Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

GERALD GAMRATH, THORSTEN KOCH, ALEXANDER MARTIN*,
MATTHIAS MILTENBERGER, DIETER WENINGER*

Progress in Presolving for Mixed Integer Programming

* FAU Erlangen-Nürnberg, Cauerstr. 11, 91058 Erlangen, Germany, {alexander.martin, dieter.weninger}@math.uni-erlangen.de

ZIB-Report 13-48 (August 2013)

Herausgegeben vom
Konrad-Zuse-Zentrum für Informationstechnik Berlin
Takustraße 7
D-14195 Berlin-Dahlem

Telefon: 030-84185-0
Telefax: 030-84185-125

e-mail: bibliothek@zib.de
URL: http://www.zib.de

ZIB-Report (Print) ISSN 1438-0064
ZIB-Report (Internet) ISSN 2192-7782

Progress in Presolving for Mixed Integer Programming

Gerald Gamrath · Thorsten Koch · Alexander Martin

Matthias Miltenberger · Dieter Weninger

Abstract

Presolving attempts to eliminate redundant information from the problem formulation and simultaneously tries to strengthen the formulation. It can be very effective and is often essential for solving instances. Especially for mixed integer programming problems, fast and effective presolving algorithms are very important. In this paper, we report on three new presolving techniques. The first method searches for singleton continuous columns and tries to fix the corresponding variables. Then we present a presolving technique which exploits a partial order of the variables to induce fixings. Finally, we show an approach based on connected components in graphs. Our computational results confirm the profitable use of the algorithms in practice.

1 Introduction

In order to eliminate redundant information and to strengthen the formulation of an integer program, solvers apply a number of techniques before the linear programming relaxation of an instance is solved. This first step is referred to as presolving or preprocessing. The solvers then work with this reduced formulation rather than the original and recover the values of original variables afterwards. Presolving techniques are not only applied before solving the linear programming relaxation at the root node in a branch-and-bound tree, a reduced form called node presolving is also performed at all other nodes of the tree.

Presolving has been applied for solving linear and mixed integer programming problems for decades. Brearly et al. [15] and Williams [28] discussed bound tightening, row elimination, and variable fixings in mathematical programming systems, while Andersen and Andersen [6] published presolving techniques in the context of linear programming. In addition, presolving techniques on zero-one inequalities have been studied by Guignard and Spielberg [19], Johnson and Suhl [22], Crowder et al. [16], and Hoffman and Padberg [20]. Williams [29] pointed out a projection method for the elimination of integer variables and Savelsbergh [26] investigated preprocessing and probing techniques for mixed integer programming problems. An overview of different presolving techniques can be found in the books of Nemhauser [25] and Wolsey [30], in Fügenschuh and Martin [18] as well as Mahajan [24]. Details on implementing presolving techniques effectively within a mixed integer linear programming solver are discussed in Suhl and Szymanski [27], Atamtürk and Savelsbergh [8] and Achterberg [2].

Gerald Gamrath, Thorsten Koch, Matthias Miltenberger: {gamrath, koch, miltenberger}@zib.de,
Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany
Alexander Martin, Dieter Weninger: {alexander.martin, dieter.weninger}@math.uni-erlangen.de,
FAU Erlangen-Nürnberg, Cauerstr. 11, 91058 Erlangen, Germany

The impact of presolving on the entire solution process of mixed integer linear problems was published in Bixby and Rothberg [12]. By disabling root presolving, a mean performance degradation of about a factor of ten was detected. Only cutting planes had an even bigger influence on the solving process. This motivated us to look for further profitable presolving algorithms.

The paper is organized as follows. In Section 2 the notation is presented. Section 3 describes a presolving technique we call *Stuffing Singleton Columns*, where continuous variables with only one non-zero coefficient in the coefficient matrix are tried to be fixed at a suitable bound. In Section 4, we show another column based method called *Dominating Columns* working on a partial order. Through this relation, a consecutive behavior of the variable values arises of which fixings and bounds can be derived. Then, in Section 5 a technique based on *Connected Components* is presented. Such an approach is obvious, but was to the best of our knowledge not yet published in the context of presolving. In Section 6 we show computational results for all three illustrated presolving techniques with SCIP [3] on MIPLIB [11, 4, 23] and supply chain management instances and close with our conclusions in Section 7.

2 Notation and Basics

Consider a mixed integer program (*MIP*) in the following form:

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & Ax \leq b \\ & 0 \leq \ell \leq x \leq u \\ & x \in \mathbb{Z}^p \times \mathbb{R}^{n-p} \end{array} \tag{1}$$

with $c \in \mathbb{R}^n, \ell \in \mathbb{R}^n_+, u \in \mathbb{R}^n_+, A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m$ and $p \in \{0, 1, \ldots, n\}$.

We will use the notation $A_{\cdot j}$ to select the entire column j of the matrix A. Accordingly, $A_{i\cdot}$ extracts all coefficients of row i.

For a vector $x \in \mathbb{R}^n$ we call $\operatorname{supp}(x) = \{i \in \{1, 2, \ldots, n\} \mid x_i \neq 0\}$ the *support* of x.

In [15] a procedure for tightening bounds of variables can be found. Fundamental are the *maximal* (2) and *minimal* (3) *activity* of a linear constraint $a_r^T x$.

$$U_r = \sum_{\forall k,\, a_{rk} > 0} a_{rk} u_k + \sum_{\forall k,\, a_{rk} < 0} a_{rk} \ell_k \tag{2}$$

$$L_r = \sum_{\forall k,\, a_{rk} > 0} a_{rk} \ell_k + \sum_{\forall k,\, a_{rk} < 0} a_{rk} u_k \tag{3}$$

L_r may be $-\infty$ and U_r may be ∞. Obviously $L_r \leq a_r^T x \leq U_r$ is satisfied. Using the minimal activity L_r, it is possible to calculate new upper and lower bounds u_j^* and ℓ_j^* for variable x_j. For all feasible solutions x it holds that

$$x_j \leq \frac{b_r - L_r + a_{rj}\ell_j}{a_{rj}} = u'_{rj}, \forall r\ a_{rj} > 0 \tag{4}$$

$$x_j \geq \frac{b_r - L_r + a_{rj} u_j}{a_{rj}} = \ell'_{rj}, \forall r\ a_{rj} < 0 \tag{5}$$

Thus we obtain potentially new bounds by

$$\begin{aligned} u_j^* &= \min\{u_j, \min_{\forall r\ a_{rj} > 0}\{u'_{rj}\}\} \\ \ell_j^* &= \max\{\ell_j, \max_{\forall r\ a_{rj} < 0}\{\ell'_{rj}\}\} \end{aligned}$$

For integer variables we may also apply rounding

$$u_j^* = \min\{u_j, \min_{\forall r\ a_{rj}>0}\{\lfloor u'_{rj} \rfloor\}\} \quad \text{or} \quad \ell_j^* = \max\{\ell_j, \max_{\forall r\ a_{rj}<0}\{\lceil \ell'_{rj} \rceil\}\}.$$

3 Stuffing Singleton Columns

A singleton column is a column of the matrix A with $|\text{supp}\,(A_{\cdot j})| = 1$. The presolving technique presented in this section works through a set of singleton columns of continuous variables x_j within a row r and tries to fix them at the relevant bound.

As an example, consider the continuous knapsack problem:

$$\begin{array}{ll} \max & c^T x \\ \text{s.t.} & a^T x \leq b \\ & 0 \leq x_j \leq 1, j = 1, \ldots, n \\ & x \in \mathbb{R}^n_+, c \in \mathbb{R}^n_+, a \in \mathbb{R}^n_+, b \in \mathbb{R}_+ \end{array}$$

Here, we are not forced to pack an item completely or not at all like in the binary knapsack problem, because it is possible to pack any fraction between 0 and 1 of an item. Suppose that we first sort the items monotonically decreasing by their ratio c_j/a_j. Let t be the greatest index such that $\sum_{j=1}^{t} a_j \leq b$. We pack all items $x_1, \ldots, x_t$ entirely, fill the left capacity with $x_{t+1} = (b - \sum_{j=1}^{t} a_j)/a_{t+1}$ and all other items $x_{t+2}, \ldots, x_n$ stay completely outside of the knapsack. Obviously, this gives us an optimal solution without solving a linear program.

Transferring the above idea to (1) evokes two difficulties. Integer variables are present and variables usually appear in more than one row. So we cannot simply proceed like in the continuous knapsack problem. To solve the difficulties we act as follows. We are only interested in the case where $c_j/a_{rj} < 0$, because *duality fixing* [18] already covers the case $c_j/a_{rj} \geq 0$. In contrast to duality fixing, we use additional information about the rows. In the following, we will focus on the case $a_{rj} > 0$ and $c_j < 0$. For a given row r, the set of variables to be considered are:

$$J(r) = \{j \in \{1, \ldots, n\} \mid x_j \in \mathbb{R} \wedge |\text{supp}\,(A_{\cdot j})| = 1 \wedge a_{rj} > 0 \wedge c_j < 0\}.$$

Furthermore, we use the following two activities, which are similar to the maximal (2) and minimal activity (3) of row r except that continuous singleton columns x_j with $j \in J(r)$ are considered at their lower bounds.

$$\begin{aligned} \tilde{U}_r &= \sum_{\substack{j \in J(r) \\ \ell_j > 0}} a_{rj}\ell_j + \sum_{\substack{j \notin J(r) \\ a_{rj} > 0}} a_{rj}u_j + \sum_{\substack{j \notin J(r) \\ a_{rj} < 0}} a_{rj}\ell_j \\ \tilde{L}_r &= \sum_{\substack{j \in J(r) \\ \ell_j > 0}} a_{rj}\ell_j + \sum_{\substack{j \notin J(r) \\ a_{rj} < 0}} a_{rj}u_j + \sum_{\substack{j \notin J(r) \\ a_{rj} > 0}} a_{rj}\ell_j \end{aligned}$$

As input parameters of the algorithm there are $J(r), \tilde{L}_r, \tilde{U}_r$, the variables x, the lower bounds ℓ, the upper bounds u, the coefficients of row r with $a_r = A_{r\cdot}$, the corresponding right-hand side b_r and the coefficients of the objective function c. Output are fixings of variables x_j with $j \in J(r)$. First, we sort the ratios $c_j/a_{rj} < 0$ and start with the smallest ratio. If $\alpha \leq b_r - \tilde{U}_r + \beta$ is fulfilled ($\alpha, \beta \in \mathbb{R}$ as defined in Algorithm 1 lines 3 and 4), there is enough room for setting x_j to the upper bound and the value of the objective function improves because $c_j < 0$. If $b_r \leq \tilde{L}_r$ is satisfied, there is not sufficient space left and we can fix x_j at its lower bound.

Algorithm 1 Stuffing Singleton Columns for $a_{rj} > 0$ and $c_j < 0$

Input: $J(r), \tilde{L}_r, \tilde{U}_r, x, \ell, u, a_r, b_r, c$
Output: Fixings of variables x_j with $j \in J(r)$

Determine a sorted index list s of $J(r)$ such that $\frac{c_{s_1}}{a_{rs_1}} \leq \ldots \leq \frac{c_{|s|}}{a_{r|s|}}$
for all $k = 1, \ldots, |s|$ **do**
  $\alpha = a_{rs_k} \cdot u_{s_k}$
  $\beta = a_{rs_k} \cdot \ell_{s_k}$
  if $\alpha \leq b_r - \tilde{U}_r + \beta$ **then**
    $x_{s_k} = u_{s_k}$
  else if $b_r \leq \tilde{L}_r$ **then**
    $x_{s_k} = \ell_{s_k}$
  end if
  $\tilde{L}_r = \tilde{L}_r + \alpha - \beta$
  $\tilde{U}_r = \tilde{U}_r + \alpha - \beta$
end for

The algorithm with $a_{rj} < 0$ and $c_j > 0$ is working similarly. This time we do not try to fix as much variables as possible at the upper bound to achieve a better objective function. Now the point is to ensure feasibility while deteriorating the objective function as little as possible. Therefore we begin with the greatest ratio. $\alpha \geq b_r - \tilde{L}_r + \beta$ indicates that we need x_{s_j} at its upper bound even if the objective function is getting worse. If $b_r \geq \tilde{U}_r$ is fulfilled we fix the corresponding variable at the lower bound.

Both algorithms can be implemented to run very fast. Hence, they do not impact the performance on instances, e.g. from the MIPLIB, where only few reductions are found. In practical problems, such as supply chain management, stuffing singleton columns may, however, find fixings quite often (see Section 6).

Finally, it should be mentioned that singleton continuous columns sometimes deliver valuable dual information for linear programming problems (see [15]). This information can also be of great interest for (1). For example, it is possible to use this dual information in conjunction with complementary slackness to fix continuous variables at the lower bound in the primal problem, as described in [10]. Ideally, one tries to exploit this information first and then apply stuffing.

4 Dominating Columns

This presolving technique is based on a relation between two variables. We first introduce the corresponding relation and present an important property of it. After that, we show criteria that often allow to derive better bounds on variables or fix variables at one bound.

4.1 Dominance Relation

Definition 1. *Let* (1) *and two variables* x_j *and* x_i *be given. In addition both variables are of the same type, i.e. binary, integer or continuous. We say* x_j *dominates* x_i *(*$x_j \succ x_i$*), if*

(i) $c_j \leq c_i$,

(ii) $a_{rj} \leq a_{ri}$ *for every constraint* r.

