

Symmetry Presolving for Mixed-Integer Programs: Binary Structure Preservation and Relations to Orbital Branching

Annika Jäger and Marc E. Pfetsch

*Department of Mathematics, TU Darmstadt, Germany,
{jaegerlpfetsch}@opt.tu-darmstadt.de*

October 15, 2026

Abstract

This paper investigates a presolving method for handling symmetries in mixed-integer programs, based on inequalities computed from so-called Schreier-Sims tables. We show that an iterative application of this method together with merging variables produces an instance for which the symmetry group is trivial. We then prove that the problem structure can be preserved for certain monotone binary problems, e.g., packing and covering integer programs, maximum stable set (or clique) problems and set covering. This results in a changed instance of the same type such that existing black-box solvers can directly be applied. In order to relate this method to symmetry handling methods working in the tree, we show that, for a fixed compatible order of the variables, branch-and-bound trees for presolved binary problems are graph minors of the trees produced by orbital branching.

We showcase our presolving approach via three computational use cases. In the first, for the stable set problem on a set of mixed graphs, it outperforms the version without symmetry handling, while being competitive with the symmetry handling methods implemented in SCIP. For Johnson and coding theory graphs, we significantly outperform both. However, orbital branching performs superior to the presolving routine demonstrating the beneficial effect of dynamic symmetry handling for the stable set problem. In the second use case, we demonstrate a significant speed-up when used in conjunction with the maximum clique solver CliSAT on Johnson graphs. The third use case shows that for Steiner triple system set covering instances, using presolving significantly improves upon the other symmetry handling methods.

Keywords: symmetry handling, Schreier-Sims table, stable set problem, set covering, orbital branching

1 Introduction

Symmetry handling for mixed-integer programs is one important component of modern branch-and-cut optimization solvers and allows for reducing the search space if symmetries are present. Over the last two decades, an abundance of different methods have been developed, from isomorphism pruning [22, 23, 24], fundamental domains [8, 32, 33], orbitopes [4, 17, 18], orbital branching [21, 26, 27, 28], Schreier-Sims inequalities [30], to newer methods [10, 13]; see also the overview article [25] and the computational evaluation in [29]. These methods are usually divided into the ones that add symmetry handling inequalities and the ones that handle symmetries during the branch-and-bound search.

A common property of these methods is that they often do not exploit the particular problem structure beyond the symmetry group. In fact, combining symmetry handling inequalities with problem structure often leads to a very complicated polyhedral description, see, e.g., [13, 15], with the exception of particular structures like packing/partitioning orbitopes [18]. Moreover, depending on the method, more or less work has to be invested into the implementation and checking whether a combination of methods is compatible with each other, or could lead to the elimination of all optimal solutions. In this paper, we propose a way to deal with these two challenges for certain monotone binary problems.

We build on inequalities for general mixed-integer linear problems based on Schreier-Sims tables, which have been proposed by Liberti and Ostrowski [20] as well as Salvagnin [30]. They iteratively add ordering constraints like $x_i \geq x_j$ for symmetric variables x_i, x_j and proceed with the subgroup that stabilizes the so-called leader variable x_i ; see Section 2.1 for a detailed description. We propose a presolving method that iteratively computes symmetries and adds these inequalities until no (formulation) symmetry is left. Indeed, we show in Section 2.1 that this process terminates if we further merge variables for which equality holds. It works well for problems (or solution methods) for which adding $x_i \geq x_j$ does not change the structure like SAT. The approach, however, changes the structure for monotone problems. We therefore develop a general presolving method, consisting of several steps, which adds inequalities as well as fixes and deletes variables. We prove in Section 3 that iteratively applying this method will terminate with a trivial symmetry group.

Then, one key contribution is the observation that for several binary problem classes, this procedure does not change the problem structure (Section 4). This has the advantage that all problem specific techniques can be applied. We consider packing and covering integer programs (Section 4.1), the maximum stable set problem (Section 4.2) and the set cover problem (Section 4.3) as examples. For instance, for stable set problems, the procedure will delete some nodes in the graph (including incident edges) and add edges resulting in a graph with trivial automorphism group; this case is motivated by the observations in [12]. The resulting stable set problem can then be solved by black-box solvers applied to the presolved graphs.

In Section 5 we provide a class of instances requiring $\Omega(\log n)$ iterative applications of presolving until no non-trivial symmetry is present anymore, where n is the number of variables. Moreover, we prove an upper bound of n^2 .

As mentioned above, the two main advantages of our presolving method are that it can be used with black-box solvers and it can allow to preserve and thus exploit problem structure in solvers. One disadvantage is that it is static in the sense that symmetry cannot be used dynamically within the branch-and-bound tree. In order to relate these two different paradigms, we compare presolving to orbital branching, both theoretically and computationally. In Section 6.2, we prove that if we fix the variable order to be compatible with the SST-cuts, the plain branch-and-bound tree for a (one-round) presolved instance is a graph minor of the tree for which SST-cuts have been applied. Moreover, the latter is isomorphic to the tree generated by orbital branching. Thus, in this setting, presolving cannot produce larger trees than orbital branching. This also shows that an advantage of orbital branching can only rely on a different and possibly dynamic choice of the branching variables. In addition, one can apply orbital fixing, which we have not captured in the theoretical results. In the computational experiments of Section 7, our presolving method is outperformed by orbital branching for the stable set problem, while for the set cover problem, our computational results show that the presolving routine outperforms orbital branching.

