location, capacitated network design and multiple knapsack problems (as shown in [9] and [18]).

Mixing sets usually do not appear in MIP problem instances explicitly. But as in aggregated c-MIR algorithms one can select any set of rows and relax them to form a mixing set. Günlük and Pochet state that selecting these rows is one of the obstacles for using mixing inequalities computationally. The solution to this problem suggested in this paper is to generate rows from mixed integer paths (see definition 2) in order to find valid inequalities for this structure that appears naturally in many MIP instances.

**Definition 2.** *A mixed integer path is described by a set of* $K$ *constraints of the form*

$$\begin{aligned}
\sum_{j=1}^{n} a_{j1} x_j + \sum_{j=1}^{p} g_{j1} y_j &= b_1 \\
\sum_{j=1}^{n} a_{j2} x_j + \sum_{j=1}^{p} g_{j2} y_j &= b_2 \\
&\vdots \\
\sum_{j=1}^{n} a_{jK} x_j + \sum_{j=1}^{p} g_{jK} y_j &= b_K
\end{aligned}$$

*with* $a_{jk} + a_{jk+1} = 0$ *for at least one* $j \in \{1 \ldots n\}$ *and all* $k = 1 \ldots K-1$. *Note that* $x \in \mathbb{R}_+^n, a \in \mathbb{R}^n \times \mathbb{R}^K, g \in \mathbb{R}^p \times \mathbb{R}^K, y \in \mathbb{Z}^p, b \in \mathbb{R}^K$ *and that* $a_{jk}$ *and* $g_{jk}$ *might be* 0 *for some* $(j,k), j = 1 \ldots n, k = 1 \ldots K$.

**Proposition 2.** *Let* $\delta \in \mathbb{R}_+^K$, $\delta > 0$ *and* $T \subseteq \{1 \ldots K\}$, $|T| = t$. *Furthermore*

$$f_i = \frac{\sum_{i=1}^{k} b_i}{\delta_k} - \left\lfloor \frac{\sum_{i=1}^{k} b_i}{\delta_k} \right\rfloor \text{ for } i = 1 \ldots K$$

*and suppose that* $i_1, \ldots, i_t$ *is an ordering of* $T$ *such that* $0 = f_{i_0} \leq f_{i_1} \leq \cdots \leq f_{i_t}$. *Then the* path mixing inequalities

$$\sum_{j=1}^{n} \bar{a}_j x_j + \sum_{j \in I_1} \bar{g}_j y_j \geq \sum_{\tau=1}^{t} \left( f_{i_\tau} - f_{i_{\tau-1}} \right) \left( \lfloor \bar{b}_{i_\tau} \rfloor + 1 - \sum_{i=1}^{i_\tau} \sum_{j \in I_2} \left\lceil \frac{g_{ji}}{\delta_k} \right\rceil y_j \right) \quad (5)$$

*and*

$$\sum_{j=1}^{n} \bar{a}_j x_j + \sum_{j \in I_1} \bar{g}_j y_j \geq \sum_{\tau=1}^{t} \left(f_{i_\tau} - f_{i_{\tau-1}}\right) \left( \lfloor \bar{b}_{i_\tau} \rfloor + 1 - \sum_{i=1}^{i_\tau} \sum_{j \in I_2} \left\lceil \frac{g_{ji}}{\delta_{i_\tau}} \right\rceil y_j \right) \dots$$
$$\dots + (1 - f_{i_t}) \left( \lfloor \bar{b}_{i_1} \rfloor - \sum_{i=1}^{i_1} \sum_{j \in I_2} \left\lceil \frac{g_{ji}}{\delta_{i_1}} \right\rceil y_j \right) \tag{6}$$

*where* $I = 1 \dots p, I = I_1 \cup I_2,\ I_1 \cap I_2 = \emptyset$,

$$\bar{a}_j = \max \left\{ 0, \max_{k \in T} \left\{ \sum_{i=1}^{k} \sum_{j=1}^{n} \frac{a_{ji}}{\delta_k} \right\} \right\} \text{ for } j = 1 \dots n,$$
$$\bar{g}_j = \max \left\{ 0, \max_{k \in T} \left\{ \sum_{i=1}^{k} \sum_{j \in I_1} \frac{g_{ji}}{\delta_k} \right\} \right\} \text{ for } j \in I_1.$$

*and*

$$\bar{b}_k = \frac{\sum_{r=1}^{k} b_r}{\delta_k} \text{ for } k = 1 \dots K$$

*are valid inequalities for a mixed integer path (see definition 2).*

*Proof.* We sum the first $k$ rows of the mixed integer path to get an aggregated path of the form

$$\sum_{i=1}^{k} \sum_{j=1}^{n} a_{ji} x_j + \sum_{i=1}^{k} \sum_{j=1}^{p} g_{ji} y_j = \sum_{i=1}^{k} b_i \text{ for } k \in \{i_1, \dots, i_t\}.$$

Now we divide each row of the aggregated path by a value $\delta_k$ and split the integer variables $y_j, j \in I$ into two sets $I_1$ and $I_2$. The rows of the aggregated path can now be relaxed to form a mixing set

$$\sum_{j=1}^{n} \bar{a}_j x_j + \sum_{j \in I_1} \bar{g}_j y_j + \sum_{i=1}^{k} \sum_{j \in I_2} \left\lceil \frac{g_{ji}}{\delta_k} \right\rceil y_j \geq \sum_{i=1}^{k} \frac{b_i}{\delta_k} \text{ for } k \in \{i_1, \dots, i_t\}$$

because

$$\sum_{j=1}^{n} \bar{a}_j x_j + \sum_{j \in I_1} \bar{g}_j y_j \geq 0$$
$$\sum_{j=1}^{n} \bar{a}_j x_j + \sum_{j \in I_1} \bar{g}_j y_j \geq \sum_{i=1}^{k} \sum_{j=1}^{n} a_{ji} x_j + \sum_{i=1}^{k} \sum_{j=1}^{p} g_{ji} y_j \text{ for } k \in T$$

and

$$\sum_{i=1}^{k} \sum_{j \in I_2} \left\lceil \frac{g_{ji}}{\delta_k} \right\rceil y_j \in \mathbb{Z} \text{ for } k = 1 \dots K.$$

Applying the mixing inequalities (3) and (4) yields the path mixing inequalities. □

We now formalize the approach used to obtain the path mixing inequalities and call it the path mixing approach. It is a formalization and generalization of steps already used in other publications (i.e. [9],[18]).

Step 1. Find a set of base rows and modify them by scaling, substituting bounds, complementing variables and introducing slack variables so that they form a mixed integer path (see definition 2).

Step 2. Aggregate the rows of the mixed integer path.

Step 3. Choose scaling values $\delta_i \in \mathbb{R}_+$,$\delta_i > 0$ for $i = 1 \dots K$.

Step 4. Select a subset $T$ of the aggregated rows.

Step 5. Decide on the sets $I_1$ and $I_2$.

Step 6. Relax the aggregated rows in $T$ to form a mixing set.

Step 7. Generate one of the two mixing inequalities.

The strength of the path mixing procedure lies in the fact that it implicitly generates a number of problem specific cutting planes for several types of lot-sizing problems (see example 2.1 and [18]). In section 3 we show that it can also be used to generalize flow path cuts and in section 5 we present two separation algorithms based on the path mixing approach.