We call x_j *the dominating variable and* x_i *the dominated variable.*

Definition 1 is a reflexive, antisymmetric and transitive relation on the coefficients of the variables and therefore a *partial order* (poset). Because by this relation a consecutive behavior of the variable values arises, it can also be seen as a dominance relation.

The idea of exploiting a kind of a dominance relation between variables for presolving is not new. Andersen and Andersen [6] used dominating columns for presolving of linear programming problems and Borndörfer [14] in the context of set partitioning problems. In addition, Babayev and Mardanov [9] and Zhu and Broughan [32] introduced procedures based on comparing pairs of columns for reducing the number of integer variables mostly applied on knapsack problems. Our method can be seen as a generalization and extension of existing dominating columns approaches for mixed integer programming problems. In combination with a special bound analysis we are not only able to fix variables to zero, but can also fix variables at an arbitrarily lower or upper bound if certain properties are fulfilled.

The following two examples illustrate Definition 1.

Example 1.

$$\begin{array}{lrcrcrcrcl} \mathit{min} & -2x_1 & - & x_2 & - & 2x_3 & - & 4x_4 & & \\ \mathit{s.t.} & 2x_1 & + & 3x_2 & + & x_3 & - & x_4 & \leq & 6 \\ & & & x_2 & + & 3x_3 & - & x_4 & \leq & 1 \\ & & & x_2 & & & + & 2x_4 & \leq & 3 \end{array}$$

$$\begin{array}{c} 0 \leq x_1, x_2 \leq 4, \;\; 0 \leq x_4 \leq 2 \\ x_1, x_2 \in \mathbb{Z}, \;\; x_3 \in \{0,1\}, \;\; x_4 \in \mathbb{R}. \end{array}$$

It holds $x_1 \succ x_2$ *and the optimal solution is* $x_1 = 3, x_2 = 0, x_3 = 0, x_4 = \frac{3}{2}$ *with optimal value* -12.

Example 2.

$$\begin{array}{lrcrcrcrcr} \mathit{min} & x_1 & + & x_2 & - & 2x_3 & - & 3x_4 & & \\ \mathit{s.t.} & -2x_1 & - & x_2 & + & 2x_3 & - & 2x_4 & \leq & -12 \\ & & & & & x_3 & + & x_4 & \leq & \frac{3}{2} \\ & -x_1 & - & x_2 & - & 2x_3 & & & \leq & -6 \end{array}$$

$$\begin{array}{c} 0 \leq x_1, x_2 \leq 4, \;\; 0 \leq x_4 \leq 2 \\ x_1, x_2 \in \mathbb{Z}, \;\; x_3 \in \{0,1\}, \;\; x_4 \in \mathbb{R}. \end{array}$$

Again $x_1 \succ x_2$ *and the optimal solution is* $x_1 = 4, x_2 = 2, x_3 = 0, x_4 = \frac{3}{2}$ *with optimal value* $\frac{3}{2}$.

In both examples, one of the variables involved in the dominance relation is at one of its bounds in the optimal solution. This is a general property of the dominance relation that we will prove in the following. In order to do that, we first show that increasing the dominating variable and decreasing the dominated variable by the same amount preserves feasibility and optimality as long as the variable bounds are still satisfied.

Lemma 1. *Let* $\bar{x}$ *be a feasible solution for* (1) *and* $x_j \succ x_i$. *For* $0 < \alpha \in \mathbb{R}$, *we define* $x^\star$ *with*

$$x_k^\star = \begin{cases} \bar{x}_k + \alpha & k = j, \\ \bar{x}_k - \alpha & k = i, \\ \bar{x}_k & \mathit{else} \end{cases}$$

If $x_j^\star = \bar{x}_j + \alpha \leq u_j$ *and* $x_i^\star = \bar{x}_i - \alpha \geq \ell_i$, *then* $x^\star$ *is feasible and its objective value is not worse than the one of* $\bar{x}$.

Proof. For every constraint $a_r^T x \leq b_r$, we get

$$\begin{aligned}\sum_{k=1}^{n} a_{rk}x_k^\star &= \sum_{\substack{k=1\\k\neq i,j}}^{n} a_{rk}\bar{x}_k + a_{rj}(\bar{x}_j+\alpha) + a_{ri}(\bar{x}_i-\alpha)\\ &= \underbrace{\sum_{\substack{k=1\\k\neq i,j}}^{n} a_{rk}\bar{x}_k}_{\leq b_r} + \alpha\underbrace{(a_{rj}-a_{ri})}_{\leq 0} \leq b_r.\end{aligned}$$

Obviously, also the bounds of the variables are fulfilled, hence $x^\star$ is feasible. Additionally, we know from Definition 1 that $c_j \leq c_i$, thus $c^T x^\star = c^T\bar{x} + \alpha(c_j - c_i) \leq c^T\bar{x}$, i.e., the objective value is not getting worse. □

This leads us to the following theorem which states that the dominated variable is at its lower bound or the dominating variable is at its upper bound in at least one optimal solution.

Theorem 1. *Let $x_j \succ x_i$, then there always exists an optimal solution $x^\star$ with*

$$x_j^\star = u_j \quad \vee \quad x_i^\star = \ell_i.$$

Proof. Let $\bar{x}$ be an optimal solution with $\bar{x}_j < u_j \wedge \bar{x}_i > \ell_i$. We construct a feasible solution $x^\star$ with $c^T x^\star \leq c^T\bar{x}$ by defining $\alpha = \min\{\bar{x}_i - \ell_i, u_j - \bar{x}_j\}$ and applying Lemma 1. Since $\bar{x}$ is optimal, $c^T x^\star = c^T\bar{x}$ and $x^\star$ is optimal, too. By the definition of α, also $x_j^\star = u_j \vee x_i^\star = \ell_i$ holds. □

4.2 Predictive Bound Analysis

Based on Theorem 1 we will now describe sufficient conditions which allow in combination with Definition 1 to tighten bounds or fix variables. We first extend the maximal and minimal row activity from (2) and (3) as a function in one variable x_t.

Definition 2. *Let a linear constraint $a_r^T x \leq b_r$ and a variable $x_t = \xi$ be given. We denote by*

$$U_r^t(\xi) = \sum_{\substack{k=1\\k\neq t\\a_{rk}>0}}^{n} a_{rk}u_k + \sum_{\substack{k=1\\k\neq t\\a_{rk}<0}}^{n} a_{rk}\ell_k + a_{rt}\xi$$

the conditional maximal activity of the linear constraint w.r.t. ξ and by

$$L_r^t(\xi) = \sum_{\substack{k=1\\k\neq t\\a_{rk}>0}}^{n} a_{rk}\ell_k + \sum_{\substack{k=1\\k\neq t\\a_{rk}<0}}^{n} a_{rk}u_k + a_{rt}\xi$$

the conditional minimal activity of the linear constraint w.r.t. ξ.

In case the sum is not well-defined because both positive as well as negative infinite contributions occur, we define $U_r^t(\xi) = +\infty$ and $L_r^t(\xi) = -\infty$. These cases will be excluded in the criteria for changing bounds or fixing variables anyway.

Definition 2 will now be used to define specific functions, which predict in dependence of the value ξ of one variable x_t the bound of another variable. We call this approach predictive bound analysis.

Definition 3. *Let* (1) *and two variables* x_s *and* $x_t = \xi$ *be given. We define the following functions:*

$$\begin{aligned}
\textsc{maxl}_s^t(\xi) &= \max_{r=1,\dots,m} \left\{ \frac{b_r - L_r^t(\xi) + a_{rs}u_s}{a_{rs}} \mid a_{rs}, a_{rt} < 0 \right\} \\
\textsc{maxu}_s^t(\xi) &= \max_{r=1,\dots,m} \left\{ \frac{b_r - U_r^t(\xi) + a_{rs}\ell_s}{a_{rs}} \mid a_{rs}, a_{rt} < 0 \right\} \\
\textsc{minl}_s^t(\xi) &= \min_{r=1,\dots,m} \left\{ \frac{b_r - L_r^t(\xi) + a_{rs}\ell_s}{a_{rs}} \mid a_{rs}, a_{rt} > 0 \right\} \\
\textsc{minu}_s^t(\xi) &= \min_{r=1,\dots,m} \left\{ \frac{b_r - U_r^t(\xi) + a_{rs}u_s}{a_{rs}} \mid a_{rs}, a_{rt} > 0 \right\}
\end{aligned}$$

Let us have a look at the meaning of these values: $\textsc{minl}_s^t(\xi)$ takes into account all constraints in which x_s and x_t have positive coefficients, i.e., a subset of the constraints that imply an upper bound on x_s. Similar to the bound tightening (see (4)), the upper bound on x_s is computed for each constraint, but instead of using the minimal activity, the conditional minimal activity w.r.t. $x_t = \xi$ is used. Therefore, each constraint gives an upper bound for x_s subject to the value of x_t and minimizing over these bounds, $\textsc{minl}_s^t(\xi)$ gives the tightest implied upper bound on x_s as a function of the value ξ of x_t. Analogously, $\textsc{maxl}_s^t(\xi)$ gives the tightest implied lower bound on x_s as a function of the value ξ of x_t. The other two functions $\textsc{maxu}_s^t(\xi)$ and $\textsc{minu}_s^t(\xi)$ take into account the maximal instead of the minimal activity. It follows that the difference between the right-hand side and residual activity subject to $x_t = \xi$ is minimized. This way we may get a larger lower bound and a smaller upper bound on x_s. Since the maximal activity is the worst-case when regarding feasibility of a $\leq$-constraint, all values of x_s which are larger than $\textsc{maxu}_s^t(\xi)$ or smaller than $\textsc{minu}_s^t(\xi)$ can never lead to an infeasibility.

Next, we show that these four functions are strictly monotonically decreasing. This property is fundamental to obtain a maximum value if we assume x_t at its lower bound and vice versa.

Lemma 2. $\textsc{maxl}_s^t(\xi)$, $\textsc{maxu}_s^t(\xi)$, $\textsc{minl}_s^t(\xi)$ *and* $\textsc{minu}_s^t(\xi)$ *are strictly monotonically decreasing functions, i.e., for* $\ell_t \leq \xi' < \xi'' \leq u_t$ *holds*

$$\begin{aligned}
\textsc{maxl}_s^t(\xi') &> \textsc{maxl}_s^t(\xi''), \\
\textsc{maxu}_s^t(\xi') &> \textsc{maxu}_s^t(\xi''), \\
\textsc{minl}_s^t(\xi') &> \textsc{minl}_s^t(\xi'') \text{ and} \\
\textsc{minu}_s^t(\xi') &> \textsc{minu}_s^t(\xi'').
\end{aligned}$$

Proof. We only prove the first inequality, the others can be shown analogously. Let $\tilde{r}$ be one row defining the maximum in the computation of $\textsc{maxl}_s^t(\xi'')$. Since $L_{\tilde{r}}^t(\xi'') - L_{\tilde{r}}^t(\xi') = a_{\tilde{r}t}(\xi'' - \xi')$

and $a_{\tilde{r}s}, a_{\tilde{r}t} < 0$ by the definition of $\text{MAXL}_s^t(\xi)$, the following holds:

$$\begin{aligned}
\text{MAXL}_s^t(\xi') - \text{MAXL}_s^t(\xi'') &= \max_{r=1,\dots,m}\left\{\frac{b_r - L_r^t(\xi') + a_{rs}u_s}{a_{rs}}\middle| a_{rs} < 0\right\} \\
&\quad - \max_{r=1,\dots,m}\left\{\frac{b_r - L_r^t(\xi'') + a_{rs}u_s}{a_{rs}}\middle| a_{rs} < 0\right\} \\
&\geq \frac{b_{\tilde{r}} - L_{\tilde{r}}^t(\xi') + a_{\tilde{r}s}u_s}{a_{\tilde{r}s}} - \frac{b_{\tilde{r}} - L_{\tilde{r}}^t(\xi'') + a_{\tilde{r}s}u_s}{a_{\tilde{r}s}} \\
&= \frac{a_{\tilde{r}t}}{a_{\tilde{r}s}}(\xi' - \xi'') \\
&> 0
\end{aligned}$$

□

These functions can help us to infer bounds for the dominating or the dominated variable in an optimal solution.

Theorem 2. *Let $x_j \succ x_i$. Then the following holds for at least one optimal solution.*

(i) $x_j \leq \text{MINL}_j^i(\ell_i)$.

(ii) $x_i \geq \text{MAXL}_i^j(u_j)$.

(iii) $\left|\text{MAXL}_j^i(\ell_i)\right| < \infty \Rightarrow x_j \geq \min\{u_j, \text{MAXL}_j^i(\ell_i)\}$.

(iv) $\left|\text{MINL}_i^j(u_j)\right| < \infty \Rightarrow x_i \leq \max\{\ell_i, \text{MINL}_i^j(u_j)\}$.

(v) $\left|\text{MINU}_j^i(\ell_i)\right| < \infty$ *and* $c_j \leq 0 \Rightarrow x_j \geq \min\{u_j, \text{MINU}_j^i(\ell_i)\}$.

(vi) $\left|\text{MAXU}_i^j(u_j)\right| < \infty$ *and* $c_i \geq 0 \Rightarrow x_i \leq \max\{\ell_i, \text{MAXU}_i^j(u_j)\}$.

Proof.

(i) Follows from (4), since a_{ri} is positive for all rows regarded for the computation of $\text{MINL}_j^i(\ell_i)$ and therefore setting x_i to ℓ_i does not change the minimal activity of the row.

