Math. Prog. Comp. (2015) 7:367–398
DOI 10.1007/s12532-015-0083-5

FULL LENGTH PAPER

Progress in presolving for mixed integer programming

Gerald Gamrath[1] · Thorsten Koch[1] · Alexander Martin[2] · Matthias Miltenberger[1] · Dieter Weninger[2]

Received: 15 August 2013 / Accepted: 26 March 2015 / Published online: 5 June 2015

Abstract This paper describes three presolving techniques for solving mixed integer programming problems (MIPs) that were implemented in the academic MIP solver SCIP. The task of presolving is to reduce the problem size and strengthen the formulation, mainly by eliminating redundant information and exploiting problem structures. The first method fixes continuous singleton columns and extends results known from duality fixing. The second analyzes and exploits pairwise dominance relations between variables, whereas the third detects isolated subproblems and solves them independently. The performance of the presented techniques is demonstrated on two MIP test sets. One contains all benchmark instances from the last three MIPLIB versions, while the other consists of real-world supply chain management problems. The computational results show that the combination of all three presolving techniques almost halves the solving time for the considered supply chain management problems. For the MIPLIB instances we obtain a speedup of 20 % on affected instances while not degrading the performance on the remaining problems.

✉ Gerald Gamrath
gamrath@zib.de

Thorsten Koch
koch@zib.de

Alexander Martin
alexander.martin@math.uni-erlangen.de

Matthias Miltenberger
miltenberger@zib.de

Dieter Weninger
dieter.weninger@math.uni-erlangen.de

1 Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany

2 FAU Erlangen-Nürnberg, Cauerstr. 11, 91058 Erlangen, Germany

Mathematics Subject Classification Primary 90C11 · 90C10; Secondary 90-04 · 90-08 · 90C90

1 Introduction

In order to eliminate redundant information and strengthen the formulation of an integer program, solvers apply a number of techniques before the first linear programming relaxation of an instance is solved. This 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 often not only applied before solving the linear programming relaxation at the root node in a branch-and-bound tree but may also be performed in a reduced fashion, called node presolving, at other nodes of the tree. In this paper, however we concentrate on techniques that we apply in the very first step of the solution process.

Presolving has been applied for solving linear and mixed integer programming problems for decades. Brearly et al. [13] and Williams [27] discussed bound tightening, row elimination, and variable fixings in mathematical programming systems, while Andersen and Andersen [5] published presolving techniques in the context of linear programming. In addition, presolving techniques on zero-one inequalities have been studied by Guignard and Spielberg [18], Johnson and Suhl [21], Crowder et al. [14], and Hoffman and Padberg [19]. Williams [28] developed a projection method for the elimination of integer variables and Savelsbergh [25] investigated preprocessing and probing techniques for mixed integer programming problems. An overview of different presolving techniques can be found in the books of Nemhauser [24] and Wolsey [29], in Fügenschuh and Martin [17] and in Mahajan [23]. Details on implementing presolving techniques effectively within a mixed integer linear programming solver are presented in Suhl and Szymanski [26], Atamtürk and Savelsbergh [7], and Achterberg [2].

The impact of presolving on the entire solution process of mixed integer linear programming problems was published in Bixby and Rothberg [10]. By disabling root presolving, a mean performance degradation of about a factor of ten was detected. Only cutting planes had a bigger influence on the solving process.

This paper is organized as follows. In Sect. 2 we present our notation. Section 3 describes a presolving technique we call *Stuffing Singleton Columns*, where continuous variables with only one non-zero coefficient in the constraint matrix are fixed at a suitable bound. In Sect. 4 we characterize a column based method called *Dominating Columns* working on a partial order. Using this dominance relation new fixings and bounds can be derived. Finally, in Sect. 5, a technique based on *Connected Components* is presented that allows to split the whole problem into subproblems which can be solved independently. In Sect. 6 we show computational results for all three presolving techniques with the state-of-the-art non-commercial MIP solver SCIP [3] on supply chain management instances and a test set comprised of all benchmark instances from the last three MIPLIB versions [4,9,22]. We compare the achieved

reductions as well as the overall performance improvements in Sect. 6 and close our paper with some conclusions in Sect. 7.

2 Notation and basics

Consider a mixed integer program (*MIP*)

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

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

We use the notation $A_{\cdot j}$ for column j of the matrix A and $A_{i\cdot}$ to denote the row vector i. The value in the i-th row and j-th column of A, is called a_{ij}.

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

In [13] a procedure for tightening bounds of variables was published. Important also for our algorithms is the so-called *maximal* (2) and *minimal activity* (3) of a linear constraint $A_{r\cdot}x \le b_r$:

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

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

L_r may be $-\infty$ and U_r may be $+\infty$. Obviously, $L_r \le A_{r\cdot}x \le U_r$ holds. By using the minimal activity L_r, it is possible to calculate upper bounds u^*_j and lower bounds ℓ^*_j for variable x_j. For all feasible solutions x the inequalities

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

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

hold. Hence we obtain potentially new bounds by

$$u^*_j = \min\left\{u_j, \min_{\forall r, a_{rj} > 0}\{u'_{rj}\}\right\} \quad \text{and} \quad \ell^*_j = \max\left\{\ell_j, \max_{\forall r, a_{rj} < 0}\{\ell'_{rj}\}\right\}.$$

For integer variables we may also apply rounding and use $\lfloor u^*_j \rfloor$ or $\lceil \ell^*_j \rceil$ instead. Thus, we may assume that integer variables have integer bounds. However, we do not require that bound tightening is applied prior to the methods described in the following sections.

3 Stuffing singleton columns

A singleton column is a column of the matrix A with $|\text{supp}(A_{\cdot j})| = 1$. In this section we analyze a set of singleton columns of continuous variables x_j within a row r and try to fix them at one of its bounds.

To illustrate the basic idea, consider the linear programming relaxation of the binary knapsack problem. Items, each with a certain profit and size, need to be selected to be put into a knapsack of a given capacity. In contrast to the binary knapsack problem, it is possible to pack any fraction of an item. An optimal solution can be obtained by sorting the items by decreasing profit/size ratio and selecting them in this order until the knapsack capacity is reached [15]. Transferring this idea to a general MIP of the form (1) causes two difficulties. Integer variables are present and variables usually appear in more than one row. So we cannot simply proceed in the same manner as in the linear programming relaxation of the binary knapsack problem, but need to modify the idea as described in the following.

Suppose there is a column j of problem (1) with $c_j \geq 0$ satisfying $a_{rj} \geq 0$ for all rows $r \in \{1, \ldots, m\}$. If $\ell_j > -\infty$, we can set variable x_j to its lower bound. If $\ell_j = -\infty$ and $c_j > 0$, then the problem is unbounded. The appropriate argument applies to a column j with $c_j \leq 0$ and $a_{rj} \leq 0$ for all rows r. If $u_j < \infty$, we can set variable x_j to its upper bound. In case $u_j = \infty$ and $c_j < 0$, the problem is unbounded. This presolving technique is called *duality fixing* [17]. Thus, duality fixing already covers the cases where $c_j/a_{rj} \geq 0$. We use additional information about the rows in order to treat the remaining cases where $c_j/a_{rj} < 0$.