**Example 2.1.** *Consider an instance of the constant capacity lot-sizing problem (called LS-CC in [18])*

$$\begin{aligned}
&\min \sum_{t=1}^{n} p_t x_t + \sum_{t=0}^{n} h_t s_t + \sum_{t=1}^{n} q_t y_t \\
&s_{t-1} + x_t = d_t + s_t && \text{for } 1 \le t \le n \\
&x_t \le C y_t && \text{for } 1 \le t \le n \\
&s \in \mathbb{R}_+^{n+1}, x \in \mathbb{R}_+^n, y \in \{0,1\}^n
\end{aligned}$$

*with* $n = 4$, $(p, h, q) = \{2, 3, 2, 1, 100, 1, 1, 1, 1, 80, 80, 80, 80\}$, $d = \{2, 6, 4, 5\}$ *and* $C = 10$. *The convex hull of the valid integer solutions of this small problem can be computed using the software PORTA [6]. See figure 1 and 2 in the appendix for the PORTA input and the facets of the convex hull. Many of these facets can be obtained using the mixing inequalities. They can be generated by using the path formed by the flow balance constraints and replacing the variable upper bounds on the production variables. The aggregated path rows for this example (with all variable bounds replaced) are:*

$$\begin{aligned}
s_0 + 10y_1 - s_1 &= 2 \\
s_0 + 10y_1 + 10y_2 - s_2 &= 8 \\
s_0 + 10y_1 + 10y_2 + 10y_3 - s_3 &= 12 \\
s_0 + 10y_1 + 10y_2 + 10y_3 + 10y_4 - s_4 &= 17
\end{aligned}$$

*To generate one of the facets using inequalities (3) or (4) we first divide all rows by* $\delta = C = 10$. *From the resulting inequalities we choose a set* $T$ *and generate*

*either (3) or (4). For a nicer representation we can then multiply the generated inequality by $\delta$. The following inequality is facet (27) of this example and can be generated using $T = \{1, 4\}$,$\delta = C = 10$ and the mixing inequality (3).*

$$s_0 \geq 12 - 7y_1 - 5y_2 - 5y_3 - 5y_4$$

# 3 Flow Path Cuts

The idea for flow path cuts origins in [19] and a separation algorithm for them was implemented in [21]. Flow path cuts are based on valid inequalities for fixed charge paths.

**Definition 3** (Structure $S_3$ from [21])**.** *A fixed charge path is described by the constraints*

$$\begin{aligned}
-x_1 + \sum_{j \in N_i^+} x_j - \sum_{j \in N_1^-} x_j &\leq b_1 \\
+x_{k-1} - x_k + \sum_{j \in N_k^+} x_j - \sum_{j \in N_k^-} x_j &\leq b_k \text{ for } k = 2, \ldots, K-1 \\
+x_{K-1} \qquad + \sum_{j \in N_K^+} x_j - \sum_{j \in N_K^-} x_j &\leq b_K \\
x_k &\geq 0 \text{ for } k = 1, \ldots, K-1 \\
l_j y_j \leq x_j &\leq u_j yj \\
y_j \in \{0,1\} \quad j &\in \bigcup_{k=1}^{K} (N_k^+ \cup N_k^-).
\end{aligned}$$

Note that mixed integer paths (see definition 2) can be relaxed to form fixed charge paths by reformulating each row of the path as shown in [20]. Van Roy and Wolsey proposed two families of valid inequalities for fixed charge paths. In the two following propositions where taken from [21] but in compliance with [19] we changed $b_t$ to $b_t^+$ where $b_t^+ = \max\{0, b_t\}$.

**Proposition 3** (similar to [21])**.** *The* simple network inequalities

$$\sum_{k=1}^{K} \sum_{j \in C_k^+} x_j \leq \sum_{k=1}^{K} \sum_{j \in C_k^+} \left( \sum_{t=k}^{K} b_t^+ \right) y_j + \sum_{k=1}^{K} \sum_{j \in N_k^-} x_j \tag{7}$$

*where $C_k^+ \subseteq N_k^+$ for $k = 1 \ldots K$ are valid for a fixed charge path $S_3$.*

**Proposition 4** (similar to [21])**.** *The* extended network inequalities

$$\sum_{k=1}^{K} \sum_{j \in C_k^+} x_j \leq \sum_{k=1}^{K} \sum_{j \in C_k^+} \left( \sum_{t=k}^{K} b_t^+ + \sum_{i \in Q_t^-} u_t \right) y_j + \sum_{k=1}^{K} \sum_{j \in N_k^- \setminus Q_k^-} x_j \tag{8}$$

*where $C_k^+ \subseteq N_k^+$ and $Q_k^- \subseteq N_k^-$ for $k = 1 \ldots K$ are valid for a fixed charge path $S_3$.*

A flow path cut separation algorithm is implemented in many state-of-the-art MIP solvers despite the fact, that they only work well on certain instances. These instances are usually lot-sizing or network design instances because fixed charge paths appear in them naturally. In fact, the facet defining valid inequalities known for uncapacitated lot-sizing can be generated as flow path cuts. As these can also be generated using mixing inequalities, it is not surprising that proposition 5 holds.

**Proposition 5.** *Simple and extended network inequalities can be generated using the path mixing approach (see section 2).*

*Proof.* The first step is to relax the rows by replacing $b_k$ by $b_k^+$. Then we generate a set of rows by aggregating the first $k$ rows of a fixed charge path for $k = 1 \dots K$ to get an aggregated fixed charge path (9).

$$\sum_{i=1}^{k} \sum_{j \in N_i^+} x_j - \sum_{i=1}^{k} \sum_{j \in N_i^-} x_j \leq \sum_{i=1}^{k} b_i^+ \text{ for } k = 1 \dots K \tag{9}$$

For each row in (9) we add a slack variable $t_k$ to reformulate the inequality to an equality. Then we relax the equalities by dropping the variables in $N_k^-$ and by substituting the variable bounds $u_j y_j$ for all $j \in C_k^+$ for all $k = 1 \dots K$. The resulting inequalities are divided by $\delta \in \mathbb{R}_+$ where $\delta = M$ is a very large number.

$$\sum_{i=1}^{k} \sum_{j \in N_i^+ \setminus C_i^+} \frac{x_j}{\delta} + \frac{t_k}{\delta} + \sum_{i=1}^{k} \sum_{j \in C_i^+} \frac{u_j}{\delta} y_j \geq \frac{\sum_{i=1}^{k} b_i^+}{\delta} \text{ for } k = 1 \dots K. \tag{10}$$

The rows (10) can now be relaxed to form the base rows of a mixing set (11).

$$\sum_{i=1}^{K} \sum_{j \in N_i^+ \setminus C_i^+} \frac{x_j}{\delta} + \frac{t_K}{\delta} + \sum_{i=1}^{k} \sum_{j \in C_i^+} y_j \geq \frac{\sum_{i=1}^{k} b_i^+}{\delta} \text{ for } k = 1 \dots K. \tag{11}$$

because

$$\sum_{i=1}^{K} \sum_{j \in N_i^+ \setminus C_i^+} \frac{x_j}{\delta} + \frac{t_K}{\delta} \geq \sum_{i=1}^{k} \sum_{j \in N_i^+ \setminus C_i^+} \frac{x_j}{\delta} + \frac{t_k}{\delta} \text{ for } k = 1 \dots K$$

and

$$1 = \left\lceil \frac{u_j}{\delta} \right\rceil \geq \frac{u_j}{\delta}.$$

These base rows are already ordered according to $f_k = \frac{\sum_{i=1}^{k} b_i^+}{\delta}$ for $k = 1 \dots K$. By using (3) with $T = \{1 \dots K\}$ and because in this case $f_k - f_{k-1} = \frac{b_k^+}{\delta}$ we get

$$\sum_{i=1}^{K} \sum_{j \in N_i^+ \setminus C_i^+} \frac{x_j}{\delta} + \frac{t_K}{\delta} \geq \sum_{k=1}^{K} \frac{b_k^+}{\delta} - \sum_{k=1}^{K} \frac{b_k^+}{\delta} \sum_{i=1}^{k} \sum_{j \in C_i^+} y_j$$

substituting

$$t_K = \sum_{i=1}^{K} b_i^+ - \sum_{i=1}^{K} \sum_{j \in N_i^+} x_j + \sum_{i=1}^{K} \sum_{j \in N_i^-} x_j$$

and multiplying with $-\delta$ yields the simple network inequality. To get the extended network inequality, some variables $j \in Q_k^- \subseteq N_k^-$ in (9) are substituted by their upper bounds. Then

$$f_k = \frac{\sum_{i=1}^{k}(b_i^+ + \sum_{j\in Q_i^-} u_j)}{\delta} \text{ for } k = 1 \dots K$$

and

$$t_K = \sum_{i=1}^{K}(b_i + \sum_{j\in Q_i^-} u_j) - \sum_{i=1}^{K}\sum_{j\in N_i^+} x_j + \sum_{i=1}^{K}\sum_{j\in N_i^-\setminus Q_i^-} x_j.$$

After following the same steps as above the result is the extended network inequality. □

**Example 3.1.** *Some of the facets of the constant capacity lot-sizing problem in example 2.1 can be generated using a flow path cut separation algorithm. For the fixed charge path*

$$s_0 + x_1 - s_1 \leq 2$$
$$s_1 + x_2 - s_2 \leq 6$$

*the simple network inequality with* $C_1^+ = \{i_{x_1}\}$ *and* $C_2^+ = \{i_{x_2}\}$ *is*

$$x_1 + x_2 \leq 8y_1 + 6y_2 + s_2.$$

*As* $x_1 + x_2 - s_2 = 8 - s_0$ *it is equivalent to facet (29) of the conves hull shown in figure 2 in the appendix:*

$$s_0 \geq 8 - 8y_1 - 6y_2.$$

# 4 Aggregated c-MIR Cuts

The aggregated c-MIR cutting plane separation algorithm is among the most important separation algorithms in modern MIP solvers (see [4]). It goes back to [11] and [12] and its strength is that it implicitly generates a number of problem specific cutting planes such as residual capacity and (non-lifted) flow cover cuts. The aggregated c-MIR cut separation algorithm consists of three steps: aggregation, bound substitution and separation.