(ii) Follows from (5), since a_{rj} is negative for all rows regarded for the computation of $\text{MAXL}_i^j(u_j)$ and therefore setting x_j to u_j does not change the minimal activity of the row.

(iii) By Definition 3, there exists one row r with $a_{rj}\text{MAXL}_j^i(\ell_i) + L_r^i(\ell_i) - a_{rj}u_j = b_r$. Suppose there is an optimal solution $x^\star$ with $\alpha = \min\{u_j, \text{MAXL}_j^i(\ell_i)\} - x_j^\star > 0$. From $x_j \succ x_i$ and Definition 3, we know $a_{rj} \leq a_{ri} < 0$, so for row r to be feasible, $x_i^\star > \ell_i + \alpha$ must hold. By Lemma 1, we can increase $x_j^\star$ by α and decrease $x_i^\star$ by α without loosing feasibility or optimality.

(iv) By Definition 3, there exists one row r with $a_{ri}\text{MINL}_i^j(u_j) + L_r^j(u_j) - a_{ri}\ell_i = b_r$. Suppose there is an optimal solution $x^\star$ with $\alpha = x_i^\star - \max\{\ell_i, \text{MINL}_i^j(u_j)\} > 0$. From $x_j \succ x_i$ and Definition 3, we know $0 < a_{rj} \leq a_{ri}$, so for row r to be feasible, $x_j^\star < u_j - \alpha$ must hold. By Lemma 1, we can decrease $x_i^\star$ by α and increase $x_j^\star$ by α without loosing feasibility or optimality.

(v) By Definition 3, there exists one row r with $a_{rj}\text{MINU}_j^i(\ell_i) + U_r^i(\ell_i) - a_{rj}u_j = b_r$. Suppose there is an optimal solution $x^\star$ with $x_j^\star < \min\{u_j, \text{MINU}_j^i(\ell_i)\}$.

Let $\alpha_j = \min\{u_j, \text{MINU}_j^i(\ell_i)\} - x_j^\star$, $\alpha_i = x_i^\star - \ell_i$, and $\alpha = \min\{\alpha_i, \alpha_j\}$. By Lemma 1, we can increase $x_j^\star$ by α and decrease $x_i^\star$ by α without loosing feasibility or optimality. If $\alpha = \alpha_j$, then we are finished because we constructed an optimal solution with $x_j = \min\{u_j, \text{MINU}_j^i(\ell_i)\}$. Otherwise, we get an optimal solution $x^\star$ with $x_i^\star = \ell_i$. Now, we show that $\bar{x}$ with $\bar{x}_j = \min\{u_j, \text{MINU}_j^i(\ell_i)\}$ and $\bar{x}_k = x_k^\star$ for $k \neq j$ is also an optimal solution. Because $x^\star$ is feasible and by definition of $\bar{x}_j$, $\bar{x}$ fulfills all bounds. By increasing x_j, we can only loose feasibility for rows r with $a_{rj} > 0$. From $x_j \succ x_i$ we know $0 < a_{rj} \leq a_{ri}$, so these rows are exactly the rows regarded in the definition of $\text{MINU}_j^i(\ell_i)$. Assume one of these rows is violated, i.e., $a_r^T\bar{x} > b_r$, then

$$\begin{aligned}
0 &> b_r - \sum_{k=1}^{n} a_{rk}\bar{x}_k \\
&= b_r - \left(\sum_{\substack{k=1 \\ k\neq i \\ a_{rk}>0}}^{n} a_{rk}\bar{x}_k + \sum_{\substack{k=1 \\ k\neq i \\ a_{rk}<0}}^{n} a_{rk}\bar{x}_k + a_{ri}\ell_i \right) \\
&\geq b_r - \left(\sum_{\substack{k=1 \\ k\neq i \\ a_{rk}>0}}^{n} a_{rk}u_k + \sum_{\substack{k=1 \\ k\neq i \\ a_{rk}<0}}^{n} a_{rk}\ell_k + a_{ri}\ell_i - a_{rj}u_j + a_{rj}\bar{x}_j \right) \\
&= b_r - U_r^i(\ell_i) + a_{rj}u_j - a_{rj}\bar{x}_j
\end{aligned}$$

It follows that $\bar{x}_j > (b_r - U_r^i(\ell_i) + a_{rj}u_j)/a_{rj} \geq \text{MINU}_j^i(\ell_i)$, but this contradicts the definition of $\bar{x}_j$, so all rows must still be feasible. $\bar{x}$ is also optimal since we get $c^T\bar{x} \leq c^Tx^\star$ from $\bar{x}_j > x_j^\star$ and $c_j \leq 0$.

(vi) By Definition 3, there exists one row r with $a_{ri}\text{MAXU}_i^j(u_j) + U_r^j(u_j) - a_{ri}\ell_i = b_r$. Suppose there is an optimal solution $x^\star$ with $x_i^\star > \max\{\ell_i, \text{MAXU}_i^j(u_j)\}$.

Let $\alpha_i = x_i^\star - \max\{\ell_i, \text{MAXU}_i^j(u_j)\}$, $\alpha_j = u_j - x_j^\star$, and $\alpha = \min\{\alpha_i, \alpha_j\}$. By Lemma 1, we can decrease $x_i^\star$ by α and increase $x_j^\star$ by α without loosing feasibility or optimality. If $\alpha = \alpha_i$, then we are finished because we constructed an optimal solution with $x_i = \max\{\ell_i, \text{MAXU}_i^j(u_j)\}$. Otherwise, we get an optimal solution $x^\star$ with $x_j^\star = u_j$. Now, we show that $\bar{x}$ with $\bar{x}_i = \max\{\ell_i, \text{MAXU}_i^j(u_j)\}$ and $\bar{x}_k = x_k^\star$ for $k \neq i$ is also an optimal solution. Because $x^\star$ is feasible and by definition of $\bar{x}_i$, $\bar{x}$ fulfills all bounds. By decreasing x_i, we can only loose feasibility for rows r with $a_{ri} < 0$. From $x_j \succ x_i$ we know $a_{rj} \leq a_{ri} < 0$, so these rows are exactly the rows regarded in the definition of $\text{MAXU}_i^j(u_j)$. Assume one of

these rows is violated, i.e., $a_r^T \bar{x} > b_r$, then

$$\begin{aligned}
0 &> b_r - \sum_{k=1}^{n} a_{rk}\bar{x}_k \\
&= b_r - \left(\sum_{\substack{k=1 \\ k\neq j \\ a_{rk}>0}}^{n} a_{rk}\bar{x}_k + \sum_{\substack{k=1 \\ k\neq j \\ a_{rk}<0}}^{n} a_{rk}\bar{x}_k + a_{rj}u_j \right) \\
&\geq b_r - \left(\sum_{\substack{k=1 \\ k\neq j \\ a_{rk}>0}}^{n} a_{rk}u_k + \sum_{\substack{k=1 \\ k\neq j \\ a_{rk}<0}}^{n} a_{rk}\ell_k + a_{rj}u_j - a_{ri}\ell_i + a_{ri}\bar{x}_i \right) \\
&= b_r - U_r^j(u_j) + a_{ri}\ell_i - a_{ri}\bar{x}_i
\end{aligned}$$

Since $a_{ri} < 0$, it follows that $\bar{x}_i < (b_r - U_r^j(u_j) + a_{ri}\ell_i)/a_{ri} \leq \text{MAXU}_i^j(u_j)$, but this contradicts the definition of $\bar{x}_i$, so all rows must still be feasible. $\bar{x}$ is also optimal since we get $c^T\bar{x} \leq c^T x^\star$ from $\bar{x}_i > x_i^\star$ and $c_i \geq 0$.

□

Whenever in Theorem 2, (iii) - (vi), the minimum or maximum is obtained for the first argument, the variable can be fixed. Since this has the highest impact regarding presolving as it reduces the problem size, we summarize the fixing criteria.

Corollary 1. *Let $x_j \succ x_i$. In the following cases, we can fix a variable while preserving at least one optimal solution.*

(i) $\infty > \text{MAXL}_j^i(\ell_i) \geq u_j \Rightarrow x_j$ *can be fixed to* u_j.

(ii) $-\infty < \text{MINL}_i^j(u_j) \leq \ell_i \Rightarrow x_i$ *can be fixed to* ℓ_i.

(iii) $c_j \leq 0$ *and* $\infty > \text{MINU}_j^i(\ell_i) \geq u_j \Rightarrow x_j$ *can be fixed to* u_j.

(iv) $c_i \geq 0$ *and* $-\infty < \text{MAXU}_i^j(u_j) \leq \ell_i \Rightarrow x_i$ *can be fixed to* ℓ_i.

These criteria rely on finite values for the predicted bounds. In particular, if the bound which is used for the computation of the conditional minimal or maximal activity is infinite, we typically get infinite predicted bounds. The following criteria are equivalent to the ones stated in Corollary 1 if $-\infty < \ell_i \leq x_i \leq u_i < \infty$ and $-\infty < \ell_j \leq x_j \leq u_j < \infty$, but use other bounds for the computation of conditional minimal and maximal activities, which can sometimes be beneficial in order to get finite conditional activities.

Corollary 2. *Let $x_j \succ x_i$. In the following cases, we can fix a variable while preserving at least one optimal solution.*

(i) $\infty > \text{MAXL}_i^j(u_j) \geq \ell_i \Rightarrow x_j$ *can be fixed to* u_j.

(ii) $-\infty < \text{MINL}_j^i(\ell_i) \leq u_j \Rightarrow x_i$ *can be fixed to* ℓ_i.

(iii) $c_j \leq 0$ *and* $\infty > \text{MINU}_i^j(u_j) \geq \ell_i \Rightarrow x_j$ *can be fixed to* u_j.

(iv) $c_i \geq 0$ *and* $-\infty < \text{MAXU}_j^i(\ell_i) \leq u_j \Rightarrow x_i$ *can be fixed to* ℓ_i.

Proof.

(i) If $\text{MAXL}_i^j(u_j) \geq \ell_i$, then by Definition 3 and Lemma 2 it follows that

$$\text{MAXL}_j^j(\ell_i) \geq \text{MAXL}_j^i(\text{MAXL}_i^j(u_j)) = u_j.$$

From $\text{MAXL}_i^j(u_j) < \ell_i$ follows

$$\text{MAXL}_j^i(\ell_i) < \text{MAXL}_j^i(\text{MAXL}_i^j(u_j)) = u_j.$$

This is the statement of Corollary 1(i).

(ii)-(iv) are similar to case (i). □

By having two alternative criteria for each variable fixing, we can select the one that fits better in a given situation. In particular, an infinite upper bound is more common than an infinite lower bound since problems are typically modeled using non-negative variables.

4.3 Utilize Conflict Information for Binary Variables

For binary variables we can use information from a conflict graph [7] for fixing additional variables in connection with the dominance relation. The use of this information has the advantage that it was concurrently extracted in preceding presolving rounds.

An undirected graph $G = (V, E)$ is called a *conflict graph* of (1), if for every binary variable x_i there are a vertex $v_i \in V$ and a vertex $\bar{v}_i \in V$ for its complement $\bar{x}_i = 1 - x_i$. The edge set E consists of edges $v_i\bar{v}_i$ for all binary variables x_i and edges between two vertices when at most one of the corresponding variables or complements can be equal to 1 in an optimal solution.

Theorem 3.

(i) Let $x_j \succ x_i$ *and* $v_jv_i \in E$*, then* x_i *can be fixed to 0.*

(ii) Let $x_j \succ x_i$ *and* $\bar{v}_j\bar{v}_i \in E$*, then* x_j *can be fixed to 1.*

Proof.

(i) With two binary variables, four variable assignments are possible. Because $x_j = 1 \wedge x_i = 1$ is not allowed, only the possibilities $x_j = 1 \wedge x_i = 0$, $x_j = 0 \wedge x_i = 0$ and $x_j = 0 \wedge x_i = 1$ remain. From Definition 1 and Lemma 1 we know that it is possible to increase x_j and decrease x_i accordingly, thereby staying feasible and optimal. Thus, only the cases $x_j = 1 \wedge x_i = 0$ and $x_j = 0 \wedge x_i = 0$ are remaining. In both cases, x_i is at its lower bound.

(ii) The case is similar to (i). Finally, the logical conjunctions $x_j = 1 \wedge x_i = 1$ and $x_j = 1 \wedge x_i = 0$ are left. In both cases, x_j is at its upper bound.

□

4.4 Finding a Dominance Relation

The complexity of an algorithm that operates on a partial order (poset) is mainly determined by the width. The width w of a poset is defined to be the maximum cardinality of an anti-chain, which is a subset of mutually incomparable elements. In [17] an algorithm was published that sorts a width-w poset of size n in $O(n(w + \log n))$. Their representation has size $O(wn)$ and permits retrieval of the relation between any two elements in time $O(1)$.

Despite the promising results in [17], we have opted for a different approach with a worse complexity because it is easy to implement and works well in practice. It consists of two stages. The first stage compares only variables which are present within equalities. This is done by an algorithm developed for detecting parallel rows or columns [13]. It is also possible to follow a procedure as in [5]. The second stage considers only those variables that have not yet been studied and takes advantage of the sparsity of A. This can be achieved by first sorting all rows by the number of non-zero coefficients. Then, we start with the row that contains the fewest non-zeros and compare only columns that have a non-zero entry in this row. After one row was executed, the processed variables therein are not compared to other variables anymore. In the worst case, in which the matrix is dense, there is a mechanism which monitors the number of fixings per ν paired comparisons. If no fixing by means of ν comparisons is found, then this row will not be further investigated. In the course ν is dynamically adjusted according to the number of found fixings. In practice, however, the matrices are usually sparse and some equalities are present, resulting in favorable operating times (see Section 6).