Let us first focus on the case $a_{rj} > 0$ and $c_j < 0$. For a given row r, consider the set of variable indexes

$$J(r) = \{j \in \{1, \ldots, n\} \mid x_j \in \mathbb{R} \wedge |\text{supp}(A_{\cdot j})| = 1 \wedge a_{rj} > 0 \wedge c_j < 0\}. \quad (6)$$

Furthermore, we define the following restricted maximal activity, which is similar to the maximal activity (2) of row r except that continuous singleton columns x_j with $j \in J(r)$ are considered to be at their lower bounds.

$$\tilde{U}_r = \sum_{j \in J(r)} a_{rj}\ell_j + \sum_{\substack{j \notin J(r) \\ a_{rj} > 0}} a_{rj}u_j + \sum_{\substack{j \notin J(r) \\ a_{rj} < 0}} a_{rj}\ell_j \quad (7)$$

The values $\tilde{U}_r$ can now be used to determine optimal variable values for singleton columns. First, we sort the indices $j \in J(r)$ by their cost/size ratios c_j/a_{rj} in non-decreasing order. Let s be the first index in this order. Define $\Delta = a_{rs}(u_s - \ell_s)$ to be the difference in the contribution of variable x_s to constraint r when setting x_s to its upper instead of its lower bound. If $\Delta \leq b_r - \tilde{U}_r$, x_s can be set to its upper bound. After this step, s is removed from $J(r)$, $\tilde{U}_r$ is updated by adding Δ and the procedure is iterated.

Theorem 1 *Given a MIP of the form (1), some row r, the index set $J(r)$ as defined in (6), $\tilde{U}_r$ as defined in (7), and the index $s \in J(r)$ with the smallest ratio c_s/a_{rs}. If $\Delta \leq b_r - \tilde{U}_r$ then $x_s = u_s$ holds for at least one optimal solution.*

Proof Let x^* be an optimal solution with $x_s^* < u_s$. If $x_j^* = \ell_j$ for all $j \in J(r)\backslash\{s\}$, then a new solution x' constructed by setting x_s to u_s is feasible because

$$\sum_{\forall j} a_{rj} x'_j = \sum_{\forall j, j \neq s} a_{rj} x'_j + a_{rs} u'_s \leq \tilde{U}_r + \Delta \leq b_r.$$

Additionally, the objective function value improves because $c_s < 0$. This contradicts our assumption of x^* being optimal. Hence there exists a $j \in J(r)\backslash\{s\}$ with $x_j^* > \ell_j$. In this case we can construct a new solution x' in which we decrease the value of x_j^* to x'_j while at the same time increasing the value of x_s such that $A_{r\cdot} x' = A_{r\cdot} x^*$. In particular, $a_{rs}(x'_s - x_s^*) = a_{rj}(x_j^* - x'_j)$ holds. The change of the objective function can be estimated by

$$\begin{aligned}
c_s x'_s + c_j x'_j &= c_s x_s^* + c_j x_j^* + c_s(x'_s - x_s^*) - c_j(x_j^* - x'_j) \\
&= c_s x_s^* + c_j x_j^* + \frac{c_s a_{rs}}{a_{rs}}(x'_s - x_s^*) - \frac{c_j a_{rj}}{a_{rj}}(x_j^* - x'_j) \\
&\leq c_s x_s^* + c_j x_j^* + \frac{c_s a_{rs}}{a_{rs}}(x'_s - x_s^*) - \frac{c_s a_{rj}}{a_{rs}}(x_j^* - x'_j) \\
&= c_s x_s^* + c_j x_j^* + \frac{c_s}{a_{rs}}(a_{rs}(x'_s - x_s^*) - a_{rj}(x_j^* - x'_j)) \\
&= c_s x_s^* + c_j x_j^*.
\end{aligned}$$

If $x'_s = u_s$, we proved the theorem. Otherwise, $x'_j = \ell_j$ holds. Applying this argument iteratively results in an optimal solution with $x'_s = u_s$ or $x'_j = \ell_j$ for all $j \in J(r)\backslash\{s\}$. But as we have shown before, the latter case contradicts the optimality of x'. □

A similar procedure is followed where $a_{rj} < 0$ and $c_j > 0$. We define

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

and

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

Now, we begin with the index $s \in J(r)$ corresponding to the greatest ratio c_j / a_{rj}. If $\Delta \geq b_r - \tilde{L}_r$, x_s can be set to its upper bound. We update the set $J(r)$ and the value $\tilde{L}_r$ according to their definition and repeat the process.

Sorting the ratios takes $O(|J(r)| \log |J(r)|)$ and computing whether the variables can be set to the upper or lower bound requires $O(|J(r)|)$. Furthermore, the size of $J(r)$ is usually small and hence the algorithm does not deteriorate the performance on instances where no or only very few reductions are found. This is especially true for most MIPLIB instances. For certain practical problems, such as supply chain management, stuffing singleton columns may, however, find fixings quite often (see Sect. 6).

4 Dominating columns

This presolving technique is based on a $\leq$-relation between the coefficients of two variables. Because this relation is reflexive, antisymmetric and transitive, it defines a partial order (poset). The relation causes a consecutive behavior of the variable values and can thus be seen as a dominance relation. The idea of exploiting a dominance relation between variables for presolving is not new. Andersen and Andersen [5] used dominating columns in the presolving of linear programming problems and Borndörfer [12] applied this technique in the context of set partitioning problems. In addition, Babayev and Mardanov [8] and Zhu and Broughan [31] 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.

4.1 Dominance relation

Definition 1 Let a MIP of the form (1) with two different variables x_j and x_i of the same type, i.e., binary, integer or continuous, be given. We say x_j dominates x_i ($x_j \succ x_i$), if

(i) $c_j \leq c_i$ and
(ii) $a_{rj} \leq a_{ri}$ for every constraint r.

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

The following example illustrates Definition 1.

Example 1

$$\begin{aligned}
\min\ & -2x_1 -1x_2 +1x_3 +1x_4 -2x_5 +1x_6 \\
\text{s.t.}\ & +2x_1 +3x_2 -2x_3 \leq 1 \\
& +1x_2 -2x_3 -1x_4 -3x_5 +1x_6 \leq -11 \\
& -1x_3 +1x_4 +2x_5 +3x_6 \leq 5 \\
& +1x_4 -2x_5 -1x_6 \leq 1.5 \\
& 0 \leq x_1, x_2 \leq 4, \quad 1 \leq x_3, x_4 \leq 3.5 \\
& x_1, x_2 \in \mathbb{Z}, \quad x_3, x_4 \in \mathbb{R}, \quad x_5, x_6 \in \{0, 1\}
\end{aligned}$$

In this example $x_1 \succ x_2$, $x_3 \succ x_4$ and $x_5 \succ x_6$ and the optimal solution is $x_1 = 4, x_2 = 0, x_3 = 3.5, x_4 = 1, x_5 = 1, x_6 = 0$ with value -5.5.

Example 1 raises suspicion that one of the variables involved in a dominance relation is at one of its bounds in the optimal solution. Indeed, this is a general property of the dominance relation that we will prove in the following. To achieve this, we first show that increasing the dominating variable and decreasing the dominated variable by the same amount preserves feasibility and optimality, provided 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^* with*

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

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

Proof For every constraint $A_{r\cdot}x \leq b_r$, we get

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

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

This leads us to the following theorem, stating 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 2 *Let $x_j \succ x_i$, then there always exists an optimal solution x^* for (1) with*

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

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^* by defining $\alpha = \min\{\bar{x}_i - \ell_i, u_j - \bar{x}_j\}$ and applying Lemma 1. Since $\bar{x}$ is optimal and $c^T x^* \leq c^T \bar{x}$, x^* is optimal. By definition of α, also $x_j^* = u_j \vee x_i^* = \ell_i$ holds. □

4.2 Predictive bound analysis

Based on Theorem 2 we 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 of one variable x_t.