In Section 7, we demonstrate the effects of the presolving technique via three use cases. First, we test an implementation of the presolving method together with a branch-and-cut solver for stable set problems on a mixed testset of graphs collected from many sources, on Johnson graphs, and graphs arising from coding theory (Section 7.1). We show that the performance of the presolving step plus “black-box” branch-and-cut solver is significantly better than applying the solver without symmetry handling on all testsets. On the mixed graphs, its performance is slightly better than the one of the solver applying symmetry handling implemented in SCIP [11]. For Johnson and especially coding theory graphs, the presolving method is vastly superior. Nevertheless, it turns out that orbital branching is the superior symmetry handling technique especially regarding the number of instances solved. In the second use case, we observe a strong performance improvement of the maximum clique solver CliSAT [31] when applied on the presolved and complemented Johnson graphs. Finally, in the third use case, we show that when applied on so-called Steiner triple system set covering instances, presolving is superior to the state-of-the-art symmetry handling with a speed-up of about 18 %.

A preliminary version of this paper was published in the IPCO proceedings [14]. This full version additionally includes strengthened lower and upper bounds on the number of presolving rounds in Section 5. Moreover, we provide a theoretical comparison to orbital branching in Section 6. The computational results have been updated, also including orbital branching as one variant. Further, Section 7.3 features new computational results for set covering instances while the initial description of SST-presolving for the set cover problem in Section 4.3 was expanded to provide more detail. We also distinguish between binary and mixed-integer problems throughout this article.

2 Preliminaries

In this paper, we consider the following mixed-integer linear problems:

$$\max \{c^\top x : Ax \leq b, x_i \in \mathbb{Z} \text{ for } i \in I, x \in \mathbb{R}^n\}, \quad (\text{P})$$

where $A = (a_{i,j})_{i \in [m], j \in [n]} \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$ and $I \subseteq [n] := \{1, \dots, n\}$. The k th row of A is denoted by a_k .

Let $\mathcal{S}_n$ be the full symmetric group of $[n]$, where $\gamma \in \mathcal{S}_n$ acts on $x \in \mathbb{R}^n$ as $\gamma(x) := (x_{\gamma^{-1}(1)}, \dots, x_{\gamma^{-1}(n)})$. We consider the formulation symmetry of the problem (P) as defined in [29], which we will simply refer to as *symmetry* in the following. The symmetry group Γ of (P) is a subgroup of $\mathcal{S}_n$, consisting of all permutations (also referred to as *symmetries* or *automorphisms*) γ for which there exists a permutation $\delta \in \mathcal{S}_m$ such that $\gamma(c) = c$, $\delta(b) = b$, $\gamma(j) \in I$ for all $j \in I$ and $a_{\delta(i), \gamma(j)} = a_{i,j}$ for all $i \in [m]$, $j \in [n]$. We call two variables x_i, x_j *symmetric* if there exists $\gamma \in \Gamma$ with $\gamma(i) = j$ and call the set of variable indices

$$\text{orb}(i, \Gamma) := \{j \in [n] : \exists \gamma \in \Gamma \text{ s.t. } \gamma(i) = j\}$$

the *orbit* of i . The set of permutations in Γ that keep each element of a set $S \subseteq [n]$ invariant, i.e.,

$$\text{stab}(\Gamma, S) := \{\gamma \in \Gamma : \gamma(i) = i \text{ for all } i \in S\}$$

is the *pointwise stabilizer* of S .

2.1 Schreier-Sims Table Inequalities

One way of handling symmetries in (P) is adding *Symmetry Handling Inequalities* (SHI). One particular case of SHIs are generated using the following *Schreier-Sims Table* (SST) algorithm, see, e.g., [22], that stores a sequence of stabilized indices in L :

(A0) initialize $\Gamma' = \Gamma$ and $L = \emptyset$;

(A1) while there exists $\ell \in [n] \setminus L$ such that $|O_\ell| > 1$ for $O_\ell := \text{orb}(\ell, \Gamma')$:

(A2) append ℓ to L and update Γ' to $\text{stab}(\Gamma', L)$;

The SHIs obtained from this algorithm are the so-called *SST-cuts*

$$x_\ell \geq x_f \quad \text{for } \ell \in L, f \in O_\ell \setminus \{\ell\},$$

where ℓ is called *leader* and f *follower* of ℓ . Liberti and Ostrowski [20] and Salvagnin [30] showed the validity of these cuts as SHIs. For binary variables, an SST-cut implies that whenever a follower is chosen, i.e., $x_f = 1$, its leader has to be chosen as well. We usually refer to a single SST-cut by a pair (ℓ, f) with $\ell \in L$ and $f \in O_\ell \setminus \{\ell\}$ which we call *SST-pair*. Moreover,

$$S(L) := \{(\ell, f) : \ell \in L, f \in O_\ell \setminus \{\ell\}\}$$

denotes the set of all SST-pairs, which we just denote by S if the corresponding L is clear. As discussed in [12], the set of SST-pairs can be computed in polynomial time.

Salvagnin [30] showed that adding all SST-cuts to a mixed-integer program (MIP) completely breaks the symmetry of the initial formulation. Nevertheless, he also demonstrated that this can produce new symmetries. A straightforward approach to obtain a problem formulation with non-trivial symmetry is to iteratively compute symmetry and add SST-cuts. This approach does not seem to have been investigated so far. We show in the following that it terminates, if we additionally merge variables for which equality holds. By *merging* variables x_i and x_j we mean fixing $x_j = 0$, updating the objective value of x_i to $c_i + c_j$ and updating the i th column of A to the sum of the i th and j th column of A .