In the aggregation heuristic rows are aggregated along a path to form a base row for the cut. The rows are selected by a greedy algorithm similar to the one used earlier for flow path cuts. In the bound substitution step variables are replaced by their lower or upper (variable) bound based on simple rules. In the last step the cut is generated by choosing a value $\delta$ and complementing some of the variables to get the cut with the largest normalized violation. All these steps are heuristic and have some implementational challenges (see [12] and [8]).

As a mixing cut for a single row ($|T| = 1$) is the same as the MIR cut for this row it is obvious that a path mixing cut separation algorithm implemented in a certain way can generate aggregated c-MIR cuts implicitly. It is less obvious that an aggregated c-MIR cut separation algorithm can implicitly generate path mixing cuts. Example 4.1 shows that at least some of the mixing inequalities

that are facets of the convex hull of the constant capacity lot-sizing problem can be generated using an aggregated c-MIR cut separation algorithm. This only works if the algorithm does not only use the original rows of the constraint matrix but also generates cuts out of cuts. In compliance with the definition of rank for Chvátal-Gomory cuts (for example in [16]) we can say that at least some of the mixing inequalities are just higher rank MIR inequalities.

Separating aggregated c-MIR cuts from cuts generated in a previous round raises some difficulties. One is that by repeatedly dividing coefficients the resulting cuts might suffer from rounding errors. Appropriate scaling can make up for this. Another problem can be that if many cuts are generated and added to the problem the runtime increases from round to round. Careful cut selection can reduce this effect to some extent. Finally, an aggregated c-MIR algorithm can not find mixing cuts in the first round of the cut generation. It has to generate (and add) the right cuts in an earlier round and use an aggregation heuristic that selects the right rows for aggregation. All these factors make it difficult to rely on the aggregated c-MIR cut separation algorithm solely for generating mixing cuts. Nevertheless computational results such as those presented in section 6 show that in practice it works very well.

**Example 4.1.** *First of all note that some of the facets of the constant capacity lot-sizing problem from example 2.1 are mixing cuts from a single row and therefore MIR cuts from aggregated rows of a path. These can be found by an aggregated c-MIR cut separation algorithm easily. The facets that are mixing inequalities from more than one row typically can not be found in the first round of an aggregated c-MIR cut separation algorithm. In the following path*

$$\begin{aligned} x_1 - 2y_1 - s_1 &\leq 0 \\ s_1 + x_2 - s_2 &= 6 \\ s_2 + x_3 - s_3 &= 4 \\ s_3 + x_4 - s_4 &= 5, \end{aligned}$$

*the first row is the MIR cut generated from the flow balance constraint* $s_0 + x_1 - s_1 = 2$. *By aggregating this path and after substituting all variable bounds we get the mixed knapsack*

$$-t_1 - t_2 - t_3 - t_4 - s_4 + 8y_1 + 10y_2 + 10y_3 + 10y_4 \leq 15,$$

*where* $t_j$ *is the slack variable of the variable upper bound constraint of* $j$. *Applying the MIR procedure using* $\delta = 10$ *this results in the cut*

$$x_1 + x_2 + x_3 + x_4 - s_4 - 7y_1 - 5y_2 - 5y_3 - 5y_4 \leq 5$$

*which is equivalent to the facet found in example 2.1. A slightly more difficult example is facet (19). In this mixed integer path*

$$\begin{aligned} x_4 - 5y_4 - s_4 &\leq 0 \\ -s_3 - x_4 + s_4 &= -5 \\ -s_2 - x_3 + s_3 &= -4 \\ -s_1 - x_2 + s_2 &= -6 \\ -s_0 - x_1 + s_1 &= -5, \end{aligned}$$

*the first row is again a simple MIR cut. The other rows have been multiplied by* $-1$*. After adding a slack variable* $p$ *for the first row, aggregating, multiplying by* $-1$ *and substituting all variable upper bounds we get the following mixed knapsack*

$$-t_1 - t_2 - t_3 - p + 10y_1 + 10y_2 + 10y_3 + 5y_4 \leq 17.$$

*Applying the MIR procedure using* $\delta = 10$ *and substituting* $p$ *yields*

$$x_1 + x_2 + x_3 + x_4 - s_4 - 7y_1 - 7y_2 - 7y_3 - 5y_4 \leq 3$$

*which is equivalent to facet 19. Note that for both examples it is not necessary to complement variables.*

# 5 Path Mixing Cut Separation Algorithms

In this section we propose two separation algorithms based on the path mixing inequalities defined in section 2. The basic idea for both algorithms is to use the mixed knapsack constraints generated through aggregation and bound substitution in a c-MIR algorithm as described in [12] as base rows for mixing cuts. As we need greater than or equal inequalities the mixed knapsack constraints have to be multiplied by $-1$.

The first algorithm proposed is called the *uncapacitated path mixing cut separation algorithm* or uPMC for short. Its purpose is to be an alternative to the flow path cut separation algorithm that is slightly more general. It is called *uncapacitated* because the intention of this algorithm is to include the generation of the facet defining valid inequalities for uncapacitated lot-sizing problems.

As already seen in the proof of proposition 5 the $f_k$ of the rows of an aggregated path increase from one row to the next if we relax the right hand side to be positive and choose very large $\delta_k$. The algorithm uses the maximum of the integer coefficients of all mixed knapsacks processed so far as $\delta_k$. Thus through the iterations of the aggregation the algorithm gets mixed knapsacks from the aggregated rows of a path with increasing $f_k$. These rows can be added to a mixing cut one after the other until it finds a violated cut. The algorithm stops then because this limits the number of non-zero entries in the cut and reduces the runtime.

If a uPMC algorithm and a flow path cut algorithm use the same rule to find the next row of a path and the bound substitution in the uPMC algorithm is done in the right way both algorithms generate the same cuts from a fixed charge path (see section 3). In addition to these a uPMC algorithm can find more cuts because it is not limited to binary variables and hence might obtain strong cuts for instances where flow path cuts fail, for example for lot-sizing problems where the capacity is an integer multiple of a constant batch size (see [17]).

One of the inherent advantages of the *uncapacitated path mixing cut separation algorithm* is that it can be integrated into a c-MIR cut separation algorithm with little implementational effort. Algorithm 1 shows in pseudocode how this might look. Note that only lines 18 to 29 in algorithm 1 are different from a c-MIR algorithm. The function called `append_PMC` simply adds a new row to