5 Connected Components

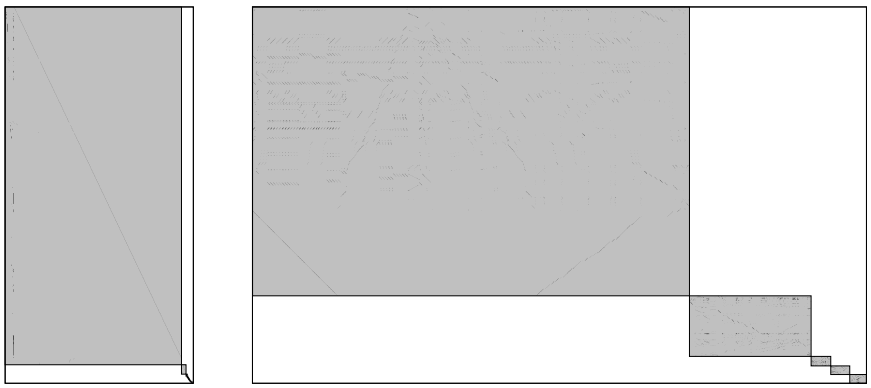
The connected components presolver aims at identifying small subproblems that are independent of the remaining part of the problem and tries to solve those to optimality during the presolving phase. After a component is solved to optimality, the variables and constraints forming the component can be removed from the remaining problem. This reduces the size of the problem and the linear program to be solved at each node.

Although a well modeled problem should in general not contain independent components, they occur regularly in practice. And even if a problem cannot be split into its components right from the beginning, it might decompose after some rounds of presolving, e.g., because constraints connecting independent problems are detected redundant and can be removed. Figure 1 depicts the constraint matrices of two real-world instances at some point in presolving, reordered in a way such that independent components can easily be identified.

We detect independent subproblems by first transferring the structure of the problem to an undirected graph G and then searching for connected components like in [21]. The graph G is constructed as follows: for every variable x_i, we create a node v_i, and for each constraint, we add edges to G connecting the variables with non-zero coefficients in the constraint. Thereby, we do not add an edge for each pair of these variables, but – in order to reduce the graph size – add a single path in the graph connecting all these variables. More formally, the graph is defined as follows: $G = (V, E)$ with

$$
\begin{aligned}
V &= \{v_i\}_{i=1,\dots,n} \\
E &= \bigcup_{k=1}^{m} \{(v_i, v_j) \mid 1 \le i < j \le n : \quad a_{k,i} \neq 0 \\
&\qquad\qquad\qquad\qquad\qquad \wedge \quad a_{k,j} \neq 0 \\
&\qquad\qquad\qquad\qquad\qquad \wedge \quad a_{k,\ell} = 0 \ \forall \ell \in \{i+1, \dots, j-1\}\}\,.
\end{aligned}
$$

Given this graph, we identify connected components using depth first search. By definition,



(a) tanglegram2 **(b)** scm-1-1

Figure 1: Matrix structures of one instance from MIPLIB 2010 and one supply chain management instance: columns and rows were permuted to visualize the block structure. Dots represent non-zero entries while gray rectangles represent the blocks, which are ordered by their size from top left to bottom right.

each constraint contains variables of only one component and can easily be assigned to the corresponding subproblem.

The size of the graph is linear in the number of variables and non-zeros. It has n nodes and – due to the representation of a constraint as a path – exactly $z - m$ edges[1], where z is the number of non-zeros in the constraint matrix. The connected components of a graph can be computed in linear time w.r.t. the number of nodes and edges of the graph [21], which is thus also linear in the number of variables and non-zeros of the MIP.

If we identify more than one subproblem, we try to solve the small ones immediately. In general, we would expect a better performance by solving all subproblems to optimality one after another rather than solving the complete original problem to optimality. However, this has the drawback that we do not compute valid primal and dual bounds until we start solving the last subproblem. In practical applications, we often do not need to find an optimal solution, but a time limit is applied or the solving process is stopped when a small optimality gap is reached. In this case, it is preferable to only solve easy components to optimality during presolving and solve remaining larger problems together, thereby computing valid dual and primal bounds for the complete problem.

To estimate the computational complexity of the components, we count the number of discrete variables. In case this number is larger then a specific amount we do not solve this particular component separately, to avoid spending too much time in this step. In particular, subproblems containing only continuous variables are always solved, despite their dimensions.

However, the number of discrete variables is not a reliable indicator for the complexity of a problem and the time needed to solve it to optimality.[2] Therefore, we also limit the number of branch-and-bound nodes for every single subproblem. If the node limit is hit, we merge the component back into the remaining problem and try to transfer as much information to the original problem as possible; however, most insight is typically lost. Therefore, it is important

[1] Assuming that no empty constraints exist; otherwise, the number of edges is still not larger than z.

[2] See, e.g., the markshare instances [1] contained in MIPLIB 2003 that are hard to solve for state-of-the-art solvers although having only 60 variables.

to choose the parameters in a way such that this scenario is avoided.

6 Computational Results

In this section, we present computational results that show the impact of the new presolving methods on the presolving performance as well as on the overall solution process.

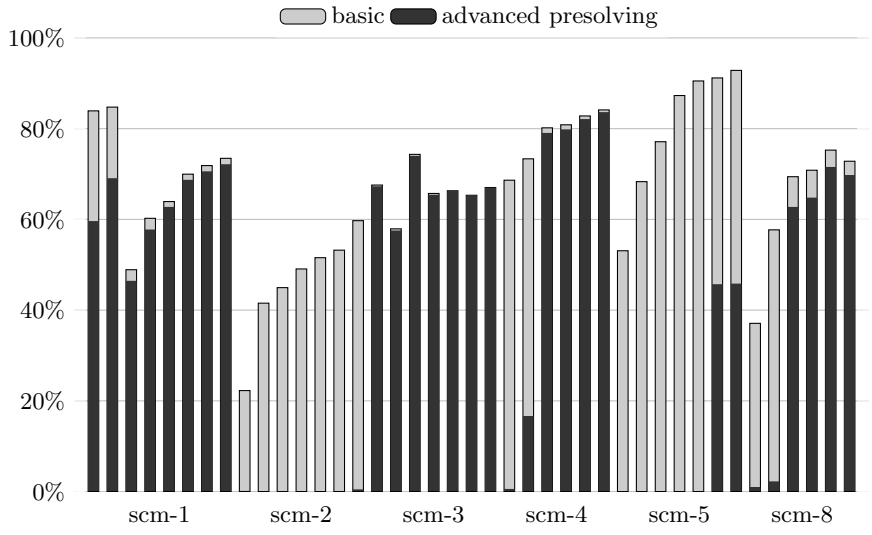
We implemented three new presolving techniques, which were already included in the SCIP 3.0 release. The stuffing algorithm is implemented within the dominated columns presolver, because it makes use of the same data structures. Since the new presolvers can be expensive, they are activated very conservatively compared to the remaining presolvers.

The experiments were performed on a cluster of Intel Xeon X5672 3.20 GHz computers, with 12 MB cache and 48 GB RAM, running Linux (in 64 bit mode). We used two different test sets: a set of real-world supply chain management instances provided by our industry partner and the MMM test set consisting of all instances from MIPLIB 3 [11], MIPLIB 2003 [4], and the benchmark set of MIPLIB 2010 [23]. For the experiments, we used the development version 3.0.1.2 of SCIP [3] (git hash `7e5af5b`) with SoPlex [31] version 1.7.0.4 (git hash `791a5cc`) as the underlying LP solver and a time limit of two hours per instance. In the following, we distinguish two versions of presolving: the *basic* and the *advanced* version. The basic version performs all the presolving steps implemented in SCIP (for more details, we refer to [2]), but disables the techniques newly introduced in this paper, which are included in the advanced presolving. This measures the impact of the new methods within an environment that already contains various presolving methods. Restarts were disabled to prevent further calls of presolvers during the solving process, thereby ensuring an unbiased comparison of the methods.

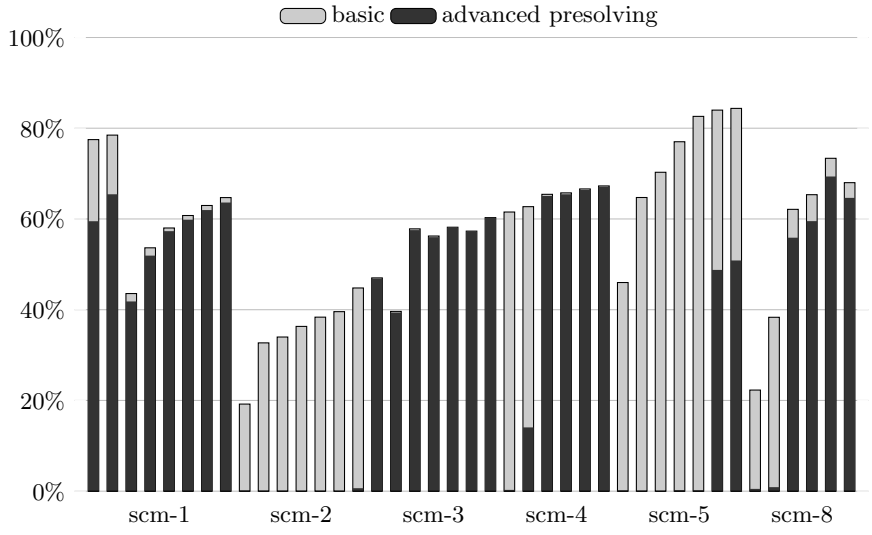
Figure 2 illustrates the presolve reductions for the supply chain management instances. For each of the instances, the percentage of remaining variables (Figure 2a) and remaining constraints (Figure 2b) after presolving is shown, both for the basic as well as the advanced presolving. While for every instance, the new presolving methods do some additional reductions, the amount of reductions varies heavily. On the one hand, only little additional reductions are found for the 1- and 3-series as well as parts of the 4-series, on the other hand, the size of some instances, in particular from the 2- and 5-series, is reduced to less than 1% of the original size. The reason for this is that these instances decompose into up to 1000 independent subproblems most of which the connected components presolver does easily solve to optimality during presolve. Average results including presolving and solving time are listed in Table 1, detailed instance-wise results can be found in Table 2 in Appendix A. This also includes statistics about the impact of the new presolvers. On average, the advanced presolving reduces the number of variables and constraints by about 59% and 64%, respectively, while the basic presolving only removes about 33% and 43%, respectively. The components presolver fixes on average about 18% of the variables and 16% of the constraints. 3.5% and 0.9% of the variables are fixed by dominating columns and stuffing, respectively. This increases the shifted geometric mean of the presolving time from 2.12 to 3.18 seconds, but pays off since the solving time can be reduced by almost 50%. For a definition and discussion of the shifted geometric mean, we refer to [2].

The structure of the supply chain management instances allows the new presolving methods to often find many reductions. This is different for the instances from the more general MMM test set, where on average, the advanced presolving removes about 3% more variables and 1% more constraints. It allows to solve one more instance within the time limit and reduces the solving time from 335 to 317 seconds in the shifted geometric mean. This slight improvement can also be registered in the performance diagram shown in Figure 3.

However, many of the instances in the MMM test set do not contain a structure that can be



(a) variables after presolve



(b) constraints after presolve

Figure 2: Size of the presolved supply chain management instances relative to the original number of variables and constraints.

used by the new presolving techniques: they are able to find reductions for less than every fourth instance. On the set of instances where no additional reductions are found, the time spent in presolving as well as the total time are almost the same, see row MMM:eq in Table 1. Slight differences are due to inaccurate time measurements. When regarding only the set of instances where the advanced presolving does additional reductions, the effects become clearer: while increasing the presolving time by about 50% in the shifted geometric mean, 14.1% additional variables and 4.5% additional constraints are removed from the problem, respectively. This is depicted in Figure 4. The majority of the variables is removed by the dominating columns presolver, which

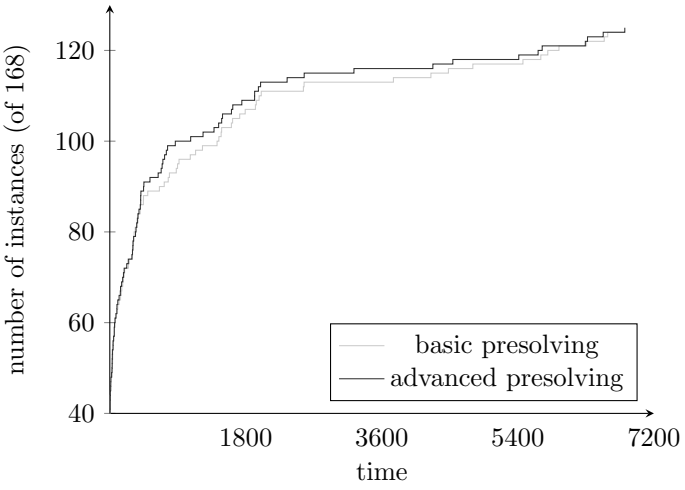


Figure 3: Performance diagram for the MMM test set. The graph indicates the number of instances solved within a certain time.