Lemma 2.1. Consider (P), iteratively compute the symmetry of the problem and add all computed SST-cuts. Further, whenever there is a constraint $x_i \geq x_j$ to be added and $x_i \leq x_j$ is already part of the problem formulation, the variables are merged. Then this procedure terminates after finitely many iterations with a problem with trivial symmetry.

Proof. Assume the statement is false. Since there are only finitely many different SST-cuts and variables that can be merged, there exists an iteration at which the problem still has non-trivial symmetry, all associated SST-cuts are already part of the problem formulation, and no variable merging is applicable. Thus, for the current leader set L , there exists $\ell \in L$. Since $|O_\ell| > 1$, there also exists a symmetry γ with $\gamma(\ell) \neq \ell$. Then also all inequalities $x_\ell \geq x_{\gamma^k(\ell)}$ with $k \in \mathbb{N}$ and $\gamma^k(\ell) \neq \ell$ are already part of the problem formulation. Since O_ℓ is finite, there exists some $p \in \mathbb{N}$ such that $\gamma^p(\ell) = \ell$, which implies $\gamma^{p-1}(\ell) \neq \ell$ because $\gamma(\ell) \neq \ell$. Thus, $x_\ell \geq x_{\gamma^{p-1}(\ell)}$ is part of the problem formulation. Since γ is a symmetry, the constraint $x_{\gamma(\ell)} \geq x_\ell$ obtained by applying γ^{-1} is also part of the problem formulation. But this contradicts that we computed the SST-pair $(\ell, \gamma(\ell))$ and no merging is applicable. $\square$

Therefore, iteratively adding SST-cuts and merging is a presolving method that removes all non-trivial symmetry from a MIP. This works well for problem classes that allow the addition of such constraints and merging without structural change.

One important example is the SAT problem, where an SST-cut $x_i \geq x_j$ can be expressed as the clause $\bar{x}_j \vee x_i$. For SAT, the use of SST clauses and the addition of binary symmetry breaking clauses was investigated in [1] and [7], where they were not added iteratively but as part of an extensive procedure to reduce symmetry. Then motivated by the proceedings version of the present paper, the iterative application until no non-trivial symmetry is left, has been investigated by Anders et al. [2] and led to a code winning several categories of the SAT Competition 2026.

3 Presolving for Mixed-Integer Linear Problems

The goal in the following is to develop an alternative presolving method that also eliminates non-trivial symmetry. Further, the method should have the additional feature that it potentially preserves structure for problems that do not allow SST-cuts in their problem formulation.

The central step in the presolving routine for (P) is adding constraints that enforce the implications of the SST-cuts for optimal solutions. Throughout this section, we argue that certain modifications of problem (P) are *valid* for (P), i.e., they do not cut off all optimal solutions. We further say that some constraint holds for (P) if it is satisfied by all feasible solutions. For $N \subseteq [n]$ and $x \in \mathbb{R}^n$, we use the notation

$$\langle a_k, x \rangle_N := \sum_{p \in N} a_{k,p} x_p.$$

Definition 3.1. For $i, j \in [n]$ and $i \neq j$ the **addition operation** is:

- Let a_k be a row of A such that $a_{k,i} > 0$. If it does not already exist in (P), we add the constraint

$$\langle a_k, x \rangle_{[n] \setminus \{i,j\}} + (a_{k,i} + a_{k,j})x_j \leq b_k. \quad (\text{AO}^+(k, i, j))$$

- Let a_k be a row of A such that $a_{k,j} < 0$. If it does not already exist in (P), we add the constraint

$$\langle a_k, x \rangle_{[n] \setminus \{i,j\}} + (a_{k,i} + a_{k,j})x_i \leq b_k. \quad (\text{AO}^-(k, i, j))$$

Note that $a_{k,j} = 0$ is possible for $(\text{AO}^+(k, i, j))$ and $a_{k,i} = 0$ for $(\text{AO}^-(k, i, j))$.

Lemma 3.2. Let $(i, j) \in S$. Then $(\text{AO}^+(k, i, j))$ and $(\text{AO}^-(k, i, j))$ are valid for (P) if they are added by an addition operation.

Proof. We show that for $(i, j) \in S$, the constraints added by the addition operation are satisfied by all optimal solutions x^* of (P) that satisfy the SST-cut $x_i^* \geq x_j^*$. Then the validity of the addition operation follows from the validity of SST-cuts. Let x^* be an optimal solution of (P) such that $x_i^* \geq x_j^*$ holds. Then for any $k \in [m]$:

$$\begin{aligned} (a_{k,i} + a_{k,j})x_j^* &\leq a_{k,i}x_i^* + a_{k,j}x_j^* && \text{if } a_{k,i} > 0, \\ (a_{k,i} + a_{k,j})x_i^* &\leq a_{k,i}x_i^* + a_{k,j}x_j^* && \text{if } a_{k,j} < 0. \end{aligned}$$

Hence, x^* being feasible for (P) and therefore satisfying $a_k x \leq b_k$ guarantees that it also satisfies the constraints $(\text{AO}^+(k, i, j))$ and $(\text{AO}^-(k, i, j))$ generated by addition operations. $\square$

We call adding $(\text{AO}^+(k, i, j))$ and $(\text{AO}^-(k, i, j))$ to (P) by addition operations for $(i, j) \in S$ applying addition operations regarding S .

The addition operation takes an inequality and uses SST-cuts $x_i \geq x_j$ to aggregate x_i and x_j . As Lemma 3.2 shows, the resulting inequality is valid for every optimal solution. This idea actually also yields that the aggregated inequalities enforce the SST-cut as we show next.