**Algorithm 1** The uncapacitated path mixing cut separation algorithm

**procedure** MIR_AND_UPMC
  $\delta \leftarrow 0$
  **for** each usable $row$ **do**
    $aggregatedRow \leftarrow \emptyset$, $pmCut = \emptyset$, $f_{last} \leftarrow 0$
    $nextRow \leftarrow row$
    $MIRCutfound \leftarrow FALSE$, $pmCutfound \leftarrow FALSE$
    **for** $k = 1 \dots K$ **do**
      $aggregatedRow \leftarrow aggregatedRow + nextRow$
      **for** $mult = 1, -1$ **do**
        $mixedKnapsack \leftarrow$ generate_MK($mult \cdot aggregatedRow$)
        **if** $MIRCutfound = FALSE$ **then**
          $MIRCut \leftarrow$ generate_cMIR($mixedKnapsack$)
          **if** violated($MIRCut$) **then**
            add_Cut($MIRCut$)
            $MIRCutfound \leftarrow TRUE$
          **end if**
        **end if**
        **if** $mult = -1$ AND $pmCutfound = FALSE$ **then**
          $\delta \leftarrow \max\{\delta, \max_{j \in mixedKnapsack}\{g_j\}\}$
          $f_{current} \leftarrow \frac{b_{MK}}{\delta} - \lfloor \frac{b_{MK}}{\delta} \rfloor$
          **if** $f_{current} > f_{last}$ **then**
            $pmCut \leftarrow$ append_PMC($pmCut$,$mixedKnapsack$,$f_{current}$)
            $f_{last} \leftarrow f_{current}$
          **end if**
          **if** violated($pmCut$) **then**
            add_Cut($pmCut$)
            $pmCutfound \leftarrow TRUE$
          **end if**
        **end if**
      **end for**
      $nextRow \leftarrow$ get_next_Row($aggregatedRow$)
    **end for**
  **end for**
**end procedure**

the best path mixing cut found so far. It also decides on the sets $I_1$ and $I_2$ locally by placing a variable $j$ in $I_1$ if

$$\max\{\bar{g}_j, g_j\} < (f_{current} - f_{last}) \left\lceil \frac{g_{ji}}{\delta_{i_1}} \right\rceil$$

where $\bar{g}_j$ is the coefficient of $j$ in the best path mixing cut found so far. Note that the two path mixing algorithms presented here only use (3), the first of the two mixing inequalities. Also note that in an uPMC algorithm variables are not complemented as in an aggregated c-MIR algorithm.

The second algorithm is called the *capacitated path mixing cut separation algorithm* or cPMC for short. The intention behind it is to investigate the potential of path mixing cuts that go beyond the scope of flow path cuts. The way the algorithm is designed enables the generation of the facet defining valid inequalities for constant capacity lot-sizing problems known as $(k, l, S, I)$ inequalities (see [17] and [18]). That mixing inequalities can be used to generate these is already shown in [9].

Again the mixed knapsacks from a c-MIR algorithm are used. Additionally the algorithm also saves the $\delta_k$ and the corresponding $f_k$ that resulted in the most violated c-MIR cut for this mixed knapsack. Note that in order to do this it either needs a c-MIR algorithm that works on $\geq$-mixed knapsacks or it has to recompute the $f_k$ correspondingly. The next step is to compute a value $\beta_k$. The value for $\beta_k$ is computed as

$$\beta_k = \lfloor \bar{b}_{i_\tau} \rfloor + 1 - \sum_{i=1}^{i_\tau} \sum_{j \in I_2} \left\lceil \frac{g_{ji}}{\delta_k} \right\rceil y_j^*.$$

This corresponds to the $\beta$ in the simple mixing set separation (see section 2). Note that in this case in contrast to the separation algorithm for mixing cuts for the simple mixing sets this procedure can not guarantee to find the most violated cut. The reason for this is that $\beta_k$ does not consider the values that the continuous variables take in the current solution of the LP relaxation. To overcome this drawback another option is to use the violation of the MIR cuts from the aggregated rows instead of $\beta_k$. Computational experiments with various instances showed no significant difference.

The next step is to sort the stored mixed knapsacks by decreasing $\beta_k$ and iteratively try to generate path mixing cuts for the first $t = 2, 3, 4, \ldots K$ mixed knapsacks with largest $\beta_k$. This involves sorting by $f_k$ and then adding mixed knapsacks one by one if the violation improves until a violated path mixing cut is found. Algorithm 2 illustrates this procedure using pseudocode. The same procedure `append_PMC` as for uPMCs can be used including the same rule to decide on $I_2$. An important difference between uPMC and cPMC is that for cPMC the mixed knapsacks, the $\delta_k$ and the $\beta_k$ have to be stored. Another difference is that in cPMC the variables in the mixed knapsacks are complemented in the same way as in a c-MIR algorithm.

For all algorithms based on paths the row selection in the aggregation is crucial for obtaining good computational results. For typical lot-sizing instances generating the right paths is easy because the algorithm can choose the only variable with a negative coefficient and search for an equality row where this variable has a positive coefficient. This is basically what is done in a flow

**Algorithm 2** The capacitated path mixing cut separation algorithm

```
1: procedure MIR_AND_CPMC
2:     for each usable row do
3:         aggregatedRow ← ∅, pathRows = ∅, pmCut = ∅, f_last ← 0
4:         nextRow ← row
5:         MIRCutfound ← FALSE, pmCutfound ← FALSE
6:         for k = 1...K do
7:             aggregatedRow ← aggregatedRow + nextRow
8:             for mult = 1, −1 do
9:                 mixedKnapsack ← generate_MK(mult · aggregateRow)
10:                MIRCut ← generate_cMIR(mixedKnapsack,δ_best)
11:                pathRow_k ← mixedKnapsack
12:                β_k ← compute_beta(MIRCut)
13:                δ_k ← δ_best
14:                if MIRCutfound = FALSE then
15:                    if violation(MIRCut)> 0 then
16:                        add_Cut(MIRCut)
17:                        MIRCutfound ← TRUE
18:                    end if
19:                end if
20:                if pmCutfound = FALSE AND mult = −1 then
21:                    pathRow ← sort_pathRows_by_decreasing_beta(1,k)
22:                    f_last ← 0
23:                    i ← 1
24:                    while i ≤ k AND pmCutfound = FALSE do
25:                        pathRow ← sort_pathRows_by_increasing_f(1,i)
26:                        t ← 1
27:                        while t ≤ i AND pmCutfound = FALSE do
28:                            pmCut ← append_PMC(pmCut,pathRow_{k_t},f_last)
29:                            if violation(pmCut) > 0 then
30:                                add_Cut(pmCut)
31:                                pmCutfound ← TRUE
32:                            end if
33:                            t ← t + 1
34:                        end while
35:                        i ← i + 1
36:                    end while
37:                end if
38:            end for
39:            nextRow ← get_next_Row(aggregatedRow)
40:        end for
41:    end for
42: end procedure
```

path cut separation algorithm. For more general situations as they appear in network design problems one might also want to consider variables with positive coefficients. For the computational results in section 6 the algorithms use the following rule:

Step 1. select the variable $r$ with the largest outflow in the current aggregated row $t$:

$$\max_{r \in N^+ \cup N^-} -a_{rt}.$$

Step 2. search a row $u$ that is an equality constraint or where the slack variable in the current LP solution is minimal.

Step 3. scale row $u$ by $\gamma$ so that $a_{rt} + \gamma a_{ru} = 0$ and return the scaled row $u$.

When implementing MIR based separation algorithms one has to be especially careful not to generate invalid inequalities due to rounding errors. In the implementations for this paper it is tried to avoid generating invalid cuts by checking for all coefficients $\pi_j$ of generated cuts whether $|\pi_j| \leq \sigma$ where $\sigma$ (by default 1e-7) is a parameter that describes the smallest coefficient allowed in the constraint matrix. If $|\pi_j| \leq \sigma$, the algorithm sets $\pi_j = 0$ but subtracts $\pi_j l_j$ (or $\pi_j u_j$ respectively) from the right hand side. The same method is used after aggregation to avoid very small coefficients in the mixed knapsacks.

# 6 Computational Results

This section gives an impression of the impact mixing cuts have on solving MIP problems with MIP solvers. For the computational tests the commercial LP and MIP solver mops [14], version 9.13, is used. Note that in mops generated cuts are not automatically added to the problem. Instead they are stored in a cut pool and only some of them are selected in each round.

In the following we compare five different variants of cut separation. All of these variants are based on aggregated c-MIR cut separation algorithms as described by Marchand and Wolsey [12]. The variant named MIR only uses the original rows of the constraint matrix as input. MIR+ allows the algorithm to generate cuts out of cuts added in a previous round. The other three variants combine MIR with separation algorithms for extended flow path cuts [21] (FP), uncapacitated path mixing cuts (uPMC) and capacitated path mixing cuts (cPMC). In all algorithms the maximal length for the paths investigated is six.

The problem instances for these computational results are a selection from MIPLIB3 [3], MIPLIB2003 [1] and MITTELMANN [15]. We selected instances were we observed notable differences between the dual bounds obtained using MIR and MIR+ as well as instances that have a network structure. Instances that caused numerical difficulties in the mops LP solver were excluded. Table 3 in the appendix lists the 20 selected problem instances and their statistics.