Definition 2 Let a linear constraint $A_{r\cdot}x \leq b_r$ of (1) and a variable x_t be given. We denote by

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

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

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

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

Definition 2 is now used to define specific functions, which predict the bounds of variable x_s depending on the value of another variable x_t. We call this approach predictive bound analysis.

Definition 3 Let (1) and two variables x_s and x_t be given. We define the functions

$$\begin{aligned}
\text{MAXL}_s^t(x_t) &= \max_{r=1,\ldots,m} \left\{ \frac{b_r - L_r^t(x_t) + a_{rs} u_s}{a_{rs}} \mid a_{rs}, a_{rt} < 0 \right\}, \\
\text{MAXU}_s^t(x_t) &= \max_{r=1,\ldots,m} \left\{ \frac{b_r - U_r^t(x_t) + a_{rs} \ell_s}{a_{rs}} \mid a_{rs}, a_{rt} < 0 \right\}, \\
\text{MINL}_s^t(x_t) &= \min_{r=1,\ldots,m} \left\{ \frac{b_r - L_r^t(x_t) + a_{rs} \ell_s}{a_{rs}} \mid a_{rs}, a_{rt} > 0 \right\} \text{ and} \\
\text{MINU}_s^t(x_t) &= \min_{r=1,\ldots,m} \left\{ \frac{b_r - U_r^t(x_t) + a_{rs} u_s}{a_{rs}} \mid a_{rs}, a_{rt} > 0 \right\}.
\end{aligned}$$

$\text{MINL}_s^t(x_t)$ 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 is used. Therefore, each constraint gives an upper bound for x_s subject to the value of x_t. Minimizing over these bounds, $\text{MINL}_s^t(x_t)$ gives the tightest implied upper bound on x_s as a function of the value of x_t. Analogously, $\text{MAXL}_s^t(x_t)$ gives the tightest implied lower bound on x_s as a function of the value of x_t.

The other two functions $\text{MAXU}_s^t(x_t)$ and $\text{MINU}_s^t(x_t)$ take into account the maximal instead of the minimal activity. Since the maximal activity is the worst-case when regarding feasibility of a $\leq$-constraint, we obtain a larger lower and a smaller upper bound for x_s subject to x_t. This range of values for x_s is feasible for all constraints, independent of the other variables. It may, however, exceed the variable's domain.

In the following, we assume $\text{MAXL}_s^t(x_t)$, $\text{MAXU}_s^t(x_t)$, $\text{MINL}_s^t(x_t)$, and $\text{MINU}_s^t(x_t)$ to be finite, in particular, that there are rows satisfying the requirements on the coefficients as demanded in Definition 3.

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 is at its lower bound and vice versa.

Lemma 2 *$\text{MAXL}_s^t(x_t)$, $\text{MAXU}_s^t(x_t)$, $\text{MINL}_s^t(x_t)$ and $\text{MINU}_s^t(x_t)$ are strictly monotonically decreasing functions, i.e., for $\ell_t \leq x_t' < x_t'' \leq u_t$ holds*

$$\begin{aligned}
\text{MAXL}_s^t(x_t') &> \text{MAXL}_s^t(x_t''),\\
\text{MAXU}_s^t(x_t') &> \text{MAXU}_s^t(x_t''),\\
\text{MINL}_s^t(x_t') &> \text{MINL}_s^t(x_t'') \text{ and}\\
\text{MINU}_s^t(x_t') &> \text{MINU}_s^t(x_t'').
\end{aligned}$$

Proof We only prove the first inequality, the others can be shown using a similar procedure. Let $\tilde{r}$ be one row defining the maximum in the computation of $\text{MAXL}_s^t(x_t'')$. Since $L_{\tilde{r}}^t(x_t'') - L_{\tilde{r}}^t(x_t') = a_{\tilde{r}t}(x_t'' - x_t')$ and $a_{\tilde{r}s}, a_{\tilde{r}t} < 0$ by the definition of $\text{MAXL}_s^t(x_t)$, the following holds:

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

□

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

Theorem 3 *Let two continuous variables x_j, x_i of (1) and $x_j \succ x_i$ be given, then the bounds*

(i) $x_j \leq \text{MINL}_j^i(\ell_i)$,
(ii) $x_i \geq \text{MAXL}_i^j(u_j)$,
(iii) $x_j \geq \min\{u_j, \text{MAXL}_j^i(\ell_i)\}$,
(iv) $x_i \leq \max\{\ell_i, \text{MINL}_i^j(u_j)\}$,
(v) *If $c_j \leq 0$ then* $x_j \geq \min\{u_j, \text{MINU}_j^i(\ell_i)\}$ *and*
(vi) *If $c_i \geq 0$ then* $x_i \leq \max\{\ell_i, \text{MAXU}_i^j(u_j)\}$

hold for at least one optimal solution.

Proof (i) By Definition 3, a_{rj} and a_{ri} are positive for all rows regarded for the computation of $\text{MINL}_j^i(\ell_i)$. Therefore setting x_i to ℓ_i does not change the minimal activity (3) of the row. Thus by (4) follows $\text{MINL}_j^i(\ell_i)$ is an upper bound of x_j.

(ii) By Definition 3, a_{rj} and a_{ri} are negative for all rows regarded for the computation of $\text{MAXL}_i^j(u_j)$. Therefore setting x_j to u_j does not change the minimal activity (3) of the row. Thus by (5) follows $\text{MAXL}_i^j(u_j)$ is a lower bound of x_i.

(iii) We only treat the interesting case $\ell_j \leq \text{MAXL}_j^i(\ell_i)$. By Definition 3, there exists one row r with $a_{rj}\text{MAXL}_j^i(\ell_i) + L_r^i(\ell_i) - a_{rj}u_j = b_r$. Let x^* be an optimal solution with $\alpha = \min\{u_j, \text{MAXL}_j^i(\ell_i)\} - x_j^* > 0$. Assuming $x_i^* = \ell_i$, then $a_{rj}\text{MAXL}_j^i(\ell_i) + L_r^i(\ell_i) - a_{rj}u_j = b_r < a_{rj}x_j^* + L_r^i(\ell_i) - a_{rj}u_j$ since $a_{rj} < 0$. This leads to a contradiction, because x^* was chosen to be feasible. Thus $x_i^* = \ell_i + \beta$ with $\beta > 0$ and $a_{rj}x_j^* + a_{ri}(\ell_i + \beta) + L_r - a_{rj}u_j - a_{ri}u_i \leq b_r$. Together with $a_{rj}(x_j^* + \alpha) + L_r^i(\ell_i) - a_{rj}u_j \geq a_{rj}\text{MAXL}_j^i(\ell_i) + L_r^i(\ell_i) - a_{rj}u_j = b_r$ we get the inequality

$$a_{rj}x_j^* + a_{ri}(\ell_i + \beta) + L_r - a_{rj}u_j - a_{ri}u_i \leq a_{rj}(x_j^* + \alpha) + L_r^i(\ell_i) - a_{rj}u_j,$$

resulting in $\beta \geq \alpha \cdot a_{rj}/a_{ri}$ with $a_{rj}/a_{ri} \geq 1$. By Lemma 1, we can increase x_j^* by α and decrease x_i^* by α without loosing feasibility or optimality.