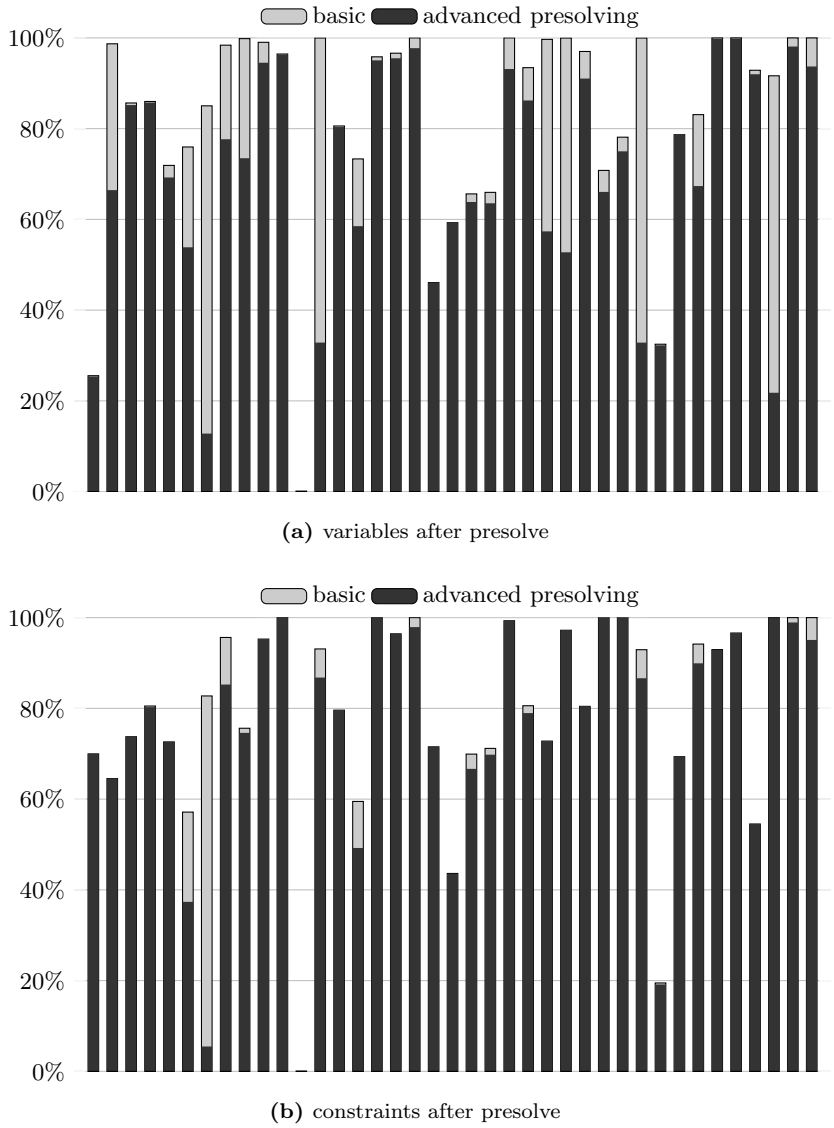
	Basic Presolving					Advanced Presolving				
Test set	Vars%	Conss%	PTime	STime	Solv.	Vars%	Conss%	PTime	STime	Solv.
scm (41)	67.24	57.29	2.22	1000.8	15	40.90	35.79	3.18	527.0	17
MMM:all (168)	83.33	82.69	0.17	334.9	124	80.04	81.65	0.19	317.1	125
MMM:eq (129)	83.53	82.62	0.13	346.4	96	83.53	82.62	0.13	346.6	96
MMM:add (39)	82.66	82.90	0.42	299.4	28	68.50	78.43	0.63	235.9	29

Table 1: Comparison of basic and advanced presolving on the supply chain management test set and the MMM test set, complete as well as divided into instances with equal presolving reductions and instances where the new presolvers found additional reductions. We list the average percentage of variables and constraints remaining after presolving, the shifted geometric means of presolving and solving times, and the number of instances solved to optimality.

removes about 11% of the variables on average, the connected components presolver and the stuffing have a smaller impact with less than 1% removed variables and constraints, respectively. Often, the reductions found by the new techniques also allow other presolving methods to find additional reductions. As an example, see bley_xl1, where the dominating columns presolver finds 76 reductions, which results in more than 4200 additionally removed variables and 135 000 additionally removed constraints. On this set of instances, the advanced presolving reduces the shifted geometric mean of the solving time by 21% in the end.

7 Conclusions

In this paper, we reported on three presolving techniques for mixed integer programming which were implemented in the state-of-the-art non-commercial MIP solver SCIP. At first, they were developed with a focus on a set of real-world supply chain management instances. Many of these contain independent subproblems which the connected components presolver can identify, solve, and remove from the problem during presolving. On the other hand, the dominating columns presolver finds reductions for all the regarded instances, removing about a quarter of the variables from some of the problems. In addition the stuffing singleton columns presolver finds reductions,



(a) variables after presolve

(b) constraints after presolve

Figure 4: Size of the presolved instances relative to the original number of variables and constraints for all instances from the MMM test set where the new presolving techniques find reductions.

although not as many as the dominating columns presolver. Together, they help to significantly improve SCIP's overall performance on this class of instances.

Besides this set of supply chain management instances, we also regarded a set of general MIP instances from various contexts. On this set, we cannot expect the presolving steps to work on all or a majority of the instances, because many of them miss the structure needed. As a consequence, it is very important that the new presolvers do not cause a large overhead when the structure is missing, a goal we obtained by our implementation. On those instances where the new presolvers find reductions, however, they notably speed up the solution process.

Our results show that there is still a need for new presolving techniques, also in an environment

which already encorporates various such techniques. In spite of the maturity of MIP solvers, these results should motivate further research in this area, especially since presolving is one of the most important components of a MIP solver.

References

[1] K. Aardal, R. E. Bixby, C. A. J. Hurkens, A. K. Lenstra, and J. W. Smeltink. Market split and basis reduction: Towards a solution of the Cornuéjols-Dawande instances. *INFORMS Journal on Computing*, 12(3):192–202, 2000.

[2] T. Achterberg. *Constraint Integer Programming*. PhD thesis, Technische Universität Berlin, 2007.

[3] T. Achterberg. SCIP: Solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, 2009.

[4] T. Achterberg, T. Koch, and A. Martin. MIPLIB 2003. *Operations Research Letters*, 34(4):1–12, 2006.

[5] E. D. Andersen. Finding all linearly dependent rows in large–scale linear programming. *Optimization Methods and Software*, 6:219–227, 1995.

[6] E. D. Andersen and K. D. Andersen. Presolving in linear programming. *Mathematical Programming*, 71:221–245, 1995.

[7] A. Atamtürk, G. L. Nemhauser, and M. W. P. Savelsbergh. Conflict graphs in solving integer programming problems. *European Journal of Operational Research*, 121(1):40–55, 2000.

[8] A. Atamtürk and M. W. P. Savelsbergh. Integer-programming software systems. *Annals of Operations Research*, 140:67–124, 2005.

[9] D. A. Babayev and S. S. Mardanov. Reducing the number of variables in integer and linear programming problems. *Computational Optimization and Applications*, 3(2):99–109, 1994.

[10] R. E. Bixby. Numerical aspects of linear and integer programming. Lecture at Universität Erlangen-Nürnberg, October 2011.

[11] R. E. Bixby, S. Ceria, C. M. McZeal, and M. W. P. Savelsbergh. An updated mixed integer programming library: MIPLIB 3.0. *Optima*, (58):12–15, June 1998.

[12] R. E. Bixby and E. Rothberg. Progress in computational mixed integer programming—a look back from the other side of the tipping point. *Annals of Operations Research*, 149:37–41, 2007.

[13] R. E. Bixby and D. K. Wagner. A note on detecting simple redundancies in linear systems. *Operations Research Letters*, 6(1):15–17, 1987.

[14] R. Borndörfer. *Aspects of Set Packing, Partitioning, and Covering*. PhD thesis, Technische Universität Berlin, 1998.

[15] A. L. Brearley, G. Mitra, and H. P. Williams. Analysis of mathematical programming problems prior to applying the simplex algorithm. *Mathematical Programming*, 8:54–83, 1975.

[16] H. Crowder, E. L. Johnson, and M. Padberg. Solving Large-Scale Zero-One Linear Programming Problems. *Operations Research*, 31(5):803–834, 1983.

[17] C. Daskalakis, R. M. Karp, E. Mossel, S. Riesenfeld, and E. Verbin. Sorting and selection in posets. In *SODA '09: Proceedings of the Nineteenth Annual ACM -SIAM Symposium on Discrete Algorithms*, pages 392–401, Philadelphia, PA, USA, 2009. Society for Industrial and Applied Mathematics.

[18] A. Fügenschuh and A. Martin. Computational integer programming and cutting planes. In K. Aardal, G. L. Nemhauser, and R. Weismantel, editors, *Discrete Optimization*, volume 12 of *Handbooks in Operations Research and Management Science*, chapter 2, pages 69–122. Elsevier, 2005.

[19] M. Guignard and K. Spielberg. Logical reduction methods in zero-one programming: Minimal preferred variables. *Operations Research*, 29(1):49–74, 1981.

[20] K. L. Hoffman and M. Padberg. Improving LP-Representations of Zero-One Linear Programs for Branch-and-Cut. *ORSA Journal on Computing*, 3(2):121–134, 1991.

[21] J. Hopcroft and R. Tarjan. Algorithm 447: efficient algorithms for graph manipulation. *Commun. ACM*, 16(6):372–378, 1973.

[22] E. L. Johnson and U. H. Suhl. Experiments in integer programming. *Discrete Applied Mathematics*, 2(1):39–55, 1980.

[23] T. Koch, T. Achterberg, E. Andersen, O. Bastert, T. Berthold, R. E. Bixby, E. Danna, G. Gamrath, A. M. Gleixner, S. Heinz, A. Lodi, H. Mittelmann, T. Ralphs, D. Salvagnin, D. E. Steffy, and K. Wolter. MIPLIB 2010. *Mathematical Programming Computation*, 3(2):103–163, 2011.

[24] A. Mahajan. Presolving mixed–integer linear programs. In J. J. Cochran, L. A. Cox, P. Keskinocak, J. P. Kharoufeh, and J. C. Smith, editors, *Wiley Encyclopedia of Operations Research and Management Science*, pages 4141–4149. John Wiley & Sons, Inc., 2011.

[25] G. L. Nemhauser and L. A. Wolsey. *Integer and combinatorial optimization.* Wiley-Interscience, New York, NY, USA, 1988.

[26] M. W. P. Savelsbergh. Preprocessing and probing techniques for mixed integer programming problems. *ORSA Journal on Computing*, 6:445–454, 1994.

[27] U. Suhl and R. Szymanski. Supernode processing of mixed-integer models. *Computational Optimization and Applications*, 3(4):317–331, 1994.

[28] H. Williams. A reduction procedure for linear and integer programming models. In *Redundancy in Mathematical Programming*, volume 206 of *Lecture Notes in Economics and Mathematical Systems*, pages 87–107. Springer Berlin Heidelberg, 1983.

[29] H. P. Williams. The elimination of integer variables. *The Journal of the Operational Research Society*, 43(5):387–393, 1992.

[30] L. A. Wolsey. *Integer programming.* Wiley-Interscience, New York, NY, USA, 1998.

[31] R. Wunderling. *Paralleler und objektorientierter Simplex-Algorithmus.* PhD thesis, Technische Universität Berlin, 1996.

[32] N. Zhu and K. Broughan. A note on reducing the number of variables in integer programming problems. *Computational Optimization and Applications*, 8(3):263–272, 1997.

A Detailed Computational Results

In this appendix, we present detailed results of our computational experiments presented in Section 6. Table 2 lists results for the supply chain management instances, while Table 3 shows the instances from the MMM test set.

For each instance, we list the original number of variables and constraints. For both the basic presolving as well as the advanced presolving, which includes the presolving techniques presented in this paper, we list the number of variables and constraints after presolving, the presolving time (PTime), and the total solving time (STime). If the time limit was reached, we list the gap at termination instead of the time, printed in italics. As in [23], the gap for a given primal bound pb and dual bound db is computed by the following formula:

$$\text{gap(db, pb)} = \begin{cases} 0.0 & pb = db \\ \infty & pb \cdot db \leq 0 \\ \frac{|pb-db|}{\min\{|pb|;|db|\}} & else \end{cases}$$

If the gap is infinite, we print "inf%", if it is larger than 100 000, we replace the last three digits by a "k". For the advanced presolving, we additionally present the increase in the root LP dual bound (before cutting plane separation) in column "LP Δ%". For the dominating columns and stuffing presolver, we show the number of calls, the time spent in the presolver, and the number of variables fixed by dominating columns (fixed) and stuffing (stuff). Finally, for the components presolver, we list the number of calls, the time, the number of components solved, and the total number of components detected as well as the number of fixed variables and deleted constraints. Whenever one variant dominates the other in one criterion significantly, we print the dominating value in bold for the instance.

At the bottom of the table, we present aggregated results. We list the average percentage of variables and constraints remaining after presolving, the average root LP dual bound increase, and the shifted geometric mean of the presolving and solving time (instances hitting the time limit account for 7200 seconds). We use a shift of 10 seconds for the solving time and 0.01 seconds for the presolving time. For the presolvers, we show the average number of presolving calls, the shifted geometric mean of the time spent in the presolver, again with a shift of 0.01, the average number of components solved and detected, and the average percentages of variables and constraints fixed or deleted by the presolvers. Underneath we print the number of solved instances for the two different presolving settings and a line which lists the same averages, but computed for only the subset of instances solved to optimality by both variants. Moreover, for the MMM test set, we print two rows with averages restricted to the instances where the advanced presolving found additional reductions ("applied") and did not find any reductions ("not appl."), together with the number of instances in the corresponding sets. These lines are only printed for the MMM test set because the advanced presolving finds additional reductions for all supply chain management instances.