As pointed out for example by Margot in [13], comparing the performance of cutting plane separation algorithms causes many difficulties. The problem is that a larger improvement in the dual bound does not necessarily indicate that a cutting plane separation algorithm is better than another one. However

| name | MIR | MIR+ | FP | uPMC | cPMC |
|---|---|---|---|---|---|
| aflow30a | 0.211 | **0.268** | 0.262 | 0.216 | 0.199 |
| aflow40b | 0.125 | 0.144 | **0.212** | 0.138 | 0.150 |
| bc1 | **0.619** | 0.618 | **0.619** | **0.619** | **0.619** |
| bienst1 | 0.066 | **0.067** | 0.067 | 0.066 | 0.066 |
| bienst2 | 0.074 | **0.075** | 0.074 | 0.074 | 0.074 |
| dano3mip | 0.002 | 0.002 | **0.002** | 0.002 | 0.002 |
| dcmulti | 0.095 | 0.095 | 0.095 | 0.095 | **0.098** |
| fiber | 0.247 | **0.337** | 0.247 | 0.247 | 0.247 |
| fixnet6 | 0.813 | 0.807 | **0.829** | 0.813 | 0.813 |
| mod011 | 0.431 | 0.458 | 0.431 | 0.469 | **0.477** |
| modglob | 0.805 | 0.796 | **0.910** | 0.776 | 0.736 |
| pp08a | 0.892 | **0.960** | 0.920 | 0.906 | 0.907 |
| pp08aCUTS | 0.818 | **0.883** | 0.815 | 0.848 | 0.846 |
| ran10x26 | 0.143 | **0.181** | 0.143 | 0.143 | 0.168 |
| ran12x21 | 0.187 | **0.226** | 0.187 | 0.187 | 0.190 |
| ran13x13 | 0.145 | **0.186** | 0.145 | 0.145 | 0.162 |
| set1ch | 0.876 | **0.994** | 0.951 | 0.992 | 0.990 |
| tr12-30 | 0.855 | **0.989** | 0.965 | 0.978 | 0.977 |
| vpm1 | 0.782 | **1.000** | 0.782 | 0.782 | 0.782 |
| vpm2 | 0.537 | **0.551** | 0.537 | 0.537 | 0.534 |
| GEO | 0.248 | 0.276 | 0.264 | 0.255 | 0.259 |

Table 1: Summary of the results for the first computational experiment. GEO indicates the geometric mean over all instances.

the dual bound can be compared to see differences between separation algorithms. Table 6 shows ratios computed from the dual bounds after 10 rounds of cuts. The ratios shown are the percentage of the gap between the values of the root relaxation and the optimal solution (or the best known solution) that the separation algorithm was able to close, i. e.

$$r = \frac{z_{cut} - z_{LP}}{z_{IP} - z_{LP}}.$$

In this experiment all other cutting plane separation algorithms are deactivated. The machine used has a Pentium 4 (3.4 GHz) CPU and 3.49 GB RAM. It uses Windows XP Professional as operating system.

A conclusion that can be drawn form these results is that MIR+, FP, uPMC and cPMC achieve similar bounds for all of these problem instances. MIR on the other hand seems to improve the bound much less for some instances. These instances are primarily the lot-sizing problems `pp08a`, `pp08aCUTS`, `set1ch` and `tr12-30`. As shown in this paper the latter four algorithms in contrast to MIR are able to generate mixing cuts and hence can improve the bound much more. For other instances, e. g. `fiber`, these results show that path mixing cuts can not explain the difference between MIR and MIR+; but it is also known that `fiber` can be solved easily using lifted flow cover cuts and that some lifted flow cover cuts can be generated using mixing inequalities (as shown in [9]). A further investigation of this is left for future research.

The second computational experiment in this paper is aimed at comparing

| name | MIR | MIR+ | FP | uPMC | cPMC |
|---|---|---|---|---|---|
| aflow30a | 1.48 | 2.20 | 2.08 | **1.06** | 1.26 |
| aflow40b | **6.10%** | 7.07% | 22.78% | 10.67% | 9.36% |
| bc1 | 6.58% | **6.37%** | 6.58% | 6.58% | 6.58% |
| bienst1 | 3.70 | 3.37 | 3.34 | **2.61** | **2.61** |
| bienst2 | 28.37 | 21.47 | 25.22 | 21.06 | **17.96** |
| dano3mip | 22.71% | 22.64% | 24.50% | 25.99% | **20.78%** |
| dcmulti | **0.01** | **0.01** | **0.01** | **0.01** | **0.01** |
| fiber | **0.02** | 0.03 | **0.02** | 0.04 | **0.02** |
| fixnet6 | **0.02** | 0.04 | 0.03 | 0.04 | 0.04 |
| mod011 | **1.51** | 2.00 | 1.90 | 1.74 | 1.83 |
| modglob | 0.04 | **0.01** | **0.01** | **0.01** | **0.01** |
| pp08a | 0.05 | 0.04 | 0.07 | 0.04 | **0.03** |
| pp08aCUTS | **0.04** | 0.06 | **0.04** | **0.04** | **0.04** |
| ran10x26 | 5.07 | **4.11** | 5.09 | 5.08 | 4.85 |
| ran12x21 | 17.64 | **14.30** | 17.57 | 17.70 | 17.66 |
| ran13x13 | 5.80 | **4.65** | 5.70 | 5.70 | 6.24 |
| set1ch | 1.36% | **0.02** | 0.10 | 0.04 | 0.03 |
| tr12-30 | 11.52% | 0.38% | 2.41% | **0.30%** | 0.35% |
| vpm1 | **0.01** | **0.01** | **0.01** | **0.01** | **0.01** |
| vpm2 | 1.03 | 0.77 | 1.03 | **0.74** | 0.81 |

Table 2: Summary of the results for the second computational experiment. The numbers shown are total time in minutes or the duality gap in percent (%) after two hours.

the separation algorithms in a realistic setting. We now compare the total time to solve the 20 problem instances to optimality; for those instances that mops does not solve in two hours we compare the duality gap after the mops MIP solver reached this time limit. In these tests we do not deactivate the other default separation algorithms in mops. These are (lifted) cover cuts, clique cuts, implication cuts, (lifted) flow cover cuts and Gomory mixed integer cuts. The same machine is used. The results are shown in table 6.

The most notable fact is, that using the MIR separation algorithm mops fails to solve `set1ch` in two hours whereas only a few seconds are needed if one of the alternatives is used. So clearly mixing inequalities are needed to solve this problem fast. A similar situation can be observed for `tr12-30` where the duality gap after using MIR is much bigger than when using one of the alternatives. The difference between the algorithms that generate mixing inequalities, however, is small.

# 7 Conclusions

The separation of mixing inequalities from paths is needed to solve some mixed integer programming problems, namely those that have a lot-sizing structure, with an MIP solver. As shown in this paper there are several possibilities to do this. Current MIP solvers use flow path and aggregated c-MIR cuts based on more than the original rows but both of these have several disadvantages. Flow path cuts are limited to mixed 0-1 problems and usually do not reuse

parts of other separation algorithms. The disadvantages of aggregated c-MIR cuts based on more than the original rows are for example numerical instability and an increasing runtime as discussed in section 4. Another possibility to solve lot-sizing problems with an MIP solver is to use an extended formulation as described in [18], but this requires one to change large parts of the model and to add additional variables which might increase the size of the model significantly. This is currently beyond the scope of what MIP solvers do automatically.

In this paper we suggest two separation algorithms that generate mixing inequalities explicitly. These two algorithms performed similarly for the typical problem instances but we suppose that instances exist were cPMC cuts have an advantage. Whether this advantage is worth the increase in runtime is a question for future research. The uncapacitated path mixing algorithm can be seen as just another way of implementing flow path cuts with a small extension. An attractive feature of this algorithm is that it can reuse parts of an aggregated c-MIR algorithm which has some advantages from a software design point of view. It also extends flow path cuts to problems with general integer variables. For the use in an MIP solver we think that the uPMC algorithm constitutes a practically well working solution. A cPMC algorithm could be an option that the user can turn on explicitly if he thinks it might work well on his specific problem instance. The fact that the path mixing algorithms reuse parts of the c-MIR algorithm has the additional advantage that future improvements of the c-MIR algorithm can also improve the path mixing algorithms.

In the literature more and more complicating structures are studied and new valid inequalities are found. For example van Vyve studied the continuous mixing set in [22] and showed that valid inequalities for the lot-sizing problem with backlogging can be generated using it. Theoretically all these valid inequalities could be used in an MIP solver but the challenge is to use separation algorithms that are *general enough* in the sense that they find cuts for many different problem types. In this respect the path mixing inequalities are already questionable. Whether implementing a separation algorithm based on other simple sets can substantially improve the performance of MIP solvers stays a question for future research.