Lemma 3.3. *Let x_i and x_j be variables in (P) such that for all rows a_k of A with $a_{k,i} > 0$ the constraint $(\text{AO}^+(k, i, j))$ holds for (P). Then if $c_i > 0$, one has $x_i^* \geq x_j^*$ for every optimal solution x^* . In particular, if the variables are binary, $x_i^* = 0$ implies $x_j^* = 0$.*

Proof. Assume x^* is an optimal solution of (P) with $x_i^* < x_j^*$. Define $\tilde{x}_p = x_p^*$ for all $p \in [n] \setminus \{i\}$ and $\tilde{x}_i = x_j^*$. Then $\tilde{x}$ is feasible for (P): Since x^* is feasible and $\tilde{x}_i > x_i^*$, we only have to check feasibility for constraints $a_k x \leq b_k$ with $a_{k,i} > 0$. In this case,

$$a_k \tilde{x} = \langle a_k, x^* \rangle_{[n] \setminus \{i, j\}} + (a_{k,i} + a_{k,j})x_j^* \leq b_k,$$

where we used that $(\text{AO}^+(k, i, j))$ holds for (P). Hence, $\tilde{x}$ is feasible for (P). Moreover, $c^\top \tilde{x} > c^\top x^*$, since $c_i > 0$. But this contradicts optimality of x^* . $\square$

Lemma 3.4. *Let x_i and x_j be variables in (P) such that for all rows a_k of A with $a_{k,j} < 0$ the constraint $(\text{AO}^-(k, i, j))$ holds for (P). Then if $c_j < 0$, one has $x_i^* \geq x_j^*$ for every optimal solution x^* . In particular, if the variables are binary, $x_i^* = 0$ implies $x_j^* = 0$.*

Proof. This can be shown analogously to the proof of Lemma 3.3, this time defining $\tilde{x}$ by $\tilde{x}_p = x_p^*$ for all $p \in [n] \setminus \{j\}$ and $\tilde{x}_j = x_i^*$. $\square$

We next merge variables for which no further constraints can be added.

Definition 3.5. Let x_i and x_j be two variables in (P). Further, let $c_i, c_j \geq 0$ and for all $k \in [m]$ the constraints $(\text{AO}^+(k, i, j))$ with $a_{k,i} > 0$ and $(\text{AO}^+(k, j, i))$ with $a_{k,j} > 0$ hold in (P) or let $c_i, c_j \leq 0$ and for all $k \in [m]$ the constraints $(\text{AO}^-(k, i, j))$ with $a_{k,j} < 0$ and $(\text{AO}^-(k, j, i))$ with $a_{k,i} < 0$ hold in (P). Then the **merge operation** merges variables x_i and x_j .

Lemma 3.6. *The merge operation is a valid presolving step for (P).*

Proof. Let $c_i, c_j \geq 0$ and, for all $k \in [m]$, let the constraints $(\text{AO}^+(k, i, j))$ with $a_{k,i} > 0$ and $(\text{AO}^+(k, j, i))$ with $a_{k,j} > 0$ hold in (P). Then if $c_i, c_j > 0$, by Lemma 3.3 all optimal solutions x^* satisfy $x_i^* \geq x_j^*$ as well as $x_i^* \leq x_j^*$, implying that $x_i^* = x_j^*$. We claim that if $c_i = 0$ or $c_j = 0$, there still exists at least one optimal solution x^* for which this equality holds. Indeed, we can construct a feasible solution $\tilde{x}$ like we did in the proof of Lemma 3.3 with $\tilde{x}_i = \tilde{x}_j = \max\{x_i^*, x_j^*\}$ and $\tilde{x} \geq x^*$. Thus, $c^\top \tilde{x} \geq c^\top x^*$, showing optimality of $\tilde{x}$. Therefore, we have $x_i = x_j$ for at least one optimal solution of (P).

If $c_i, c_j \leq 0$, we can argue analogously and generally get $x_i^* = x_j^*$ for at least one optimal solution x^* of (P). Therefore, we can replace every x_j in the problem formulation by x_i , which agrees with the definition of the merge operation. $\square$

Remark 3.7. Note that all constraints changed by the merge operation are of types we could possibly add by an addition operation. For instance, for a constraint $a_k x \leq b_k$ with $x_i = x_j$ we get

$$a_k x = \langle a_k, x \rangle_{[n] \setminus \{i, j\}} + a_{k,i} x_i + a_{k,j} x_j = \langle a_k, x \rangle_{[n] \setminus \{i, j\}} + (a_{k,i} + a_{k,j}) x_i \leq b_k,$$

which agrees with $(\text{AO}^+(k, j, i))$ if $a_{k,j} > 0$, $(\text{AO}^-(k, i, j))$ if $a_{k,j} < 0$ and the initial constraint if $a_{k,j} = 0$.

Proposition 3.8. *If no addition operation regarding S or merge operation is applicable for (P), the symmetry group acting on the problem variables is trivial.*

Proof. Assume the symmetry group is non-trivial. Then there exists $(i, j) \in S$ with corresponding automorphism $\gamma \in \Gamma$. Since the addition operation does not add any constraints to (P), we know that for all $k \in [m]$ with $a_{k,i} > 0$ or $a_{k,j} < 0$, the constraints $(\text{AO}^+(k, i, j))$ and $(\text{AO}^-(k, i, j))$ exist in (P), respectively.