Instance	Original		Basic Presolving				Advanced Presolving					Domcol + Stuffing				Components				
	Vars	Conss	Vars	Conss	PTime	STime	Vars	Conss	PTime	LP Δ%	STime	Calls	Time	Fixed	Stuff	Calls	Time	solv/total	Fixed	DelConss
snp-001-01	3314	2195	2781	1701	0.2	*0.1%*	**1976**	**1306**	0.2	+0.00	*0.1%*	3	0.01	4	0	1	0.03	4/5	800	395
snp-001-02	16001	10308	13561	8090	**1.0**	*0.2%*	**11053**	**6746**	1.2	+0.00	*0.1%*	5	0.06	14	0	1	0.06	10/11	2490	1341
snp-001-03	18071	10973	8839	4780	**1.9**	*0.1%*	**8396**	**4589**	2.2	**+6.58**	*0.1%*	21	0.36	62	51	1	0.03	8/9	260	125
snp-001-04	36874	22518	22213	12078	**1.8**	*21.4%*	**21302**	**11692**	3.3	**+6.23**	**10.1%**	6	0.23	27	122	1	1.28	8/9	742	372
snp-001-05	56586	34605	36175	20076	**4.0**	**29.9%**	**35506**	**19832**	20.6	+0.00	*33.0%*	8	0.53	49	92	1	16.12	8/10	484	209
snp-001-06	80237	50535	56138	30707	**12.8**	*5.4%*	**55138**	**30241**	21.4	**+0.00**	**4.1%**	14	1.53	134	2	1	0.06	10/12	785	395
snp-001-07	104078	66598	74787	41939	28.8	**190.8%**	**73473**	**41264**	**15.1**	-0.00	*216.2%*	13	1.77	119	3	1	0.09	12/14	1112	604
snp-001	139949	90739	102813	58714	82.8	*288.3%*	**100989**	**57726**	**21.5**	-0.00	**197.9%**	16	2.97	141	3	1	0.12	13/15	1616	930
snp-002-01	12041	6395	2681	1228	**0.1**	**0.2**	**19**	**8**	0.4	+0.00	0.4	4	0.00	584	2	1	0.22	172/173	1141	552
snp-002-02	25632	13576	10648	4438	**0.4**	1.4	**31**	**10**	0.8	**+0.04**	**0.8**	5	0.03	3270	574	1	0.28	355/356	2606	1201
snp-002-03	40459	21378	18192	7264	**0.5**	9.1	**37**	**12**	1.3	**+0.03**	**1.4**	8	0.06	6229	1195	1	0.42	480/481	3914	1705
snp-002-04	70164	37012	34435	13444	**1.3**	51.2	**49**	**16**	1.5	**+0.01**	**1.6**	12	0.10	14467	385	1	0.44	484/485	5003	2016
snp-002-05	115043	60685	59315	23278	15.4	*0.0%*	**23**	**17**	**2.2**	**+0.02**	**2.4**	18	0.29	26180	613	1	0.43	490/490	7000	3089
snp-002-06	149904	79517	79772	31461	32.0	*0.0%*	**33**	**33**	**3.1**	**+0.02**	**3.3**	22	0.40	35193	1658	1	0.46	529/530	8282	3679
snp-002	382553	209616	228477	93891	117.5	*216.5%*	**1904**	**1375**	**47.7**	**+0.16**	**0.0%**	44	2.76	107214	843	1	33.06	1102/1110	23303	13087
snp-003-01	2205	1238	1490	582	0.0	0.1	**1484**	**580**	0.1	+0.00	0.1	2	0.00	2	0	1	0.00	0/1	0	0
snp-003-02	3748	2435	2171	965	0.1	0.1	**2157**	**959**	0.1	+0.00	0.1	4	0.01	6	0	1	0.00	0/1	0	0
snp-003-03	9091	5052	6757	2921	**0.1**	**0.5**	**6727**	**2907**	0.3	+0.00	0.8	8	0.05	14	0	1	0.00	0/1	0	0
snp-003-04	26564	15031	17458	8454	**0.4**	*12.8%*	**17380**	**8416**	0.9	+0.00	**9.7%**	20	0.31	38	0	1	0.01	0/1	0	0
snp-003-05	42468	23489	28160	13666	**0.4**	*13.7%*	**28146**	**13660**	0.8	+0.00	**12.0%**	4	0.16	8	0	1	0.01	0/1	0	0
snp-003-06	47713	26674	31165	15289	0.5	*12.1%*	**31141**	**15283**	0.6	+0.00	*12.5%*	4	0.10	18	0	1	0.02	0/1	0	0
snp-003	87652	47907	58750	28887	**1.1**	**14.9%**	**58730**	**28877**	1.3	+0.00	*17.2%*	5	0.22	10	0	1	0.04	0/1	0	0
snp-004-02	26597	13130	18257	8079	**0.2**	1.8	**156**	**38**	0.9	**+0.00**	**1.0**	4	0.04	736	0	1	0.60	732/733	17359	8037
snp-004-03	58693	26637	43050	16703	**0.4**	1981.6	**9785**	**3744**	1.6	**+0.01**	**3.1**	3	0.08	738	16	1	0.90	577/578	32499	12948
snp-004-04	126553	49032	101464	32080	**0.9**	*1728k%*	**100052**	**31912**	1.4	**+0.01**	*1727k%*	3	0.23	785	55	1	0.08	4/5	604	157
snp-004-05	140535	53508	113625	35182	**1.2**	*1729k%*	**112210**	**35014**	1.6	**+0.01**	*1729k%*	3	0.28	788	55	1	0.09	4/5	604	157
snp-004-06	210279	75833	174116	50518	**2.1**	*1735k%*	**172651**	**50340**	2.6	**+0.02**	*1735k%*	4	0.59	827	48	1	0.14	5/6	618	163
snp-004	321527	111563	270488	75058	**3.9**	*1745k%*	**268919**	**74864**	4.2	**+0.02**	*1744k%*	5	1.10	852	42	1	0.20	5/6	700	179
snp-005-01	13518	5019	7177	2308	**0.2**	0.9	**8**	**8**	0.6	**+0.00**	**0.7**	5	0.04	65	395	1	0.34	272/273	6397	2038
snp-005-02	34933	11631	23865	7529	**0.4**	3.1	**10**	**9**	1.1	**+0.00**	**1.1**	6	0.07	411	907	1	0.40	248/249	21987	6989
snp-005-03	83330	25458	64265	17894	**0.9**	16.1	**10**	**9**	3.2	**+0.00**	**3.4**	11	0.46	1037	1477	1	1.28	314/315	61082	17258
snp-005-04	310163	71357	270763	54960	**6.7**	397.5	**146**	**158**	15.3	**+0.00**	**20.6**	16	3.30	2698	1372	1	4.52	279/280	265543	53974
snp-005-05	560637	121943	507447	100741	55.3	1425.5	**223**	**231**	**40.6**	**+0.00**	**102.5**	21	9.45	2174	1203	1	12.20	370/371	502652	99441
snp-005-06	680745	146213	620715	122807	107.6	*0.0%*	**311220**	**71302**	**82.5**	**+0.00**	*0.0%*	26	13.75	2352	1094	1	5.91	322/323	304656	50296
snp-005	1182136	255629	1097503	215672	399.4	*187.5%*	**542064**	**129989**	**330.7**	**+0.00**	**41.4%**	39	36.11	3039	1108	1	10.92	321/322	549126	83910
snp-008-01	4214	1584	1563	353	0.1	0.3	**43**	**8**	0.2	**+0.00**	0.2	5	0.00	54	206	1	0.11	38/39	1191	281
snp-008-02	20840	7743	12025	2968	**0.4**	*0.0%*	**470**	**69**	7.8	**+0.15**	*0.0%*	18	0.11	379	2321	1	7.13	65/68	8389	2213
snp-008-03	57707	27374	40053	17005	**7.4**	*0.0%*	**36210**	**15293**	7.8	-0.00	*0.0%*	16	0.72	350	591	1	0.38	21/22	2239	1310
snp-008-04	67128	33017	47555	21578	11.0	*0.1%*	**43496**	**19653**	10.9	-0.00	*0.1%*	17	1.04	359	565	1	0.44	21/22	2521	1544
snp-008-05	194751	109140	146594	80078	111.8	*1044%*	**139365**	**75696**	**66.9**	-0.00	*1044%*	40	9.73	341	656	1	1.50	25/26	6526	4829
snp-008	379030	210508	276023	143139	**258.1**	*1073%*	**264498**	**136102**	287.2	+0.00	*1073%*	58	26.82	435	965	1	2.97	25/26	10764	7883
average (41)	100%	100%	67.24%	57.29%	2.22	1000.8	40.90%	35.79%	3.18	+0.32	527.0	13.3	0.32	3.53%	0.86%	1.0	0.31	178.9/180.1	18.43%	15.89%
solved (of 41)						15					17									
all opt (15)	100%	100%	60.87%	50.25%	0.39	23.6	14.52%	10.64%	0.99	+0.01	3.8	7.6	0.08	4.23%	1.24%	1.0	0.28	288.1/289.1	36.28%	32.28%

Table 2. Detailed computational results on the set of supply-chain management instances