**Acknowledgements:** The author would like to thank Laurence Wolsey for his support and many hints and comments.

# References

[1] Tobias Achterberg, Thorsten Koch, and Alexander Martin. MIPLIB 2003. *Operations Research Letters*, 34(4):1–12, 2006. See `http://miplib.zib.de`.

[2] Egon Balas and Michael Perregaard. A precise correspondence between lift-and-project cuts, simple disjunctive cuts, and mixed integer Gomory cuts for 0-1 programming. *Mathematical Programming Series B*, 94(2–3):221–245, 2003.

[3] Robert E. Bixby, Sebastián Ceria, Cassandra M. McZeal, and Martin W. P. Savelsbergh. An updated mixed integer programming library: MIPLIB 3.0. *Optima*, 58:12–15, 1998.

[4] Robert E. Bixby and Edward Rothberg. Progress in computational mixed integer programming - a look back from the other side of the tipping point. *Annals of Operations Research*, 149(1):37–41, 2007.

[5] Alberto Caprara and Matteo Fischetti. $0, \frac{1}{2}$-Chvátal-Gomory cuts. *Mathematical Programming*, 74(3):221–235, 1996.

[6] Thomas Christof and Andreas Löbel. PORTA – POlyhedron Representation Transformation Algorithm, 2008. Version 1.4.0. See `http://www.zib.de/Optimization/Software/Porta/`.

[7] Dash Optimization Inc. Xpress-Optimizer, 2008. See `http://www.dashoptimization.com`.

[8] João Gonçalves and Laszlo Ladanyi. An implementation of a separation procedure for mixed integer rounding inequalities. IBM Research Report RC23686, IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, N.Y., August 2005.

[9] Oktay Günlük and Yves Pochet. Mixing mixed-integer inequalities. *Mathematical Programming Series A*, 90(3):429–457, 2001.

[10] ILOG Inc. ILOG CPLEX 11.0, 2008. See `http://www.cplex.com`.

[11] Hugues Marchand. *A Polyhedral Study of the Mixed Knapsack Set and its Use to Solve Mixed Integer Programs*. PhD thesis, Faculté des Sciences Appliquées, Université catholique de Louvain, 1998.

[12] Hugues Marchand and Laurence A. Wolsey. Aggregation and mixed integer rounding to solve MIPs. *Operations Research*, 49(3):363–371, 2001.

[13] François Margot. Testing cut generators for mixed-integer linear programming. Technical report, Tepper School of Business, Carnegie Mellon University, 2007.

[14] Mathematische Optimierungssysteme GmbH. mops LP/MIP Optimizer, 2008. See `http://www.mops-optimizer.com`.

[15] Hans Mittelmann. Benchmarks for optimization software, 2007. See `http://plato.asu.edu/bench.html`.

[16] George L. Nemhauser and Laurence A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, 1999.

[17] Yves Pochet and Laurence A. Wolsey. Lot-sizing with constant batches: Formulation and valid inequalities. *Mathematics of Operations Research*, 18(4):767 – 785, 1993.

[18] Yves Pochet and Laurence A. Wolsey. *Production planning by mixed integer programming*. Springer, 2006.

[19] Tony J. van Roy and Laurence A. Wolsey. Valid inequalities and separation for uncapacitated fixed charge networks. *Operations Research Letters*, 4:105 – 213, 1985.

[20] Tony J. van Roy and Laurence A. Wolsey. Valid inequalities for mixed 0-1 programs. *Discrete Applied Mathematics*, 14:199–213, 1986.

[21] Tony J. van Roy and Laurence A. Wolsey. Solving mixed integer programming problems using automatic reformulation. *Operations Research*, 35(1):45–57, 1987.

[22] Mathieu van Vyve. The continuous mixing polyhedron. *Mathematics of Operations Research*, 30(2):441 – 452, 5 2005.

[23] Laurence A. Wolsey. *Integer Programming*. John Wiley & Sons Inc., 1998.

```
DIM = 9
VALID
0 2 6 4 5 1 1 1 1

LOWER_BOUNDS
0 0 0 0 0 0 0 0 0

UPPER_BOUNDS
17 10 10 10 10 1 1 1 1

INEQUALITIES_SECTION
x5 - 10 x9 <= 0
x4 - 10 x8 <= 0
x3 - 10 x7 <= 0
x2 - 10 x6 <= 0
x1 + x2>= 2
x1 + x2 + x3 >= 8
x1 + x2 + x3 + x4 >= 12
x1 + x2 + x3 + x4 + x5 >= 17
x1 >= 0
x2 >= 0
x3 >= 0
x4 >= 0
x5 >= 0
x6 >= 0
x7 >= 0
x8 >= 0
x9 >= 0
x1 <= 17
x2 <= 10
x3 <= 10
x4 <= 10
x5 <= 10
x6 <= 1
x7 <= 1
x8 <= 1
x9 <= 1
END
```

Figure 1: Input file for the example (PORTA-Format).

```
DIM = 9

VALID
2 0 6 4 5 1 1 1 1

INEQUALITIES_SECTION
(  1) -10x1   -5x3   -9x5-80x6-30x7-65x8      <= -145
(  2) - 3x1   - x3       -24x6-14x7-21x8-15x9 <=  -45
(  3) - 3x1   - x3   -3x5-24x6-14x7-21x8      <=  -45
(  4) - 3x1   - x3       -24x6-12x7-19x8-15x9 <=  -43
(  5) - 3x1   - x3   -3x5-24x6-12x7-19x8      <=  -43
(  6) - 2x1   - x3       -16x6- 6x7-13x8- 9x9 <=  -29
(  7) - 2x1   - x3   - x5-12x6- 2x7- 9x8      <=  -21
(  8) - 2x1   - x3       -12x6- 2x7- 9x8- 5x9 <=  -21
(  9) - 2x1   - x3       -16x6- 6x7- 4x8      <=  -20
( 10) -  x1-x2- x3   - x5          - 9x8      <=  -17
( 11) -  x1-x2- x3-x4- x5                     <=  -17
( 12) -  x1-x2- x3                - 9x8- 5x9 <=  -17
( 13) -  x1-x2- x3-x4                   - 5x9 <=  -17
( 14) - 2x1   - x3       -12x6- 2x7- 4x8      <=  -16
( 15) -  x1   - x3   - x5- 8x6     - 7x8      <=  -15
( 16) -  x1   - x3       - 8x6     - 7x8- 5x9 <=  -15
( 17) -  x1          - x5- 7x6- 7x7- 7x8      <=  -14
( 18) -  x1       -x4- x5- 7x6- 7x7           <=  -14
( 19) -  x1              - 7x6- 7x7- 7x8- 5x9 <=  -14
( 20) -  x1       -x4    - 7x6- 7x7     - 5x9 <=  -14
( 21) -  x1          - x5- 8x6- 6x7- 5x8      <=  -13
( 22) -  x1-x2       - x5     - 6x7- 5x8      <=  -13
( 23) -  x1              - 8x6- 6x7- 5x8- 5x9 <=  -13
( 24) -  x1-x2                - 6x7- 5x8- 5x9 <=  -13
( 25) -  x1-x2- x3                - 4x8      <=  -12
( 26) -  x1          - x5- 7x6- 5x7- 5x8      <=  -12
( 27) -  x1              - 7x6- 5x7- 5x8- 5x9 <=  -12
( 28) -  x1   - x3       - 8x6     - 2x8      <=  -10
( 29) -  x1              - 8x6- 6x7           <=   -8
( 30) -  x1-x2                - 6x7           <=   -8
( 31) -  x1       -x4    - 2x6- 2x7           <=   -4
( 32) -  x1              - 2x6- 2x7- 2x8      <=   -4
( 33) -  x1              - 2x6                <=   -2
( 34)               + x5               -10x9 <=    0
( 35)             +x4              -10x8      <=    0
( 36)         + x3            -10x7           <=    0
( 37)      +x2           -10x6                <=    0
( 38)      -x2                                 <=    0
( 39)         - x3                             <=    0
( 40)             -x4                          <=    0
( 41)               - x5                       <=    0
( 42)                                  +  x9 <=    1
( 43)                               +  x8      <=    1
( 44)                          +  x7           <=    1
( 45)                     +  x6                <=    1
( 46) +  x1                                    <=   17
END
```