(iv) We only treat the interesting case $\text{MINL}_i^j(u_j) \leq u_i$. By Definition 3, there exists one row r with $a_{ri}\text{MINL}_i^j(u_j) + L_r^j(u_j) - a_{ri}\ell_i = b_r$. Let x^* be an optimal solution with $\alpha = x_i^* - \max\{\ell_i, \text{MINL}_i^j(u_j)\} > 0$. Assuming $x_j^* = u_j$, then $a_{ri}\text{MINL}_i^j(u_j) + L_r^j(u_j) - a_{ri}\ell_i = b_r < a_{ri}x_i^* + L_r^j(u_j) - a_{ri}\ell_i$ since $a_{ri} > 0$. This leads to a contradiction, because x^* was chosen to be optimal, and therefore also feasible. Thus $x_j^* = u_j - \beta$ with $\beta > 0$ and $a_{rj}(u_j - \beta) + a_{ri}x_i^* + L_r - a_{rj}\ell_j - a_{ri}\ell_i \leq b_r$. Together with $a_{ri}(x_i^* - \alpha) + L_r^j(u_j) - a_{ri}\ell_i \geq a_{ri}\text{MINL}_i^j(u_j) + L_r^j(u_j) - a_{ri}\ell_i = b_r$ we get the inequality

$$a_{rj}(u_j - \beta) + a_{ri}x_i^* + L_r - a_{rj}\ell_j - a_{ri}\ell_i \leq a_{ri}(x_i^* - \alpha) + L_r^j(u_j) - a_{ri}\ell_i$$

yielding $\beta \geq \alpha \cdot a_{ri}/a_{rj}$ with $a_{ri}/a_{rj} \geq 1$. By Lemma 1, we can decrease x_i^* by α and increase x_j^* by α without loosing feasibility or optimality.

(v) We only treat the interesting case $\ell_j \leq \text{MINU}_j^i(\ell_i)$. 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^* with $x_j^* < \min\{u_j, \text{MINU}_j^i(\ell_i)\}$. Let $\alpha_j = \min\{u_j, \text{MINU}_j^i(\ell_i)\} - x_j^*$, $\alpha_i = x_i^* - \ell_i$ and $\alpha = \min\{\alpha_i, \alpha_j\}$. By Lemma 1, we can increase x_j^* by α and decrease x_i^* by α without loosing feasibility or optimality. If $\alpha = \alpha_j$ 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^* with $x_i^* = \ell_i$. Now, we show that $\bar{x}$ with $\bar{x}_j = \min\{u_j, \text{MINU}_j^i(\ell_i)\}$, $\bar{x}_i = \ell_i$ and $\bar{x}_k = x_k^*$ for $k \neq j, i$ is also an optimal solution. Because x^* 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 $c^T\bar{x} \leq c^T x^*$ from $\bar{x}_j > x_j^*$ and $c_j \leq 0$.

(vi) We only treat the interesting case $\text{MAXU}_i^j(u_j) \leq u_i$. 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^* with $x_i^* > \max\{\ell_i, \text{MAXU}_i^j(u_j)\}$. Let $\alpha_i = x_i^* - \max\{\ell_i, \text{MAXU}_i^j(u_j)\}$, $\alpha_j = u_j - x_j^*$, and $\alpha = \min\{\alpha_i, \alpha_j\}$. By Lemma 1, we can decrease x_i^* by α and increase x_j^* 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^* with $x_j^* = u_j$. Now, we show that $\bar{x}$ with $\bar{x}_i = \max\{\ell_i, \text{MAXU}_i^j(u_j)\}$, $\bar{x}_j = u_j$ and $\bar{x}_k = x_k^*$ for $k \neq i, j$ is also an optimal solution. Because x^* 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 $c^T\bar{x} \leq c^T x^*$ from $\bar{x}_i < x_i^*$ and $c_i \geq 0$. □

Whenever in Theorem 3, (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, and we do not need to pay attention to rounding $\text{MAXL}_s^t(x_t)$, $\text{MAXU}_s^t(x_t)$, $\text{MINL}_s^t(x_t)$ and $\text{MINU}_s^t(x_t)$ for integer variables, we summarize the fixing criteria.

Corollary 1 *Let a MIP of form (1) be given as well as two variables x_j, x_i of the same type, i.e., binary, integer or continuous, with $x_j \succ x_i$. In the following cases, we can fix a variable while preserving at least one optimal solution.*

(i) $\text{MAXL}_j^i(\ell_i) \geq u_j \Rightarrow x_j$ *can be set to* u_j.
(ii) $\text{MINL}_i^j(u_j) \leq \ell_i \Rightarrow x_i$ *can be set to* ℓ_i.
(iii) $c_j \leq 0$ *and* $\text{MINU}_j^i(\ell_i) \geq u_j \Rightarrow x_j$ *can be set to* u_j.
(iv) $c_i \geq 0$ *and* $\text{MAXU}_i^j(u_j) \leq \ell_i \Rightarrow x_i$ *can be set to* ℓ_i.

We now apply Corollary 1 on Example 1. First we need to calculate some conditional activities $L_1^1(u_1) = 1, L_2^4(\ell_4) = -11, U_3^6(\ell_6) = 4.5, U_4^5(u_5) = 1.5$, which allows us to determine the values $\text{MINL}_2^1(u_1) = 0$, $\text{MAXL}_3^4(\ell_4) = 3.5$, $\text{MINU}_5^6(\ell_6) = 1.25$ and $\text{MAXU}_6^5(u_5) = 0$. Thus we can set x_2 and x_6 to their lower bounds and x_3 and x_5 to their upper bounds.

The following criteria can be used instead of Corollary 1. 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 many problems are modeled using non-negative variables.

Corollary 2 *Given a MIP of the form (1) and two variables x_j, x_i of the same type, i.e., binary, integer or continuous, with $x_j \succ x_i$. In the following cases, we can fix a variable while preserving at least one optimal solution.*

(i) $\text{MAXL}_i^j(u_j) \geq \ell_i \Rightarrow x_j$ can be set to u_j.
(ii) $\text{MINL}_j^i(\ell_i) \leq u_j \Rightarrow x_i$ can be set to ℓ_i.
(iii) $c_j \leq 0$ and $\text{MINU}_i^j(u_j) \geq \ell_i \Rightarrow x_j$ can be set to u_j.
(iv) $c_i \geq 0$ and $\text{MAXU}_j^i(\ell_i) \leq u_j \Rightarrow x_i$ can be set 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

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

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). □

4.3 Utilize conflict information for binary variables

For binary variables we can use information from a conflict graph [6] to fix 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 is 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 4 (i) *Let two binary variables x_j, x_i of (1) with $x_j \succ x_i$ and $v_j v_i \in E$ be given, then x_i can be set to 0.*
(ii) *Let two binary variables x_j, x_i of (1) with $x_j \succ x_i$ and $\bar{v}_j \bar{v}_i \in E$ be given, then x_j can be set to 1.*