Instance	Original		Basic Presolving				Advanced Presolving					Domcol + Stuffing				Components				
	Vars	Conss	Vars	Conss	PTime	STime	Vars	Conss	PTime	LP Δ%	STime	Calls	Time	Fixed	Stuff	Calls	Time	solv/total	Fixed	DelConss
10teams	2025	230	1600	210	0.1	34.5	1600	210	0.1	+0.00	35.9	1	0.01	0	0	1	0.00	0/1	0	0
30n20b8	18380	576	4696	403	**7.3**	**502.8**	**4665**	403	8.4	+0.00	681.9	137	0.77	31	0	1	0.00	0/1	0	0
a1c1s1	3648	3312	2492	2232	0.2	*17.1%*	2492	2232	0.3	+0.00	*17.1%*	2	0.02	0	0	1	0.00	0/1	0	0
acc-tight5	1339	3052	996	2257	1.1	105.0	996	2257	1.2	+0.00	105.2	10	0.03	0	0	1	0.00	0/1	0	0
aflow30a	842	479	841	478	0.2	12.6	841	478	0.2	+0.00	13.0	2	0.00	0	0	1	0.00	0/1	0	0
aflow40b	2728	1442	2726	1440	0.6	2798.4	2726	1440	0.6	+0.00	2805.0	2	0.01	0	0	1	0.00	0/1	0	0
air03	10757	124	10617	80	0.5	5.6	**7148**	80	0.6	+0.00	**1.4**	3	0.19	3469	0	1	0.01	0/1	0	0
air04	8904	823	7627	607	1.2	78.8	**7586**	607	1.3	+0.00	**70.2**	5	0.07	41	0	1	0.01	0/1	0	0
air05	7195	426	6187	343	0.2	47.9	**6170**	**342**	0.3	+0.00	**38.3**	4	0.04	16	0	1	0.00	0/1	0	0
app1-2	26871	53467	26265	52555	**4.8**	1081.5	26265	52555	5.4	+0.00	1078.7	13	0.58	0	0	1	0.03	0/1	0	0
arki001	1388	1048	998	761	0.1	*0.0%*	**961**	761	0.2	+0.00	*0.0%*	3	0.01	37	0	1	0.01	0/1	0	0
ash608gpia-3col	3651	24748	3651	24748	0.3	80.1	3651	24748	0.3	+0.00	80.6	1	0.03	0	0	1	0.01	0/1	0	0
atlanta-ip	48738	21732	17240	19083	**1.1**	*8.6%*	17240	19083	1.4	+0.00	*8.6%*	4	0.12	0	0	1	0.02	0/1	0	0
beasleyC3	2500	1750	1704	1153	0.0	*14.6%*	1704	1153	0.0	+0.00	*14.6%*	1	0.00	0	0	1	0.00	0/1	0	0
bell3a	133	123	88	70	0.0	6.2	88	70	0.0	+0.00	6.4	1	0.00	0	0	1	0.00	0/1	0	0
bell5	104	91	79	52	0.0	0.8	**56**	**34**	0.0	**+0.44**	**0.6**	4	0.00	1	0	1	0.03	1/2	22	18
bab5	21600	4964	21432	4740	4.4	*1.9%*	21432	4740	4.3	+0.00	*1.9%*	1	0.03	0	0	1	0.02	0/1	0	0
biella1	7328	1203	7311	1202	0.7	879.1	7311	1202	0.7	+0.00	876.9	1	0.02	0	0	1	0.01	0/1	0	0
bienst2	505	576	449	520	0.0	408.2	449	520	0.0	+0.00	408.9	1	0.01	0	0	1	0.00	0/1	0	0
binkar10_1	2298	1026	1443	825	0.1	184.8	1443	825	0.1	+0.00	184.2	1	0.00	0	0	1	0.00	0/1	0	0
blend2	353	274	306	156	0.0	0.7	306	156	0.0	+0.00	0.8	2	0.00	0	0	1	0.00	0/1	0	0
bley_xl1	5831	175620	4958	145307	**18.7**	*165.9%*	**746**	**9616**	311.9	**+5.99**	**425.8**	90	5.28	76	0	1	0.00	0/1	0	0
bnatt350	3150	4923	1738	1767	0.7	358.0	1738	1767	0.8	+0.00	360.5	2	0.01	0	0	1	0.00	0/1	0	0
cap6000	6000	2176	5904	2081	**0.2**	36.1	**4660**	**1855**	0.5	+0.00	**23.7**	2	0.03	1216	0	1	0.01	0/1	0	0
core2536-691	15293	2539	15269	1920	**0.8**	723.9	**11238**	**1894**	1.1	+0.00	**233.7**	6	0.15	4030	0	1	0.01	0/1	0	0
cov1075	120	637	120	637	0.1	*7.2%*	120	637	0.1	+0.00	*7.3%*	1	0.00	0	0	1	0.01	0/1	0	0
csched010	1758	351	1654	295	0.1	6365.7	1654	295	0.1	+0.00	6358.6	1	0.00	0	0	1	0.00	0/1	0	0
dano3mip	13873	3202	13837	3151	0.9	*21.6%*	13837	3151	0.8	+0.00	*21.6%*	1	0.03	0	0	1	0.01	0/1	0	0
danoint	521	664	513	656	0.0	5783.3	513	656	0.0	+0.00	5794.5	1	0.00	0	0	1	0.00	0/1	0	0
dcmulti	548	290	547	271	0.0	1.9	547	271	0.0	+0.00	1.9	2	0.00	0	0	1	0.00	0/1	0	0
dfn-gwin-UUM	938	158	936	156	0.0	154.8	936	156	0.0	+0.00	153.7	1	0.00	0	0	1	0.00	0/1	0	0
disctom	10000	399	9991	394	0.1	3.2	9991	394	0.1	+0.00	3.3	1	0.01	0	0	1	0.01	0/1	0	0
ds	67732	656	67076	625	**2.5**	*572.6%*	**64030**	625	5.8	+0.00	*600.8%*	2	3.24	3046	0	1	0.12	0/2	0	0
dsbmip	1886	1182	1638	987	0.2	2.0	1638	987	0.2	+0.00	1.9	3	0.01	0	0	1	0.01	0/1	0	0
egout	141	98	49	37	0.0	0.0	49	37	0.0	+0.00	0.0	1	0.00	0	0	1	0.00	0/1	0	0
eil33-2	4516	32	4484	32	0.4	60.3	4484	32	0.5	+0.00	60.3	1	0.13	0	0	1	0.00	0/1	0	0
eilB101	2818	100	2718	100	0.2	**316.8**	**2715**	100	0.2	+0.00	405.7	2	0.02	3	0	1	0.01	0/1	0	0
enigma	100	21	100	21	0.0	0.6	100	21	0.0	+0.00	0.6	1	0.00	0	0	1	0.00	0/1	0	0
enlight13	338	169	338	169	0.0	*56.0%*	338	169	0.0	+0.00	*56.0%*	1	0.00	0	0	1	0.00	0/1	0	0
enlight14	392	196	392	196	0.0	*inf%*	392	196	0.0	+0.00	*inf%*	1	0.00	0	0	1	0.01	0/1	0	0
ex9	10404	40962	12	10	30.4	30.6	**0**	**0**	30.9	+0.00	31.0	15	0.68	4	0	0	0.00	0/0	0	0
fast0507	63009	507	62997	472	**0.8**	2556.3	**20700**	**440**	1.8	+0.00	**300.3**	5	0.83	42285	0	1	0.02	0/1	0	0
fiber	1298	363	1046	289	0.0	**1.4**	**1043**	289	0.0	+0.00	1.8	2	0.00	3	0	1	0.00	0/1	0	0
fixnet6	878	478	877	477	0.0	1.6	877	477	0.0	+0.00	1.5	1	0.00	0	0	1	0.00	0/1	0	0
flugpl	18	18	14	13	0.0	0.0	14	13	0.0	+0.00	0.1	1	0.00	0	0	1	0.00	0/1	0	0

continue next page

Instance	Original		Basic Presolving				Advanced Presolving					Domcol + Stuffing				Components				
	Vars	Conss	Vars	Conss	PTime	STime	Vars	Conss	PTime	LP Δ%	STime	Calls	Time	Fixed	Stuff	Calls	Time	solv/total	Fixed	DelConss
gen	870	780	638	464	0.0	0.1	**509**	**384**	0.0	+0.00	0.1	2	0.00	75	49	1	0.01	0/1	0	0
gesa2-o	1224	1248	1176	1200	0.1	1.2	1176	1200	0.1	+0.00	1.3	2	0.00	0	0	1	0.00	0/1	0	0
gesa2	1224	1392	1176	1344	0.1	1.2	1176	1344	0.1	+0.00	1.2	2	0.00	0	0	1	0.00	0/1	0	0
gesa3	1152	1368	1080	1296	0.1	1.6	1080	1296	0.1	+0.00	1.7	2	0.00	0	0	1	0.00	0/1	0	0
gesa3_o	1152	1224	1080	1152	0.0	1.5	1080	1152	0.0	+0.00	1.5	2	0.01	0	0	1	0.00	0/1	0	0
glass4	322	396	317	392	0.0	*50.0%*	317	392	0.0	+0.00	*50.0%*	1	0.01	0	0	1	0.00	0/1	0	0
gmu-35-40	1205	424	652	357	0.2	*0.0%*	652	357	0.2	+0.00	*0.0%*	3	0.02	0	0	1	0.00	0/1	0	0
gt2	188	29	173	28	0.0	0.1	173	28	0.0	+0.00	0.0	1	0.00	0	0	1	0.00	0/1	0	0
harp2	2993	112	999	92	0.0	1914.2	999	92	0.0	+0.00	1904.3	1	0.00	0	0	1	0.00	0/1	0	0
iis-100-0-cov	100	3831	100	3831	0.1	1914.3	100	3831	0.1	+0.00	1913.5	1	0.01	0	0	1	0.00	0/1	0	0
iis-bupa-cov	345	4803	341	4803	0.1	6533.7	341	4803	0.1	+0.00	6594.4	1	0.00	0	0	1	0.00	0/1	0	0
iis-pima-cov	768	7201	736	7201	0.1	**814.1**	**730**	7201	0.2	+0.00	862.0	2	0.02	6	0	1	0.02	0/1	0	0
khb05250	1350	101	1299	100	0.0	0.5	1299	100	0.0	+0.00	0.5	1	0.00	0	0	1	0.00	0/1	0	0
lectsched-4-obj	7901	14163	2605	4788	1.5	308.7	2605	4788	1.6	+0.00	310.0	12	0.04	0	0	1	0.00	0/1	0	0
liu	1156	2178	1154	2178	0.0	*127.5%*	1154	2178	0.0	+0.00	*127.5%*	1	0.00	0	0	1	0.00	0/1	0	0
l152lav	1989	97	1989	97	0.0	**2.5**	1989	97	0.0	+0.00	2.7	1	0.00	0	0	1	0.00	0/1	0	0
lseu	89	28	86	27	0.0	0.5	**85**	27	0.0	+0.00	0.5	2	0.00	1	0	1	0.00	0/1	0	0
m100n500k4r1	500	100	500	100	0.0	*4.2%*	500	100	0.0	+0.00	*4.2%*	1	0.01	0	0	1	0.00	0/1	0	0
macrophage	2260	3164	2260	3164	0.1	*34.3%*	**2209**	**3098**	0.1	**+900.00**	**30.2%**	2	0.02	0	0	1	0.06	3/4	51	66
manna81	3321	6480	3321	6480	0.1	0.9	3321	6480	0.1	+0.00	0.8	1	0.00	0	0	1	0.00	0/1	0	0
map18	164547	328818	15412	31207	2.0	405.4	15412	31207	2.1	+0.00	407.2	1	0.02	0	0	1	0.02	0/1	0	0
map20	164547	328818	15412	31207	2.0	446.9	15412	31207	2.0	+0.00	448.2	1	0.02	0	0	1	0.02	0/1	0	0
markshare1	62	6	50	6	0.0	*inf%*	50	6	0.0	+0.00	*inf%*	1	0.00	0	0	1	0.00	0/1	0	0
markshare2	74	7	60	7	0.0	*inf%*	60	7	0.0	+0.00	*inf%*	1	0.00	0	0	1	0.00	0/1	0	0
mas74	151	13	150	13	0.0	645.2	150	13	0.0	+0.00	643.0	1	0.00	0	0	1	0.00	0/1	0	0
mas76	151	12	150	12	0.0	50.9	150	12	0.0	+0.00	50.3	1	0.00	0	0	1	0.00	0/1	0	0
mcsched	1747	2107	1495	1853	0.0	163.5	1495	1853	0.1	+0.00	165.7	1	0.00	0	0	1	0.00	0/1	0	0
mik-250-1-100-1	251	151	251	100	0.0	1938.3	251	100	0.0	+0.00	1941.8	1	0.00	0	0	1	0.00	0/1	0	0
mine-166-5	830	8429	709	6698	1.6	43.5	709	6698	1.6	+0.00	43.2	4	0.03	0	0	1	0.00	0/1	0	0
mine-90-10	900	6270	867	5814	0.8	560.8	867	5814	0.7	+0.00	567.9	2	0.01	0	0	1	0.01	0/1	0	0
misc03	160	96	138	95	0.0	**1.2**	138	95	0.0	+0.00	1.4	2	0.00	0	0	1	0.00	0/1	0	0
misc06	1808	820	1260	517	0.0	0.6	1260	517	0.0	+0.00	0.6	1	0.00	0	0	1	0.00	0/1	0	0
misc07	260	212	232	223	0.1	11.6	232	223	0.1	+0.00	11.8	2	0.00	0	0	1	0.00	0/1	0	0
mitre	10724	2054	4941	**1469**	**4.1**	**4.4**	**4938**	1470	5.2	+0.00	5.5	157	0.85	67	0	1	0.00	0/1	0	0
mkc	5325	3411	3273	1287	0.2	*1.2%*	3273	1287	0.3	+0.00	*1.2%*	1	0.06	0	0	1	0.00	0/1	0	0
mod008	319	6	319	6	0.0	1.0	319	6	0.0	+0.00	0.9	1	0.00	0	0	1	0.00	0/1	0	0
mod010	2655	146	2572	144	0.1	0.9	2572	144	0.1	+0.00	**0.7**	1	0.00	0	0	1	0.00	0/1	0	0
mod011	10958	4480	6495	1954	0.3	**151.9**	**6490**	**1951**	0.4	**+0.00**	160.2	2	0.06	2	0	1	0.00	0/1	0	0
modglob	422	291	384	286	0.0	1.0	384	286	0.0	+0.00	1.1	1	0.00	0	0	1	0.00	0/1	0	0
momentum1	5174	42680	2746	13212	4.9	*18.6%*	2746	13212	5.0	+0.00	*18.6%*	12	0.12	0	0	1	0.00	0/1	0	0
momentum2	3732	24237	2774	14861	12.9	*0.7%*	2774	14861	13.4	+0.00	*0.7%*	13	0.25	0	0	1	0.01	0/1	0	0
momentum3	13532	56822	13151	49375	203.1	*254.4%*	13151	49375	205.3	+0.00	*254.4%*	6	0.42	0	0	1	0.05	0/1	0	0
msc98-ip	21143	15850	12733	14987	**0.7**	*12.4%*	12733	14987	0.9	+0.00	*12.4%*	3	0.07	0	0	1	0.01	0/1	0	0
mspp16	29280	561657	4065	524814	484.4	4265.7	4065	524814	484.9	+0.00	4244.1	1	3.65	0	0	1	2.26	0/1	0	0
mzzv11	10240	9499	6719	6642	**20.4**	**259.9**	**6537**	**6333**	26.2	**+1.10**	307.8	79	1.07	194	0	1	0.01	0/1	0	0