Figure 2: Convex hull of the example (PORTA-Format).

| NAME | N | BIN | M | LP | IP |
|---|---|---|---|---|---|
| aflow30a | 842 | 421 | 479 | 983.17 | 1158.00 |
| aflow40b | 2728 | 1364 | 1442 | 1005.66 | 1168.00 |
| bc1 | 1751 | 252 | 1913 | 0.78 | 3.34 |
| bienst1 | 505 | 28 | 576 | 11.72 | 46.75 |
| bienst2 | 505 | 35 | 576 | 11.72 | 54.60 |
| dano3mip | 13873 | 552 | 3202 | 576.23 | 727.89* |
| dcmulti | 548 | 75 | 290 | 183975.54 | 188182.00 |
| fiber | 1298 | 1254 | 363 | 156082.52 | 405935.18 |
| fixnet6 | 878 | 378 | 478 | 1200.88 | 3983.00 |
| mod011 | 10958 | 96 | 4480 | -62121982.55 | -54558535.01 |
| modglob | 422 | 98 | 291 | 20430947.62 | 20740508.09 |
| pp08a | 240 | 64 | 136 | 2748.35 | 7350.00 |
| pp08aCUTS | 240 | 64 | 246 | 5480.61 | 7350.00 |
| ran10x26 | 520 | 260 | 296 | 3857.02 | 4270.00 |
| ran12x21 | 504 | 252 | 285 | 3157.38 | 3664.00 |
| ran13x13 | 338 | 169 | 195 | 2691.44 | 3252.00 |
| set1ch | 712 | 240 | 492 | 32007.73 | 54537.75 |
| tr12-30 | 1080 | 360 | 750 | 18124.17 | 130596.00 |
| vpm1 | 378 | 168 | 234 | 15.42 | 20.00 |
| vpm2 | 378 | 168 | 234 | 9.89 | 13.75 |

Table 3: Statistics for the problem instances used in the computational results. To the best knowledge of the author dano3mip is unsolved. For this instance IP reports the best solution mops found.

| | | MIR | | | MIR+ | | | FP | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| name | LP | MIR cuts | xLP | sec. | MIR cuts | xLP | sec. | MIR cuts | FP cuts | xLP | sec. |
| aflow30a | 983.17 | 69 | 1020.00 | 0.67 | 87 | 1030.00 | 1.80 | 59 | 36 | 1029.00 | 0.94 |
| aflow40b | 1005.66 | 108 | 1026.00 | 7.31 | 118 | 1029.00 | 25.76 | 97 | 48 | 1040.00 | 5.99 |
| bc1 | 0.78 | 30 | 2.36 | 540.11 | 30 | 2.36 | 532.03 | 30 | 0 | 2.36 | 546.75 |
| bienst1 | 11.72 | 71 | 14.03 | 0.28 | 89 | 14.07 | 0.34 | 64 | 32 | 14.06 | 0.34 |
| bienst2 | 11.72 | 96 | 14.91 | 0.33 | 111 | 14.93 | 0.36 | 67 | 46 | 14.91 | 0.42 |
| dano3mip | 576.23 | 446 | 576.50 | 32.67 | 432 | 576.50 | 31.78 | 323 | 293 | 576.54 | 47.91 |
| dcmulti | 183975.54 | 39 | 184375.07 | 0.06 | 42 | 184376.06 | 0.08 | 39 | 0 | 184375.07 | 0.05 |
| fiber | 156082.52 | 30 | 217748.92 | 0.09 | 47 | 240249.39 | 0.17 | 30 | 0 | 217748.92 | 0.09 |
| fixnet6 | 1200.88 | 32 | 3461.63 | 0.34 | 40 | 3446.25 | 0.34 | 18 | 47 | 3507.56 | 0.28 |
| mod011 | -62121982.55 | 195 | -58863761.93 | 0.78 | 183 | -58661526.54 | 0.91 | 195 | 0 | -58863790.68 | 0.81 |
| modglob | 20430947.62 | 78 | 20680087.64 | 0.09 | 89 | 20677270.71 | 0.12 | 68 | 50 | 20712524.28 | 0.12 |
| pp08a | 2748.35 | 111 | 6854.66 | 0.09 | 142 | 7164.63 | 0.14 | 104 | 43 | 6983.39 | 0.12 |
| pp08aCUTS | 5480.61 | 87 | 7009.72 | 0.14 | 90 | 7131.27 | 0.14 | 75 | 22 | 7004.98 | 0.14 |
| ran10x26 | 3857.02 | 43 | 3916.03 | 0.28 | 49 | 3931.67 | 0.67 | 43 | 0 | 3916.03 | 0.30 |
| ran12x21 | 3157.38 | 35 | 3251.97 | 0.33 | 49 | 3271.99 | 0.59 | 35 | 0 | 3251.97 | 0.33 |
| ran13x13 | 2691.44 | 44 | 2772.83 | 0.12 | 48 | 2795.48 | 0.30 | 44 | 0 | 2772.83 | 0.14 |
| set1ch | 32007.73 | 361 | 51736.41 | 0.49 | 367 | 54398.60 | 0.34 | 221 | 203 | 53439.61 | 0.48 |
| tr12-30 | 18124.17 | 730 | 114276.30 | 0.56 | 770 | 129327.90 | 0.76 | 401 | 503 | 126673.13 | 0.77 |
| vpm1 | 15.42 | 56 | 19.00 | 0.08 | 48 | 20.00 | 0.06 | 56 | 0 | 19.00 | 0.09 |
| vpm2 | 9.89 | 94 | 11.96 | 0.14 | 99 | 12.02 | 0.22 | 94 | 0 | 11.96 | 0.14 |

Table 4: Results for 10 rounds, all other cuts deactivated. *xLP* shows the objective function value of the root relaxation after adding selected cuts. The column *sec.* reports the time spent for generating and selecting the cuts as well as for resolving the LP relaxations. See the next page for uPMC and cPMC.

| | | uPMC | | | | cPMC | | | |
|---|---|---|---|---|---|---|---|---|---|
| name | LP | MIR cuts | PM cuts | xLP | sec. | MIR cuts | PM cuts | xLP | sec. |
| aflow30a | 983.17 | 59 | 13 | 1021.00 | 1.41 | 46 | 25 | 1018.00 | 1.66 |
| aflow40b | 1005.66 | 108 | 10 | 1028.00 | 12.55 | 90 | 14 | 1030.00 | 11.30 |
| bc1 | 0.78 | 30 | 0 | 2.36 | 641.78 | 30 | 0 | 2.36 | 652.12 |
| bienst1 | 11.72 | 59 | 16 | 14.04 | 0.30 | 56 | 17 | 14.04 | 0.30 |
| bienst2 | 11.72 | 80 | 19 | 14.91 | 0.34 | 80 | 19 | 14.91 | 0.34 |
| dano3mip | 576.23 | 317 | 119 | 576.52 | 34.38 | 325 | 117 | 576.53 | 38.84 |
| dcmulti | 183975.54 | 20 | 10 | 184375.90 | 0.05 | 5 | 24 | 184389.25 | 0.06 |
| fiber | 156082.52 | 30 | 0 | 217748.92 | 0.09 | 30 | 0 | 217748.92 | 0.08 |
| fixnet6 | 1200.88 | 32 | 0 | 3461.63 | 0.75 | 32 | 0 | 3461.63 | 0.72 |
| mod011 | -62121982.55 | 206 | 1 | -58572842.34 | 0.91 | 183 | 20 | -58516878.74 | 0.91 |
| modglob | 20430947.62 | 58 | 29 | 20671025.11 | 0.12 | 55 | 29 | 20658768.87 | 0.12 |
| pp08a | 2748.35 | 92 | 26 | 6918.36 | 0.12 | 93 | 20 | 6921.69 | 0.12 |
| pp08aCUTS | 5480.61 | 58 | 30 | 7066.56 | 0.16 | 62 | 31 | 7062.38 | 0.17 |
| ran10x26 | 3857.02 | 43 | 0 | 3916.03 | 0.47 | 37 | 9 | 3926.28 | 0.39 |
| ran12x21 | 3157.38 | 35 | 0 | 3251.97 | 0.47 | 35 | 2 | 3253.50 | 0.50 |
| ran13x13 | 2691.44 | 44 | 0 | 2772.83 | 0.26 | 39 | 5 | 2782.05 | 0.25 |
| set1ch | 32007.73 | 182 | 176 | 54363.50 | 0.34 | 182 | 174 | 54308.46 | 0.39 |
| tr12-30 | 18124.17 | 293 | 411 | 128134.45 | 0.62 | 296 | 410 | 128060.86 | 0.74 |
| vpm1 | 15.42 | 56 | 0 | 19.00 | 0.11 | 60 | 7 | 19.00 | 0.11 |
| vpm2 | 9.89 | 94 | 0 | 11.96 | 0.22 | 93 | 7 | 11.95 | 0.20 |