Proof (i) With two binary variables, four variable assignments are possible. Because $x_j = 1 \wedge x_i = 1$ is not allowed due to $v_j v_i \in E$, 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 $x_j = 0 \wedge x_i = 0$ remain. 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

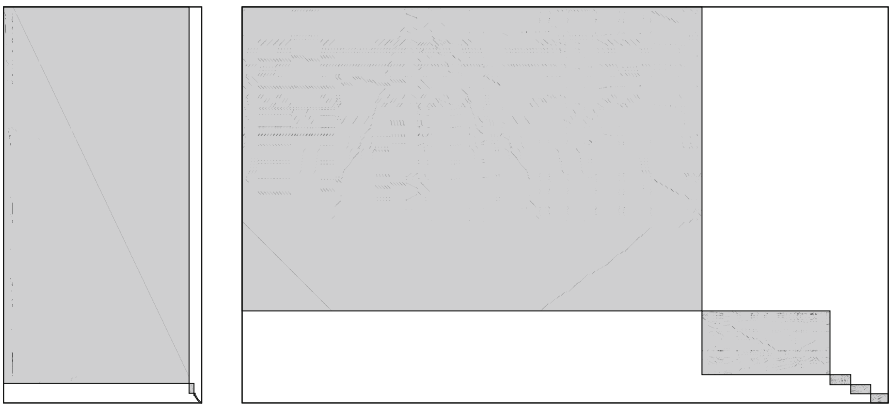
The complexity of an algorithm that operates on a partial order (poset) is mainly determined by the width w of the poset. w is defined to be the maximum cardinality of an anti-chain, which is a subset of mutually incomparable or non-dominating elements. In [16] an algorithm was introduced that sorts a width-w poset of size n in $O(n(w + \log n))$. Its representation has size $O(wn)$ and permits retrieval of the relation between any two elements in time $O(1)$. Since we cannot assume to verify the relation between any two elements in $O(1)$ and w is usually large in our case, we decided to go for it heuristically. This approach works well in practice, which can be seen in the computational results of Sect. 6. Let $R^{=} \subseteq \{1, \ldots, m\}$ be the set of row indices of equalities and $C^{=} = \{j \in \{1, \ldots, n\} \mid a_{kj} \neq 0, k \in R^{=}\}$ be the set of columns of matrix $A \in \mathbb{R}^{m \times n}$ having non-zero entries within equalities. $C^{\leq} = \{1, \ldots, n\} \backslash C^{=}$ is the set of columns having only non-zeros within inequalities. The approach consists of two stages. First we sub-divide $C^{=}$ into different parallel classes $C^{=}_p$ concerning the non-zero entries within equalities. The detection of the parallel classes is performed by an algorithm [11] developed for detecting parallel

columns in $O(\sum_{k \in R^=} |\text{supp}\,(A_{k.})| \log |\text{supp}\,(A_{k.})|)$. Then we compare all columns of every $C^=_p$ in $O(m \cdot \binom{|C^=_p|}{2})$. The first stage guarantees to find all dominance relations within $C^=$. The second stage considers only columns of $C^\leq$ and takes advantage of the sparsity of A via analyzing the rows by increasing $|\text{supp}\,(A_{r.})|$ in $O(m \cdot \binom{|\text{supp}(A_{r.})|}{2})$. After one row was executed, the processed columns therein are not compared to other columns anymore. Thus the number of columns which should be compared is usually much smaller than n. In cases where we have to compare a lot of columns, there is a mechanism which monitors the number of fixings per number of paired comparisons. If not enough fixings are found, then this row will not be investigated further.

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 at the beginning, it might decompose after some rounds of presolving, e.g., because constraints connecting independent problems are detected to be redundant and can be removed. Figure 1 depicts the constraint matrices of two real-world instances at some point during presolving, reordered in a way such that independent components can easily be identified.



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

Fig. 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*

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 [20]. 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

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

Given this graph, we identify connected components using depth first search. By definition, 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 [20], which is 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

[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.

insight is typically lost. Therefore, it is important to choose the parameters in a way such that this scenario is avoided.

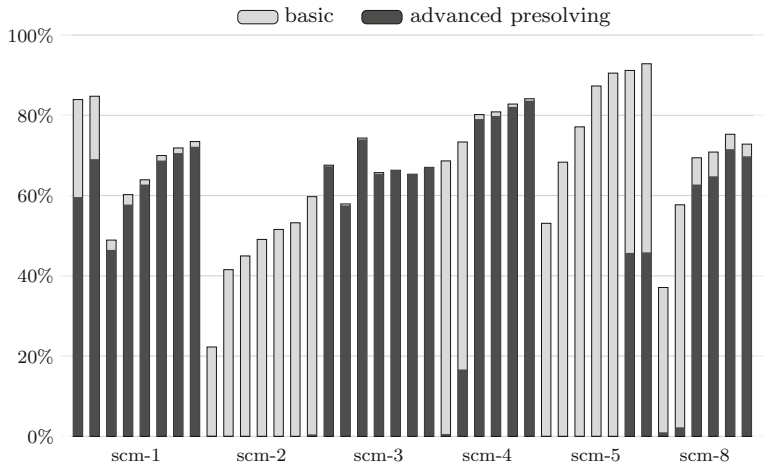
6 Computational results

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.

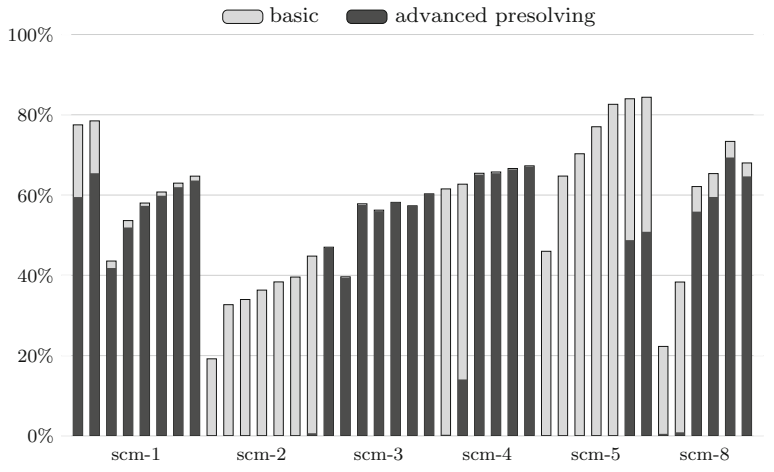
We implemented three new presolving techniques, which were already included in the SCIP 3.0 release. The stuffing algorithm is implemented within the dominated columns presolver, because it makes use of the same data structures.

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, which is the union of MIPLIB 3 [9], MIPLIB 2003 [4], and the benchmark set of MIPLIB 2010 [22]. For the experiments, we used the development version 3.0.1.2 of SCIP [3] (git hash `7e5af5b`) with SoPlex [30] 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. SCIP triggers a so-called *restart* if during root node processing a certain fraction of integer variables has been fixed. In this case presolving is also applied once more. 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 (Fig. 2a) and remaining constraints (Fig. 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 few 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



(a) variables after presolve



(b) constraints after presolve

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

stuffing, respectively. This increases the shifted geometric mean of the presolving time from 2.12 to 3.18 s, 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 s in the shifted geometric mean. This slight improvement can also be registered in the performance diagram shown in Fig. 3.