We first claim that for a row a_k with $a_{k,j} > 0$, the constraint $(\text{AO}^+(k, j, i))$, i.e., $\langle a_k, x \rangle_{[n] \setminus \{i, j\}} + (a_{k,i} + a_{k,j}) x_i \leq b_k$, holds for all feasible x of (P). Since constraint $a_k x = \langle a_k, x \rangle_{[n] \setminus \{i, j\}} + a_{k,i} x_i + a_{k,j} x_j \leq b_k$ is in (P), there exists some row $\ell \in [m]$ such that

$$a_\ell x = \langle a_k, \gamma(x) \rangle_{[n] \setminus \{\gamma^{-1}(i), i\}} + a_{k,i} x_{\gamma^{-1}(i)} + a_{k,j} x_i \leq b_\ell = b_k$$

due to formulation symmetry. Now we will use that $\gamma \in \Gamma$, $\gamma(i) \neq i$ implies that also $\gamma^{-1} \in \Gamma$, $\gamma^{-1}(i) \neq i$ and therefore, $(i, \gamma^{-1}(i)) \in S$. Thus, since $a_{k,j} > 0$, we have by assumption that the constraint

$$\langle a_k, \gamma(x) \rangle_{[n] \setminus \{\gamma^{-1}(i), i\}} + (a_{k,i} + a_{k,j}) x_{\gamma^{-1}(i)} \leq b_k$$

is in the formulation of (P), because it corresponds to $(\text{AO}^+(\ell, i, \gamma^{-1}(i)))$. Since γ^{-1} is a symmetry, we obtain that the constraint $\langle a_k, x \rangle_{[n] \setminus \{i, j\}} + (a_{k,i} + a_{k,j}) x_i \leq b_k$ is in (P) which is the desired constraint.

We can analogously show that the constraints of the type $(\text{AO}^-(k, j, i))$ hold in (P). Thus, the claim is true and the merge operation is applicable. But this contradicts the initial assumption. Hence, there cannot exist a non-trivial symmetry. $\square$

Note that for the above proof we only need to merge symmetric variables. Nevertheless, the definition is more general in order to increase the chances to fix variables and the efficiency of presolving.

We propose the following presolving routine:

Definition 3.9 ((Mixed-integer) SST-presolving).

1. *Merging:* Apply merge operations exhaustively.
2. Compute symmetry group of Problem (P) and corresponding SST-pairs.
3. *Addition:* Apply addition operations regarding S exhaustively.

Proposition 3.10. *After finitely many applications of SST-presolving, the symmetry group of problem (P) becomes trivial.*

Proof. Assume this is not true. By Proposition 3.8, this means that there can be infinitely many applications of SST-presolving in which the problem is modified. Clearly, there can only be finitely many rounds of SST-presolving in which merge presolving is active since it reduces the number of unfixed variables of the problem. So we need to show that the number of constraints that can possibly be added by addition operations is also bounded. But this is clear, since all added constraints use a constraint of the current problem formulation and update coefficients by sums of the initial coefficients that only appear once in these sums. Hence, there are only finitely many possibilities for a constraint that can be added by an addition operation. $\square$

3.1 Presolving for Binary Linear Problems

Proposition 3.10 shows that iteratively applying SST-presolving on a MIP terminates with a MIP with trivial symmetry. At first glance, being able to apply this symmetry handling technique for general MIPs seems as strong as possible. However, if the goal is to handle symmetry while preserving problem structure, this goal becomes harder to reach, the more restrictive the problem class is. Therefore, in the following we are going to extend SST-presolving such that it is structure preserving for several binary problems which we will consider in detail in Section 4. We consider binary linear problems of the form

$$\max \{c^\top x : Ax \leq b, x_i \in \{0, 1\}^n\}, \quad (\text{BP})$$

where $A = (a_{i,j})_{i \in [m], j \in [n]} \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$. The k th row of A again is denoted by a_k .

For a binary problem, it is possible that the addition operation adds constraints that enforce the fixing of a certain variable. Since variable fixing is usually more efficient in presolving than the addition of constraints, we want to fix the variable right away, instead of adding constraints that imply the fixing. This approach aims not only to handle symmetry, but also to make use of it as soon as possible. Further, it turns out to be helpful when it comes to preserving structure.

For $i \in [n]$ we define $S^i := \{p \in [m] : (i, p) \in S\}$, the set of followers of i in S and $\bar{S}_i := \{p \in [m] : (p, i) \in S\}$, the set of leaders having follower i . The complements of these sets $[m] \setminus S^i$ and $[m] \setminus \bar{S}_i$ are denoted by $\bar{S}^i$ and $\bar{\bar{S}}_i$, respectively. Further, we define $\mathcal{A}_k^i := \{p \in [m] \setminus \{i\} : a_{k,p} < 0\}$ and for $J \subseteq [m]$ we define $a_k(J) := \sum_{i \in J} a_{k,i}$.

Definition 3.11. Let a_k be a row in A and $i \in [n]$ such that

$$a_k(S_i) + a_{k,i} + a_k(\bar{S}_i \cap \mathcal{A}_k^i) > b_k. \quad (\text{DO}(k, i))$$

Then we call fixing x_i to 0 **deletion operation**.

Let a_k be a row in A and $i \in [n]$ such that

$$a_k(\bar{S}^i \cap \mathcal{A}_k^i) > b_k. \quad (\text{FO}(k, i))$$

Then we call fixing x_i to 1 **fixation operation**.

Lemma 3.12. The deletion operation is a valid presolving step for (BP).