continue next page

Instance	Original		Basic Presolving				Advanced Presolving					Domcol + Stuffing				Components				
	Vars	Conss	Vars	Conss	PTime	STime	Vars	Conss	PTime	LP Δ%	STime	Calls	Time	Fixed	Stuff	Calls	Time	solv/total	Fixed	DelConss
mzzv42z	11717	10460	7728	7445	22.8	**197.5**	**7446**	**7300**	23.9	+0.00	225.7	84	1.22	162	0	1	0.03	1/2	88	71
n3div36	22120	4484	22120	4453	**1.3**	*9.8%*	**20602**	4453	2.0	+0.00	*9.3%*	2	0.08	1518	0	1	0.04	0/1	0	0
n3seq24	119856	6044	119856	5950	**11.7**	*10.4%*	119856	5950	12.8	+0.00	*10.4%*	1	0.72	0	0	1	0.46	0/1	0	0
n4-3	3596	1236	3360	996	0.1	775.8	**3100**	**976**	0.1	**+20.90**	754.5	4	0.02	230	0	1	0.01	5/6	13	5
neos-1109824	1520	28979	1520	9979	**0.7**	189.4	1520	9979	0.9	+0.00	189.4	1	0.01	0	0	1	0.00	0/1	0	0
neos-1337307	2840	5687	2840	2023	1.1	*0.0%*	2840	2023	1.2	+0.00	*0.0%*	2	0.14	0	0	1	0.01	0/1	0	0
neos-1396125	1161	1494	1158	1491	0.1	1486.1	1158	1491	0.1	+0.00	1479.1	1	0.00	0	0	1	0.00	0/1	0	0
neos13	1827	20852	1827	17320	1.4	*32.3%*	1827	17320	1.4	+0.00	*32.3%*	1	0.12	0	0	1	0.01	0/1	0	0
neos-1601936	4446	3131	3920	3105	0.3	*33.3%*	3920	3105	0.3	+0.00	*33.3%*	1	0.02	0	0	1	0.01	0/1	0	0
neos18	3312	11402	758	3290	0.1	30.7	758	3290	0.2	+0.00	31.0	1	0.01	0	0	1	0.00	0/1	0	0
neos-476283	11915	10015	11843	9604	18.5	256.5	11843	9604	18.9	+0.00	255.0	1	0.50	0	0	1	0.42	0/1	0	0
neos-686190	3660	3664	3660	3658	0.1	80.8	3660	3658	0.1	+0.00	79.6	1	0.00	0	0	1	0.00	0/1	0	0
neos-849702	1737	1041	1692	987	0.1	448.8	1692	987	0.1	+0.00	445.4	1	0.01	0	0	1	0.00	0/1	0	0
neos-916792	1474	1909	1361	1408	0.6	417.3	1361	1408	0.7	+0.00	416.4	1	0.02	0	0	1	0.02	0/1	0	0
neos-934278	23123	11495	8121	8123	0.8	*4.0%*	8121	8123	0.7	+0.00	*4.0%*	1	0.01	0	0	1	0.00	0/1	0	0
net12	14115	14021	12523	12767	**3.6**	4496.2	12523	12767	4.1	+0.00	4481.6	20	0.54	0	0	1	0.01	0/1	0	0
netdiversion	129180	119589	128968	99483	16.2	6162.8	128968	99483	16.1	+0.00	6124.6	1	0.12	0	0	1	0.12	0/1	0	0
newdano	505	576	449	520	0.0	6364.0	449	520	0.0	+0.00	6306.8	1	0.00	0	0	1	0.00	0/1	0	0
noswot	128	182	120	171	0.0	171.3	120	171	0.0	+0.00	171.2	2	0.00	0	0	1	0.00	0/1	0	0
ns1208400	2883	4289	2596	1981	0.3	1611.0	2596	1981	0.3	+0.00	1624.2	1	0.01	0	0	1	0.01	0/1	0	0
ns1688347	2685	4191	1460	3090	5.5	719.1	1460	3090	5.5	+0.00	720.8	25	0.20	0	0	1	0.00	0/1	0	0
ns1758913	17956	624166	17824	615190	24.4	*527.6%*	17824	615190	24.7	+0.00	*527.6%*	1	0.26	0	0	1	0.13	0/1	0	0
ns1766074	100	182	100	110	0.0	793.5	100	110	0.0	+0.00	785.3	1	0.00	0	0	1	0.00	0/1	0	0
ns1830653	1629	2932	673	1406	0.6	871.6	673	1406	0.6	+0.00	868.6	2	0.00	0	0	1	0.01	0/1	0	0
nsrand-ipx	6621	735	6600	535	**0.7**	*3.3%*	**3798**	535	1.1	+0.00	*3.2%*	2	0.06	2802	0	1	0.01	0/1	0	0
nw04	87482	36	87454	35	**3.8**	38.2	**46143**	35	8.3	+0.00	**31.0**	2	5.16	41311	0	1	0.06	0/1	0	0
opm2-z7-s2	2023	31798	1896	26691	3.2	1291.1	1896	26691	3.1	+0.00	1285.5	1	0.01	0	0	1	0.01	0/1	0	0
opt1217	769	64	759	64	0.0	0.9	759	64	0.0	+0.00	1.0	1	0.00	0	0	1	0.01	0/1	0	0
p0033	33	16	26	12	0.0	0.0	26	12	0.0	+0.00	0.0	1	0.00	0	0	1	0.00	0/1	0	0
p0201	201	133	195	107	0.0	**1.3**	**183**	107	0.0	**+0.42**	1.6	2	0.00	12	0	1	0.00	0/1	0	0
p0282	282	241	200	305	0.0	0.7	200	305	0.0	+0.00	0.7	1	0.00	0	0	1	0.01	0/1	0	0
p0548	548	176	388	239	0.0	0.3	**362**	**209**	0.0	+0.00	0.3	2	0.00	3	0	1	0.00	0/1	0	0
p2756	2756	755	2153	1466	0.2	1.5	**2067**	**1416**	0.3	**+1.41**	1.5	3	0.03	33	0	1	0.00	0/1	0	0
pg5_34	2600	225	2600	225	0.1	1374.0	2600	225	0.1	+0.00	1377.8	1	0.00	0	0	1	0.00	0/1	0	0
pigeon-10	490	931	390	525	0.0	*11.1%*	390	525	0.0	+0.00	*11.1%*	1	0.00	0	0	1	0.00	0/1	0	0
pk1	86	45	86	45	0.0	56.5	86	45	0.0	+0.00	56.3	1	0.00	0	0	1	0.00	0/1	0	0
pp08a	240	136	234	133	0.0	1.9	234	133	0.0	+0.00	**1.5**	1	0.00	0	0	1	0.00	0/1	0	0
pp08aCUTS	240	246	237	243	0.0	**1.3**	237	243	0.0	+0.00	1.5	1	0.00	0	0	1	0.00	0/1	0	0
protfold	1835	2112	1835	2112	0.1	*inf%*	1835	2112	0.1	+0.00	*inf%*	1	0.01	0	0	1	0.00	0/1	0	0
pw-myciel4	1059	8164	1013	4180	0.7	5671.5	1013	4180	0.6	+0.00	5665.0	2	0.01	0	0	1	0.00	0/1	0	0
qiu	840	1192	840	1192	0.0	70.1	840	1192	0.0	+0.00	69.9	1	0.00	0	0	1	0.00	0/1	0	0
qnet1	1541	503	1417	364	0.1	5.5	1417	364	0.1	+0.00	5.7	1	0.00	0	0	1	0.00	0/1	0	0
qnet1_o	1541	456	1330	245	0.0	3.1	1330	245	0.0	+0.00	3.1	1	0.00	0	0	1	0.00	0/1	0	0
rail507	63019	509	62997	473	**1.4**	1797.3	**20698**	**441**	2.7	+0.00	**363.4**	5	1.36	42288	0	1	0.02	0/1	0	0
ran16x16	512	288	512	288	0.0	300.1	512	288	0.0	+0.00	299.8	1	0.00	0	0	1	0.00	0/1	0	0

continue next page

Instance	Original		Basic Presolving				Advanced Presolving					Domcol + Stuffing				Components				
	Vars	Conss	Vars	Conss	PTime	STime	Vars	Conss	PTime	LP Δ%	STime	Calls	Time	Fixed	Stuff	Calls	Time	solv/total	Fixed	DelConss
reblock67	670	2523	627	2271	0.8	372.1	627	2271	0.8	+0.00	367.3	3	0.00	0	0	1	0.00	0/1	0	0
rd-rplusc-21	622	125899	522	25272	36.1	*171k%*	522	25272	36.5	+0.00	*171k%*	4	0.12	0	0	1	0.02	0/1	0	0
rentacar	9557	6803	3105	1325	0.3	**1.9**	**3081**	**1303**	0.3	+0.00	2.1	5	0.03	3	0	1	0.01	2/3	23	22
rgn	180	24	175	24	0.0	0.2	175	24	0.0	+0.00	0.2	1	0.00	0	0	1	0.00	0/1	0	0
rmatr100-p10	7359	7260	7359	7260	0.9	142.6	7359	7260	0.8	+0.00	141.8	2	0.02	0	0	1	0.00	0/1	0	0
rmatr100-p5	8784	8685	8784	8685	1.0	292.3	8784	8685	0.9	+0.00	292.0	2	0.00	0	0	1	0.00	0/1	0	0
rmine6	1096	7078	1084	7066	0.9	2563.6	1084	7066	0.9	+0.00	2573.9	1	0.00	0	0	1	0.01	0/1	0	0
rocII-4-11	9234	21738	1266	3449	5.9	438.1	1266	3449	5.8	+0.00	434.6	20	0.10	0	0	1	0.00	0/1	0	0
rococoC10-001000	3117	1293	2442	576	0.1	*1.6%*	2442	576	0.1	+0.00	*1.7%*	1	0.00	0	0	1	0.01	0/1	0	0
roll3000	1166	2295	832	1219	0.5	*1.2%*	832	1219	0.5	+0.00	*1.2%*	2	0.01	0	0	1	0.00	0/1	0	0
rout	556	291	555	290	0.1	35.7	555	290	0.1	+0.00	35.7	1	0.00	0	0	1	0.00	0/1	0	0
satellites1-25	9013	5996	**7091**	**4158**	9.2	4038.7	7102	4160	9.5	+0.00	**1515.6**	18	0.23	220	0	1	0.00	0/1	0	0
set1ch	712	492	646	426	0.0	**0.8**	646	426	0.0	+0.00	1.1	1	0.00	0	0	1	0.00	0/1	0	0
seymour	1372	4944	1140	4656	**0.1**	*2.2%*	**924**	**4446**	0.4	+0.00	**1.9%**	5	0.01	190	0	1	0.01	0/1	0	0
sp97ar	14101	1761	14099	1637	**0.9**	*7.3%*	**14067**	**1636**	1.4	-0.03	**6.9%**	3	0.15	31	0	1	0.05	0/1	0	0
sp98ic	10894	825	10894	797	**0.8**	*2.5%*	**10877**	797	1.2	+0.00	**2.0%**	2	0.12	17	0	1	0.05	0/1	0	0
sp98ir	1680	1531	1557	1375	1.0	122.7	1557	1375	0.9	+0.00	123.6	1	0.01	0	0	1	0.01	0/1	0	0
stein27	27	118	27	118	0.0	1.1	27	118	0.0	+0.00	1.2	1	0.00	0	0	1	0.01	0/1	0	0
stein45	45	331	45	331	0.0	14.4	45	331	0.0	+0.00	14.1	1	0.00	0	0	1	0.00	0/1	0	0
stp3d	204880	159488	136241	96985	32.6	*inf%*	136241	96985	32.9	+0.00	*inf%*	4	0.67	0	0	1	0.14	0/1	0	0
swath	6805	884	6320	482	0.1	**15.6%**	**6260**	482	0.2	+0.00	*22.1%*	2	0.02	60	0	1	0.01	0/1	0	0
t1717	73885	551	67716	551	**0.9**	*41.2%*	**16102**	551	1.2	+0.00	**38.5%**	2	0.27	51614	0	1	0.01	0/1	0	0
tanglegram1	34759	68342	34759	68342	0.7	907.7	**34099**	**67614**	0.8	+0.00	**683.0**	2	0.08	0	0	1	0.07	137/138	660	728
tanglegram2	4714	8980	4714	8980	0.1	8.3	**4417**	**8538**	0.2	+0.00	**6.9**	2	0.02	0	0	1	0.04	36/37	297	442
timtab1	397	171	201	166	0.0	449.4	201	166	0.0	+0.00	446.4	2	0.00	0	0	1	0.00	0/1	0	0
timtab2	675	294	341	289	0.0	*46.8%*	341	289	0.0	+0.00	*46.8%*	2	0.00	0	0	1	0.00	0/1	0	0
tr12-30	1080	750	1040	722	0.1	1100.8	1040	722	0.1	+0.00	1103.2	1	0.00	0	0	1	0.00	0/1	0	0
triptim1	30055	15706	25446	15574	10.2	2945.4	25446	15574	10.1	+0.00	2944.6	3	0.15	0	0	1	0.05	0/1	0	0
unitcal_7	25755	48939	20297	38656	15.3	2412.0	20297	38656	16.1	+0.00	2426.8	27	0.70	0	0	1	0.02	0/1	0	0
vpm1	378	234	182	129	0.0	0.0	182	129	0.0	+0.00	0.0	1	0.00	0	0	1	0.00	0/1	0	0
vpm2	378	234	181	128	0.0	1.1	181	128	0.0	+0.00	1.0	3	0.00	0	0	1	0.00	0/1	0	0
vpphard	51471	47280	27488	23418	2.1	*inf%*	27488	23418	2.1	+0.00	*inf%*	1	0.03	0	0	1	0.04	0/1	0	0
zib54-UUE	5150	1809	5069	1761	0.1	1997.4	5069	1761	0.1	+0.00	2002.7	1	0.01	0	0	1	0.00	0/1	0	0
average (168)	100%	100%	83.33%	82.69%	0.17	334.9	80.04%	81.65%	0.19	+5.54	317.1	5.9	0.02	2.60%	0.03%	1.0	0.01	1.1/2.1	0.20%	0.17%
solved (of 168)						124					125									
all opt (124)	100%	100%	83.38%	83.49%	0.12	107.3	80.68%	82.76%	0.14	+0.20	101.7	6.5	0.02	2.37%	0.05%	1.0	0.00	1.5/2.5	0.25%	0.22%
applied (39)	100%	100%	82.66%	82.90%	0.42	299.4	68.50%	78.43%	0.63	+23.85	235.9	17.3	0.10	11.18%	0.14%	1.0	0.01	4.7/5.7	0.85%	0.75%
not applied (129)	100%	100%	83.53%	82.62%	0.13	346.4	83.53%	82.62%	0.13	+0.00	346.6	2.5	0.01	0.00%	0.00%	1.0	0.00	0.0/1.0	0.00%	0.00%

Table 3. Detailed computational results on the MMM set