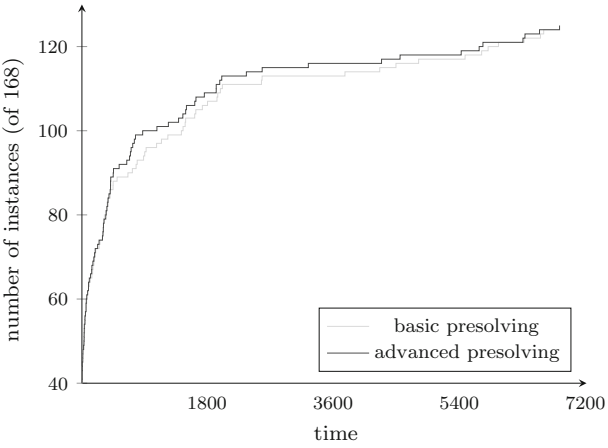


Fig. 3 Performance diagram for the MMM test set. The graph indicates the number of instances solved within a certain time

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

Test set	Basic presolving					Advanced presolving				
	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

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

However, many of the instances in the MMM test set do not contain a structure that can be used by the new presolving techniques: they are able to find reductions for less than a quarter of the instances. 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 % more variables and 4.5 % more constraints are removed from the problem. This is depicted in Fig. 4. The majority of the variables is removed by the dominating columns presolver, which 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

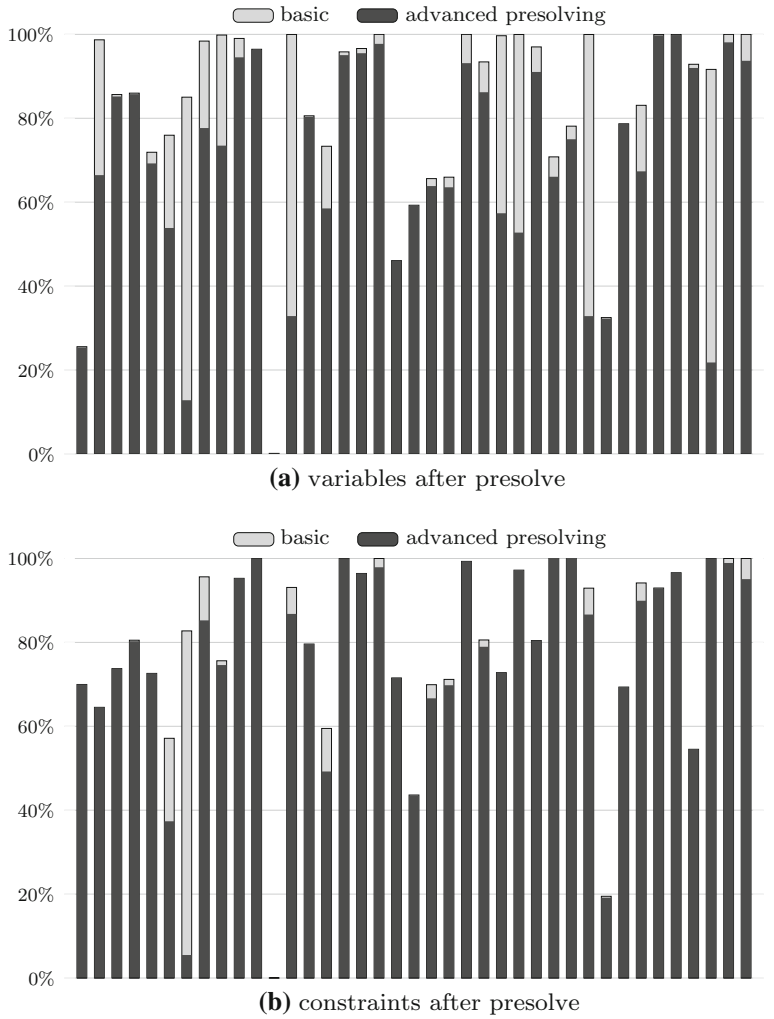


Fig. 4 Size of the presolved instances relative to the original number of variables and constraints for all instances from the MMM test set where the new presolving techniques find reductions

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 academic MIP solver SCIP. At first, they were developed with a focus on a set of real-world supply chain manage-

ment 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, 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 do 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 incorporates 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.

Acknowledgments The authors would like to thank the anonymous reviewers for helpful comments on the paper. The work for this article has been partly conducted within the *Research Campus Modal* funded by the German Federal Ministry of Education and Research (fund number 05M14ZAM). Finally, we thank the DFG for their support within Projects A05, B07, and Z02 in CRC TRR 154.

Appendix A: Detailed computational results

In this appendix, we present detailed results of our computational experiments presented in Sect. 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 [22], the gap for a given primal bound pb and dual bound db is computed by the following formula:

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

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

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
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	16,001	10,308	13,561	8090	**1.0**	*0.2 %*	**11,053**	**6746**	1.2	+0.00	*0.1 %*	5	0.06	14	0	1	0.06	10/11	2490	1341
snp-001-03	18,071	10,973	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	36,874	22,518	22,213	12,078	**1.8**	*21.4 %*	**21,302**	**11,692**	3.3	**+6.23**	***10.1** %*	6	0.23	27	122	1	1.28	8/9	742	372
snp-001-05	56,586	34,605	36,175	20,076	**4.0**	***29.9** %*	**35,506**	**19,832**	20.6	+0.00	*33.0 %*	8	0.53	49	92	1	16.12	8/10	484	209
snp-001-06	80,237	50,535	56,138	30,707	**12.8**	*5.4 %*	**55,138**	**30,241**	21.4	**+0.00**	***4.1** %*	14	1.53	134	2	1	0.06	10/12	785	395
snp-001-07	104,078	66,598	74,787	41,939	28.8	***190.8** %*	**73,473**	**41,264**	**15.1**	−0.00	*216.2 %*	13	1.77	119	3	1	0.09	12/14	1112	604
snp-001	139,949	90,739	102,813	58,714	82.8	*288.3 %*	**100,989**	**57,726**	**21.5**	−0.00	***197.9** %*	16	2.97	141	3	1	0.12	13/15	1616	930
snp-002-01	12,041	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	25,632	13,576	10,648	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	40,459	21,378	18,192	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	70,164	37,012	34,435	13,444	**1.3**	51.2	**49**	**16**	1.5	**+0.01**	**1.6**	12	0.10	14,467	385	1	0.44	484/485	5003	2016
snp-002-05	115,043	60,685	59,315	23,278	15.4	*0.0 %*	**23**	**17**	**2.2**	**+0.02**	**2.4**	18	0.29	26,180	613	1	0.43	490/490	7000	3089
snp-002-06	149,904	79,517	79,772	31,461	32.0	*0.0 %*	**33**	**33**	**3.1**	**+0.02**	**3.3**	22	0.40	35,193	1658	1	0.46	529/530	8282	3679
snp-002	382,553	209,616	228,477	93,891	117.5	*216.5 %*	**1904**	**1375**	**47.7**	**+0.16**	***0.0** %*	44	2.76	107,214	843	1	33.06	1102/1110	23,303	13,087
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	26,564	15,031	17,458	8454	**0.4**	*12.8 %*	**17,380**	**8416**	0.9	+0.00	***9.7** %*	20	0.31	38	0	1	0.01	0/1	0	0
snp-003-05	42,468	23,489	28,160	13,666	**0.4**	*13.7 %*	**28,146**	**13,660**	0.8	+0.00	***12.0** %*	4	0.16	8	0	1	0.01	0/1	0	0
snp-003-06	47,713	26,674	31,165	15,289	0.5	*12.1 %*	**31,141**	**15,283**	0.6	+0.00	*12.5 %*	4	0.10	18	0	1	0.02	0/1	0	0
snp-003	87,652	47,907	58,750	28,887	**1.1**	***14.9** %*	**58,730**	**28,877**	1.3	+0.00	*17.2 %*	5	0.22	10	0	1	0.04	0/1	0	0
snp-004-02	26,597	13,130	18,257	8079	**0.2**	1.8	**156**	**38**	0.9	**+0.00**	**1.0**	4	0.04	736	0	1	0.60	732/733	17,359	8037