Proof. Assume (DO(k, i)) holds for some $k \in [m]$. Then fixing $x_i = 1$ implies $x_p = 1$ for all $p \in S_i$. Therefore, we have for feasible $\tilde{x}$ with $\tilde{x}_i = 1$ respecting the SST-cuts in S that

$$a_k \tilde{x} = a_k(S_i) + a_{k,i} + \sum_{p \in \bar{S}_i \setminus \{i\}} a_{k,p} \tilde{x}_p \geq a_k(S_i) + a_{k,i} + a_k(\bar{S}_i \cap \mathcal{A}_k^i) > b_k,$$

contradicting that $a_k \tilde{x} \leq b_k$. Hence $\tilde{x}_i$ has to be 0. $\square$

Lemma 3.13. The fixation operation is a valid presolving step for (BP).

Proof. Assume (FO(k, i)) holds for some $k \in [m]$. Then fixing $x_i = 0$ implies $x_p = 0$ for all $p \in \bar{S}^i$. Therefore, we have for feasible $\tilde{x}$ with $\tilde{x}_i = 0$ respecting the SST-cuts in S that

$$a_k \tilde{x} = \sum_{p \in \bar{S}^i \setminus \{i\}} a_{k,p} \tilde{x}_p \geq a_k(\bar{S}^i \cap \mathcal{A}_k^i) > b_k,$$

contradicting that $a_k \tilde{x} \leq b_k$. Hence $\tilde{x}_i$ has to be 1. $\square$

It is possible that some problem structure does not allow to sum up several coefficients of a constraint. In this case, SST-presolving might not modify a problem in a structure preserving way. Fixing variables, however, usually never changes the structure of a particular problem. Therefore, adding the deletion and fixation operations to SST-presolving has the potential to avoid the addition of structure violating constraints through the fixation of variables. For binary problems, we therefore propose the following extended SST-presolving:

Definition 3.14 ((Binary) SST-presolving). 1. *Merging*: Apply merge operations exhaustively.

2. Compute symmetry group of Problem (P) and corresponding SST-pairs.

3. *Deletion & Fixation*: Apply deletion and fixation operations exhaustively.

4. *Addition*: Apply addition operations regarding S exhaustively.

Note that Proposition 3.8 remains true, if additionally no deletion and fixation operations are applicable. Therefore, Proposition 3.10 is also correct for binary SST-presolving. This is because the deletion and fixation operation both reduce the number of unfixed variables of the problem when applied and thus, just as for the merge operation, it can be argued that there can only be finitely many rounds in which these operations can be applied. Hence, iteratively applying binary SST-presolving to a binary problem results in a problem with trivial symmetry.

In the following, we will refer to binary SST-presolving just by SST-presolving and specify if we mean mixed-integer SST-presolving instead.

4 Structure Preserving Presolving

In this section, we discuss examples where SST-presolving leaves the particular structure invariant.

4.1 Packing and Covering Integer Programs

A *packing integer program* arises from Problem (BP) if A , b and c are nonnegative. A *covering integer program* arises if A , b and c are nonpositive, which can be reformulated as a minimization problem over $-Ax \geq -b$. In both cases, SST-presolving preserves the structure, i.e., the coefficients stay nonnegative and nonpositive, respectively. Then, for instance, approximation algorithms can be applied, see, e.g., [19].

4.2 Presolving for the Stable Set Problem

Let $G = (V, E)$ be an undirected simple graph and $c \in \mathbb{R}_+^V$ be nonnegative node weights. The *maximum (weight) stable set problem* is

$$\max \{c^\top x : x_u + x_v \leq 1 \ \forall \{u, v\} \in E, x \in \{0, 1\}^V\}. \quad (\text{MSSP})$$

A special feature of the stable set problem is that the formulation symmetry of the problem coincides with the overall symmetry of the problem. Some of the following steps have been investigated in [12], but without finiteness proof. For a node $u \in V$ we define the neighborhood of u by $N(u) := \{v \in V : \{u, v\} \in E\}$.

Lemma 4.1. *If SST-presolving is applied on (MSSP), the presolving steps can be translated to the stable set setting in the following way:*

- *Merge*: Let $u, v \in V$ such that $N(u) = N(v)$, in particular $\{u, v\} \notin E$. Fix $x_v = 0$, delete all edges incident to v from E and update c_u to $c_u + c_v$.
- *Deletion*: Let $\{u, v\} \in E$ and $(u, v) \in S$. Fix variable $x_v = 0$.
- *Fixation*: Not applicable.
- *Addition*: Let $\{u, v\} \in E$ and $(v, w) \in S$. Then the edge $\{u, w\}$ is added to E if it is not already part of G .

Proof. Since A is nonnegative for the stable set problem, the condition for the deletion operation reduces to $a_k(S_v) + a_{k,v} > 1$. If the variables for both members of an SST-pair are contained in the same constraint, which means there is an edge between leader and follower in E , this condition is fulfilled and the follower variable is fixed to 0. The condition of the fixation operation reduces to $0 > 1$ and is therefore never fulfilled.

Let $(u, v) \in S$. Again by the nonnegativity of A , the addition operation only adds constraints of the form $(\text{AO}^+(k, u, v))$ with $a_{k,u} > 0$. Every row k with $a_{k,u} > 0$ corresponds to an edge $\{u, w\} \in E$. If $w \neq v$ and therefore $a_{k,v} = 0$, the constraint $(\text{AO}^+(k, u, v))$ just replaces x_u by x_v , i.e., it corresponds to the edge $\{v, w\}$.

If $w = v$, the variables for both members of the SST-pair (u, v) are contained in the constraint corresponding to row k . Thus, x_v would have been removed by the deletion operation.