Table 5: Results for 10 rounds, all other cuts deactivated. *xLP* shows the objective function value of the root relaxation after adding selected cuts. The column *sec.* reports the time spent for generating and selecting the cuts as well as for resolving the LP relaxations. See the previous page for MIR, MIR+ and FP.

| | MIR | | | | MIR+ | | | | FP | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NAME | cuts | xLP | IP | min / % | cuts | xLP | IP | min / % | cuts | xLP | IP | min / % |
| aflow30a | 117 | 1058.00 | 1158.00 | 1.48 | 130 | 1059.00 | 1158.00 | 2.20 | 128 | 1061.00 | 1158.00 | 2.08 |
| aflow40b | 185 | 1073.00 | 1181.00 | **6.10%** | 200 | 1071.00 | 1202.00 | 7.07% | 292 | 1076.00 | 1431.00 | 22.78% |
| bc1 | 30 | 2.36 | 3.35 | 6.58% | 30 | 2.36 | 3.35 | 6.37% | 30 | 2.36 | 3.35 | 6.58% |
| bienst1 | 121 | 14.08 | 46.75 | 3.70 | 87 | 14.06 | 46.75 | 3.37 | 91 | 14.07 | 46.75 | 3.34 |
| bienst2 | 169 | 14.95 | 54.60 | 28.37 | 114 | 14.93 | 54.60 | 21.47 | 120 | 14.93 | 54.60 | 25.22 |
| dano3mip | 448 | 576.50 | 746.04 | 22.71% | 427 | 576.50 | 745.30 | 22.64% | 615 | 576.55 | 763.76 | 24.50% |
| dcmulti | 122 | 186924.72 | 188182.00 | **0.01** | 94 | 186871.87 | 188182.00 | **0.01** | 122 | 186924.72 | 188182.00 | **0.01** |
| fiber | 90 | 383141.97 | 405935.18 | **0.02** | 104 | 384117.66 | 405935.18 | 0.03 | 90 | 383141.97 | 405935.18 | **0.02** |
| fixnet6 | 97 | 3661.17 | 3983.00 | **0.02** | 99 | 3694.05 | 3983.00 | 0.04 | 123 | 3693.16 | 3983.00 | 0.03 |
| mod011 | 641 | -57228628.22 | -54558535.01 | 1.51 | 680 | -57152277.15 | -54558535.01 | 2.00 | 644 | -57179478.56 | -54558535.01 | 1.90 |
| modglob | 127 | 20724948.91 | 20740508.09 | 0.04 | 129 | 20723848.31 | 20740508.09 | 0.01 | 143 | 20728117.03 | 20740508.09 | **0.01** |
| pp08a | 138 | 7071.97 | 7350.00 | 0.05 | 166 | 7169.79 | 7350.00 | 0.04 | 170 | 7093.31 | 7350.00 | 0.07 |
| pp08aCUTS | 109 | 7121.00 | 7350.00 | **0.04** | 115 | 7153.56 | 7350.00 | 0.06 | 120 | 7125.96 | 7350.00 | **0.04** |
| ran10x26 | 133 | 4030.55 | 4270.00 | 5.07 | 166 | 4045.88 | 4270.00 | **4.11** | 133 | 4030.55 | 4270.00 | 5.09 |
| ran12x21 | 198 | 3394.44 | 3664.00 | 17.64 | 196 | 3395.44 | 3664.00 | **14.30** | 198 | 3394.44 | 3664.00 | 17.57 |
| ran13x13 | 135 | 2939.83 | 3252.00 | 5.80 | 138 | 2950.64 | 3252.00 | **4.65** | 135 | 2939.83 | 3252.00 | 5.70 |
| set1ch | 403 | 52363.40 | 54537.75 | 1.36% | 397 | 54512.85 | 54537.75 | 0.02 | 521 | 54133.44 | 54537.75 | 0.10 |
| tr12-30 | 795 | 116031.84 | 132343.00 | 11.52% | 813 | 129375.97 | 130596.00 | 0.38% | 898 | 127092.01 | 130817.00 | 2.41% |
| vpm1 | 102 | 19.00 | 20.00 | **0.01** | 56 | 20.00 | 20.00 | **0.01** | 102 | 19.00 | 20.00 | **0.01** |
| vpm2 | 124 | 12.60 | 13.75 | 1.03 | 130 | 12.81 | 13.75 | 0.77 | 124 | 12.60 | 13.75 | 1.03 |

Table 6: Results for a 2 hour time limit, all other cuts on default. The column *min / %* shows the time to solve the problem instance or the duality gap after 2 hours. See the next page for uPMC and cPMC.

| | uPMC | | | | cPMC | | | |
|---|---|---|---|---|---|---|---|---|
| NAME | cuts | xLP | IP | min / % | cuts | xLP | IP | min / % |
| aflow30a | 151 | 1062.00 | 1158.00 | **1.06** | 120 | 1061.00 | 1158.00 | 1.26 |
| aflow40b | 187 | 1070.00 | 1246.00 | 10.67% | 179 | 1073.00 | 1229.00 | 9.36% |
| bc1 | 30 | 2.36 | 3.35 | 6.58% | 30 | 2.36 | 3.35 | 6.58% |
| bienst1 | 86 | 14.06 | 46.75 | **2.61** | 86 | 14.06 | 46.75 | **2.61** |
| bienst2 | 114 | 14.93 | 54.60 | 21.06 | 113 | 14.93 | 54.60 | 17.96 |
| dano3mip | 461 | 576.52 | 779.11 | 25.99% | 459 | 576.52 | 727.89 | 20.78% |
| dcmulti | 131 | 187254.13 | 188182.00 | **0.01** | 132 | 187347.01 | 188182.00 | **0.01** |
| fiber | 90 | 383141.97 | 405935.18 | 0.04 | 90 | 383141.97 | 405935.18 | **0.02** |
| fixnet6 | 97 | 3661.17 | 3983.00 | 0.04 | 97 | 3661.17 | 3983.00 | 0.04 |
| mod011 | 644 | -57199255.69 | -54558535.01 | 1.74 | 657 | -57139468.33 | -54558535.01 | 1.83 |
| modglob | 132 | 20725386.86 | 20740508.09 | **0.01** | 133 | 20725933.53 | 20740508.09 | **0.01** |
| pp08a | 144 | 7157.19 | 7350.00 | 0.04 | 138 | 7126.60 | 7350.00 | **0.03** |
| pp08aCUTS | 94 | 7112.45 | 7350.00 | **0.04** | 113 | 7068.32 | 7350.00 | **0.04** |
| ran10x26 | 133 | 4030.55 | 4270.00 | 5.08 | 131 | 4028.92 | 4270.00 | 4.85 |
| ran12x21 | 198 | 3394.44 | 3664.00 | 17.70 | 198 | 3394.44 | 3664.00 | 17.66 |
| ran13x13 | 135 | 2939.83 | 3252.00 | 5.70 | 130 | 2946.67 | 3252.00 | 6.24 |
| set1ch | 393 | 54465.02 | 54537.75 | 0.04 | 392 | 54404.17 | 54537.75 | 0.03 |
| tr12-30 | 746 | 129797.86 | 130596.00 | **0.30%** | 743 | 129672.88 | 130596.00 | 0.35% |
| vpm1 | 102 | 19.00 | 20.00 | **0.01** | 88 | 19.00 | 20.00 | **0.01** |
| vpm2 | 130 | 12.59 | 13.75 | **0.74** | 134 | 12.62 | 13.75 | 0.81 |

Table 7: Results for a 2 hour time limit, all other cuts on default. The column *min / %* shows the time to solve the problem instance or the duality gap after 2 hours. See the previous page for MIR, MIR+ and FP.