Table 2 continued

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-004-03	58,693	26,637	43,050	16,703	**0.4**	1981.6	**9785**	**3744**	1.6	**+0.01**	**3.1**	3	0.08	738	16	1	0.90	577/578	32,499	12,948
snp-004-04	126,553	49,032	101,464	32,080	**0.9**	*1728*k %	**100,052**	**31,912**	1.4	**+0.01**	*1727*k %	3	0.23	785	55	1	0.08	4/5	604	157
snp-004-05	140,535	53,508	113,625	35,182	**1.2**	*1729*k %	**112,210**	**35,014**	1.6	**+0.01**	*1729*k %	3	0.28	788	55	1	0.09	4/5	604	157
snp-004-06	210,279	75,833	174,116	50,518	**2.1**	*1735k %*	**172,651**	**50,340**	2.6	**+0.02**	*1735k %*	4	0.59	827	48	1	0.14	5/6	618	163
snp-004	321,527	111,563	270,488	75,058	**3.9**	*1745k %*	**268,919**	**74,864**	4.2	**+0.02**	*1744k %*	5	1.10	852	42	1	0.20	5/6	700	179
snp-005-01	13,518	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	34,933	11,631	23,865	7529	**0.4**	3.1	**10**	**9**	1.1	**+0.00**	**1.1**	6	0.07	411	907	1	0.40	248/249	21,987	6989
snp-005-03	83,330	25,458	64,265	17,894	**0.9**	16.1	**10**	**9**	3.2	**+0.00**	**3.4**	11	0.46	1037	1477	1	1.28	314/315	61,082	17,258
snp-005-04	310,163	71,357	270,763	54,960	**6.7**	397.5	**146**	**158**	15.3	**+0.00**	**20.6**	16	3.30	2698	1372	1	4.52	279/280	265,543	53,974
snp-005-05	560,637	121,943	507,447	100,741	55.3	1425.5	**223**	**231**	**40.6**	**+0.00**	**102.5**	21	9.45	2174	1203	1	12.20	370/371	502,652	99,441
snp-005-06	680,745	146,213	620,715	122,807	107.6	*0.0 %*	**311,220**	**71,302**	**82.5**	**+0.00**	*0.0 %*	26	13.75	2352	1094	1	5.91	322/323	304,656	50,296
snp-005	1,182,136	255,629	1,097,503	215,672	399.4	*187.5 %*	**542,064**	**129,989**	**330.7**	**+0.00**	***41.4*** *%*	39	36.11	3039	1108	1	10.92	321/322	549,126	83,910
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	20,840	7743	12,025	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	57,707	27,374	40,053	17,005	**7.4**	*0.0 %*	**36,210**	**15,293**	7.8	−0.00	*0.0 %*	16	0.72	350	591	1	0.38	21/22	2239	1310
snp-008-04	67,128	33,017	47,555	21,578	11.0	*0.1 %*	**43,496**	**19,653**	10.9	−0.00	*0.1 %*	17	1.04	359	565	1	0.44	21/22	2521	1544
snp-008-05	194,751	109,140	146,594	80,078	111.8	*1044 %*	**139,365**	**75,696**	**66.9**	−0.00	*1044 %*	40	9.73	341	656	1	1.50	25/26	6526	4829
snp-008	379,030	210,508	276,023	143,139	**258.1**	*1073 %*	**264,498**	**136,102**	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 3 Detailed computational results on the MMM set

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
10 teams	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	10,757	124	10,617	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	26,871	53,467	26,265	52,555	**4.8**	1081.5	26,265	52,555	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	24,748	3651	24,748	0.3	80.1	3651	24,748	0.3	+0.00	80.6	1	0.03	0	0	1	0.01	0/1	0	0
atlanta-ip	48,738	21,732	17,240	19,083	**1.1**	*8.6%*	17,240	19,083	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	21,600	4964	21,432	4740	4.4	*1.9%*	21,432	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	175,620	4958	145,307	**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

Table 3 continued

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
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	15,293	2539	15269	1920	**0.8**	723.9	**11,238**	**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	13,873	3202	13,837	3151	0.9	*21.6%*	13,837	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	10,000	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	67,732	656	67,076	625	**2.5**	*572.6%*	**64,030**	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	10,404	40,962	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	63,009	507	62,997	472	**0.8**	2556.3	**20,700**	**440**	1.8	+0.00	**300.3**	5	0.83	42,285	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
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

Table 3 continued

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
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	14,163	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	164,547	328,818	15,412	31,207	2.0	405.4	15,412	31,207	2.1	+0.00	407.2	1	0.02	0	0	1	0.02	0/1	0	0
map20	164,547	328,818	15,412	31,207	2.0	446.9	15,412	31,207	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

Table 3 continued

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
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	10,724	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	10,958	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	42,680	2746	13,212	4.9	*18.6%*	2746	13,212	5.0	+0.00	*18.6%*	12	0.12	0	0	1	0.00	0/1	0	0
momentum2	3732	24,237	2774	14,861	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	13,532	56,822	13,151	49,375	203.1	*254.4%*	13,151	49,375	205.3	+0.00	*254.4%*	6	0.42	0	0	1	0.05	0/1	0	0
msc98-ip	21,143	15,850	12,733	14,987	**0.7**	*12.4%*	12,733	14,987	0.9	+0.00	*12.4%*	3	0.07	0	0	1	0.01	0/1	0	0
mspp16	29,280	561,657	4065	524,814	484.4	4265.7	4065	524,814	484.9	+0.00	4244.1	1	3.65	0	0	1	2.26	0/1	0	0
mzzv11	10,240	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
mzzv42z	11,717	10,460	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	22,120	4484	22,120	4453	**1.3**	*9.8%*	**20,602**	4453	2.0	+0.00	*9.3%*	2	0.08	1518	0	1	0.04	0/1	0	0

Table 3 continued

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
n3seq24	119,856	6044	119,856	5950	**11.7**	*10.4%*	119,856	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	28,979	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	20,852	1827	17,320	1.4	*32.3%*	1827	17,320	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	11,402	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	11,915	10,015	11,843	9604	18.5	256.5	11,843	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	23,123	11,495	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	14,115	14,021	12,523	12,767	**3.6**	4496.2	12,523	12,767	4.1	+0.00	4481.6	20	0.54	0	0	1	0.01	0/1	0	0
netdiversion	129,180	119,589	128,968	99,483	16.2	6162.8	128,968	99,483	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	17,956	624,166	17,824	615,190	24.4	*527.6%*	17,824	615,190	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