Since $c \geq 0$, the merge condition is that for $u, v \in V$, the constraints $(AO^+(k, u, v))$ with $a_{k,u} > 0$ and $(AO^+(k, v, u))$ with $a_{k,v} > 0$ hold in (MSSP) for all $k \in [m]$. First assume that $\{u, v\} \in E$. Then the constraint $(AO^+(k, u, v))$ is $2x_v \leq 1$. This constraint can neither be part of the initial formulation of (MSSP), since G is assumed to be a simple graph, nor implied by edge constraints nor added by an addition operation. Thus, the merge condition is never met if the involved nodes are adjacent. For non-adjacent nodes u and v , the constraints $(AO^+(k, u, v))$ with $a_{k,u} > 0$ correspond to edges between v and neighbors of u (see the translation of the addition operation). More precisely, the merge condition that all constraints $(AO^+(k, u, v))$ with $a_{k,u} > 0$ for all $k \in [m]$ hold in (MSSP) means that all neighbors of u are also neighbors of v . Here we use again that all edge inequalities that hold for the stable set problem need to be part of the problem formulation. Since this condition also has to hold if u and v switch places, we obtain the condition $N(u) = N(v)$. Thus, $a_{k,u}$ is updated by a merge operation if $a_{k,v} > 0$ which means that $a_{k,v} = 1$. This implies that $a_{k,u} = 0$, because u and v are non-adjacent. So the update changes an edge $\{v, w\}$ to an edge $\{u, w\}$. But this edge already has to exist because $N(u) = N(v)$. Thus, it is valid to just delete the edge $\{v, w\}$. $\square$

No constraint with a coefficient larger than 1 can be added by an addition operation, because of the prior deletion operation. Therefore, the addition operation only adds constraints corresponding to edges. Hence, as we will show in Corollary 5.2 below, we obtain the upper bound of n^2 on the number of SST-presolving rounds for (MSSP), here derived from the maximal number of non-parallel edges that can be added to and nodes that can be deleted from a simple graph. Section 7.1 will include an investigation of the gap between this naive upper bound and practical results.

Remark 4.2. As a side note, the *maximum k -colorable subgraph problem* is a special stable set problem arising from the product of a graph with a complete graph with k nodes [15]. Here, SST-presolving alters the problem structure to some extent – the added constraints might run across different color classes.

4.3 Presolving for the Set Cover Problem

Let U be a finite set of items and let P be a finite family of sets $\sigma \subseteq U$. The *weighted set cover problem* with set weights $c \in \mathbb{R}_+^P$ is defined by

$$\min \{c^\top x : \sum_{u \in \sigma \in P} x_\sigma \geq 1 \quad \forall u \in U, x \in \{0, 1\}^P\}. \quad (\text{WSCP})$$

We assume that every $u \in U$ is contained in some $\sigma \in P$, since (WSCP) would be trivially infeasible otherwise.

The presolving steps of Definition 3.14 can be translated to the set cover setting in the following way:

- *Addition:* Let $(\sigma, \tau) \in S$ and let $u \in \tau$. Add a new item $\tilde{u}$ to U and add it to σ as well as all sets in $P \setminus \{\tau\}$ that contain u .
- *Merge:* Let $\sigma, \tau \in P$ such that for all $u \in \sigma$ with $u \notin \tau$ there exists $u' \in \tau$ such that $\{\rho \in P : u \in \rho\} \setminus \{\sigma\} = \{\rho \in P : u' \in \rho\} \setminus \{\tau\}$ and the other way around. Further, for all $u \in \sigma \cap \tau$ let there exist $u' \in \sigma$ with $u' \notin \tau$ as well as $u'' \in \tau$ with $u'' \notin \sigma$ such that

$$\{\rho \in P : u \in \rho\} \setminus (\{\sigma\} \cup \{\tau\}) = \{\rho \in P : u' \in \rho\} \setminus \{\sigma\} = \{\rho \in P : u'' \in \rho\} \setminus \{\tau\}.$$

Then fix $x_\tau = 0$, delete all items $u \in \tau$ with $u \notin \sigma$ from U and update c_σ to $c_\sigma + c_\tau$.

- *Deletion:* Not applicable.
- *Fixation:* Let $u \in U$ and $\sigma \in P$ such that for all $\tau \in P$ with $u \in \tau$ we either have $\tau = \sigma$ or $(\sigma, \tau) \in S$. Then the variable x_σ is fixed to 1.

Analogously to Lemma 4.1 for the stable set problem we are going to show that the above translation is correct.

Lemma 4.3. *If SST-presolving is applied on (WSCP), we obtain the operations described above.*

Proof. First, we bring (WSCP) into the general form (P):

$$\max \left\{ -c^\top x : \sum_{u \in \sigma \in P} -x_\sigma \leq -1 \ \forall u \in U, x \in \{0, 1\}^P \right\}.$$

In particular, the objective vector is nonpositive and all entries of A are either 0 or -1 .

Assume the deletion operation was applicable. Then there exists $u \in U$ and $\sigma \in P$ such that

$$a_u(S_\sigma) + a_{u,\sigma} + a_u(\bar{S}_\sigma \cap \mathcal{A}_u^\sigma) > -1.$$

Since the entries in A are either 0 or -1 , this implies that

$$a_u(S_\sigma) = a_{u,\sigma} = a_u(\bar{S}_\sigma \cap \mathcal{A}_u^\sigma) = 0.$$

Moreover, we get

$$\begin{aligned} a_u(\bar{S}_\sigma \cap \mathcal{A}_u^\sigma) = 0 &\iff \sum_{i \in (\bar{S}_\sigma \cap \mathcal{A}_u^\sigma)} a_{u,i} = 0 \iff \sum_{i \in \bar{S}_\sigma \setminus \{\sigma\}} a_{u,i} = 0 \\ &\iff a_{u,i} = 0 \quad \forall i \in \bar{S}_\sigma \setminus \{\sigma\} \end{aligned}$$

as well as

$$a_u(S_\sigma) = 0 \iff \sum_{i \in S_\sigma} a_{u,i} = 0 \iff a_{u,i} = 0 \quad \forall i \in S_\sigma.$$

But this all together contradicts our assumption that A does not contain an all-zero row. Thus, the deletion operation is not applicable for the set cover problem.