Table 3 continued

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
nw04	87,482	36	87,454	35	**3.8**	38.2	**46,143**	35	8.3	+0.00	**31.0**	2	5.16	41,311	0	1	0.06	0/1	0	0
opm2-z7-s2	2023	31,798	1896	26,691	3.2	1291.1	1896	26,691	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	63,019	509	62,997	473	**1.4**	1797.3	**20,698**	**441**	2.7	+0.00	**363.4**	5	1.36	42,288	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
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	125,899	522	25,272	36.1	*171k%*	522	25,272	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

Table 3 continued

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
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	21,738	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	14,101	1761	14,099	1637	**0.9**	*7.3%*	**14,067**	**1636**	1.4	−0.03	***6.9%***	3	0.15	31	0	1	0.05	0/1	0	0
sp98ic	10,894	825	10,894	797	**0.8**	*2.5%*	**10,877**	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	204,880	159,488	136,241	96,985	32.6	*inf%*	136,241	96,985	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	73,885	551	67,716	551	**0.9**	*41.2%*	**16,102**	551	1.2	+0.00	***38.5%***	2	0.27	51,614	0	1	0.01	0/1	0	0
tanglegram1	34,759	68,342	34,759	68,342	0.7	907.7	**34,099**	**67,614**	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

Table 3 continued

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
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	30,055	15,706	25,446	15,574	10.2	2945.4	25,446	15,574	10.1	+0.00	2944.6	3	0.15	0	0	1	0.05	0/1	0	0
unitcal_7	25,755	48,939	20,297	38,656	15.3	2412.0	20,297	38,656	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	51,471	47,280	27,488	23,418	2.1	*inf*%	27,488	23,418	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 %

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 s). We use a shift of 10 s for the solving time and 0.01 s 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.

References

1. Aardal, K., Bixby, R.E., Hurkens, C.A.J., Lenstra, A.K., Smeltink, J.W.: Market split and basis reduction: towards a solution of the Cornuéjols–Dawande instances. INFORMS J. Comput. **12**(3), 192–202 (2000)
2. Achterberg, T.: Constraint Integer Programming. PhD thesis, Technische Universität Berlin (2007)
3. Achterberg, T.: SCIP: solving constraint integer programs. Math. Program. Comput. **1**(1), 1–41 (2009)
4. Achterberg, T., Koch, T., Martin, A.: MIPLIB 2003. Oper. Res. Lett. **34**(4), 1–12 (2006)
5. Andersen, E.D., Andersen, K.D.: Presolving in linear programming. Math. Program. **71**, 221–245 (1995)
6. Atamtürk, A., Nemhauser, G.L., Savelsbergh, M.W.P.: Conflict graphs in solving integer programming problems. Eur. J. Oper. Res. **121**(1), 40–55 (2000)
7. Atamtürk, A., Savelsbergh, M.W.P.: Integer-programming software systems. Ann. Oper. Res. **140**, 67–124 (2005)
8. Babayev, D.A., Mardanov, S.S.: Reducing the number of variables in integer and linear programming problems. Comput. Optim. Appl. **3**(2), 99–109 (1994)
9. Bixby, R.E., Ceria, S., McZeal, C.M., Savelsbergh, M.W.P.: An updated mixed integer programming library: MIPLIB 3.0. Optima **58**, 12–15 (1998)
10. Bixby, R.E., Rothberg, E.: Progress in computational mixed integer programming-a look back from the other side of the tipping point. Ann. Oper. Res. **149**, 37–41 (2007)
11. Bixby, R.E., Wagner, D.K.: A note on detecting simple redundancies in linear systems. Oper. Res. Lett. **6**(1), 15–17 (1987)
12. Borndörfer, R.: Aspects of set packing, partitioning, and covering. PhD thesis, Technische Universität Berlin (1998)
13. Brearley, A.L., Mitra, G., Williams, H.P.: Analysis of mathematical programming problems prior to applying the simplex algorithm. Math. Program. **8**, 54–83 (1975)

14. Crowder, H., Johnson, E.L., Padberg, M.: Solving large-scale zero-one linear programming problems. Oper. Res. **31**(5), 803–834 (1983)
15. Dantzig, G.B.: Discrete-variable extremum problems. Oper. Res. **5**(2), 266–277 (1957)
16. Daskalakis, C., Karp, R.M., Mossel, E., Riesenfeld, S., Verbin, E.L.: Sorting and selection in posets. In: SODA '09: Proceedings of the Nineteenth Annual ACM–SIAM Symposium on Discrete Algorithms, pp. 392–401. Society for Industrial and Applied Mathematics, Philadelphia (2009)
17. Fügenschuh, A., Martin, A.: Computational integer programming and cutting planes. In: Aardal, K., Nemhauser, G.L., Weismantel, R. (eds.) Discrete Optimization. Handbooks in Operations Research and Management Science, vol. 12, chap. 2, pp. 69–122. Elsevier, Amsterdam (2005)
18. Guignard, M., Spielberg, K.: Logical reduction methods in zero-one programming: minimal preferred variables. Oper. Res. **29**(1), 49–74 (1981)
19. Hoffman, K.L., Padberg, M.: Improving LP-representations of zero-one linear programs for branch-and-cut. ORSA J. Comput. **3**(2), 121–134 (1991)
20. Hopcroft, J., Tarjan, R.: Algorithm 447: efficient algorithms for graph manipulation. Commun. ACM **16**(6), 372–378 (1973)
21. Johnson, E.L., Suhl, U.H.: Experiments in integer programming. Discrete Appl. Math. **2**(1), 39–55 (1980)
22. Koch, T., Achterberg, T., Andersen, E., Bastert, O., Berthold, T., Bixby, R.E., Danna, E., Gamrath, G., Gleixner, A.M., Heinz, S., Lodi, A., Mittelmann, H., Ralphs, T., Salvagnin, D., Steffy, D.E., Wolter, K.: MIPLIB 2010. Math. Program. Comput. **3**(2), 103–163 (2011)
23. Mahajan, A.: Presolving mixed-integer linear programs. In: Cochran, J.J., Cox, L.A., Keskinocak, P., Kharoufeh, J.P., Smith, J.C. (eds) Wiley Encyclopedia of Operations Research and Management Science, pp. 4141–4149. Wiley, New York (2011)
24. Nemhauser, G.L., Wolsey, L.A.: Integer and Combinatorial Optimization. Wiley, New York (1988)
25. Savelsbergh, M.W.P.: Preprocessing and probing techniques for mixed integer programming problems. ORSA J. Comput. **6**, 445–454 (1994)
26. Suhl, U., Szymanski, R.: Supernode processing of mixed-integer models. Comput. Optim. Appl. **3**(4), 317–331 (1994)
27. Williams, H.P.: A reduction procedure for linear and integer programming models. In: Redundancy in Mathematical Programming. Lecture Notes in Economics and Mathematical Systems, vol. 206, pp. 87–107. Springer, Berlin (1983)
28. Williams, H.P.: The elimination of integer variables. J. Oper. Res. Soc. **43**(5), 387–393 (1992)
29. Wolsey, L.A.: Integer Programming. Wiley, New York (1998)
30. Wunderling, R.: Paralleler und objektorientierter Simplex-Algorithmus. PhD thesis, Technische Universität Berlin (1996)
31. Zhu, N., Broughan, K.: A note on reducing the number of variables in integer programming problems. Comput. Optim. Appl. **8**(3), 263–272 (1997)