The fixation operation is applicable if there exists $u \in U$ and $\sigma \in P$ such that

$$a_u(\bar{S}^\sigma \cap \mathcal{A}_u^\sigma) > -1.$$

As above, this implies that $a_{u,\tau} = 0$ for all $\tau \in \bar{S}^\sigma \setminus \{\sigma\}$. In other words, the fixation operation fixes x_σ to 1 if there exists $u \in U$ such that $a_{u,\tau} = -1$ (i.e., $u \in \tau$) implies that $\tau \in S^\sigma \cup \{\sigma\}$ which means that $(\sigma, \tau) \in S$ or $\tau = \sigma$.

Now consider the addition operation for $(\sigma, \tau) \in S$. By the nonpositivity of A , the addition operation only adds constraints of the form $(AO^-(u, \sigma, \tau))$ with $u \in \tau$. Such a new constraint corresponds to a new item $\tilde{u}$ with constraint

$$\langle a_u, x \rangle_{P \setminus \{\sigma, \tau\}} + (a_{\tilde{u}, \sigma} + a_{\tilde{u}, \tau})x_\sigma \leq -1.$$

As in the case of the stable set problem, the addition operation can possibly generate constraints with coefficients smaller than -1 . But since all variables are binary in (WSCP), the constraints $\langle a_u, x \rangle_{P \setminus \{\sigma, \tau\}} - 2x_\sigma \leq -1$ and $\langle a_u, x \rangle_{P \setminus \{\sigma, \tau\}} - x_\sigma \leq -1$ are equivalent. Therefore it is equally valid to add the second constraint, while preserving the problem structure. This constraint corresponds to a new item $\tilde{u}$ as described in the addition operation above.

Since the objective vector of (WSCP) is nonpositive, the condition for merging σ and $\tau \in P$ is that the constraints $(AO^-(u, \sigma, \tau))$ for all $u \in \tau$ and $(AO^-(u, \tau, \sigma))$ for all $u \in \sigma$ hold for all feasible solutions of (WSCP). It is straightforward to see that the first condition of the above described merge operation checks that these conditions are met for all $u \in \sigma \setminus \tau$ and all $u \in \tau \setminus \sigma$. The second condition does the same for all $u \in \sigma \cap \tau$. Therefore, the conditions ensure that the merge operation is applicable and one can set $x_\tau = x_\sigma$ by Lemma 3.6. For constraints corresponding to $u \in \tau \cap \sigma$, the same argument is applicable as for the addition operation regarding setting coefficients smaller than -1 to -1 . This is why it is valid to fix x_τ to 0 instead of replacing x_τ by x_σ . If we have $u' \in \tau \setminus \sigma$ and the merge conditions are satisfied, there exists some $u \in U$ such that $u \in \sigma \setminus \tau$ and $\{\rho \in P : u \in \rho\} \setminus \{\sigma\} = \{\rho \in P : u' \in \rho\} \setminus \{\tau\}$. Replacing x_τ by x_σ in the constraint corresponding to u' leads to $\langle a_{u'}, x \rangle_{P \setminus \{\tau\}} - x_\sigma \leq -1$. Now since $\{\rho \in P : u \in \rho\} \setminus \{\sigma\} = \{\rho \in P : u' \in \rho\} \setminus \{\tau\}$, we have $\langle a_{u'}, x \rangle_{P \setminus \{\tau\}} = \langle a_u, x \rangle_{P \setminus \{\sigma\}}$ and therefore, the constraint we obtain by replacing x_τ by x_σ is $\langle a_u, x \rangle_P \leq -1$, the constraint corresponding to u . Consequently, it is valid to instead of adding the redundant modified constraint to remove it from the problem as it is done in the translation of the merge operation above. $\square$

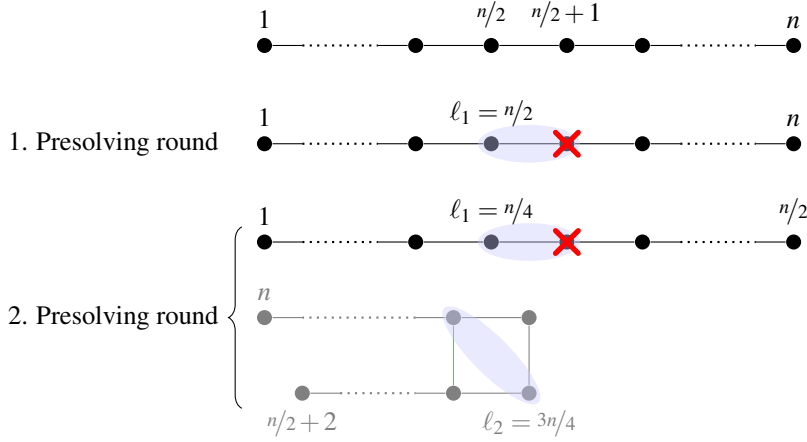


Figure 1: First two SST-presolving rounds applied on a stable set instance that decomposes into two subgraphs with one behaving similarly as the initial graph. Orbits of leaders are marked in blue. Nodes deleted by deletion operations are crossed out in red. Edges added by addition operations are colored in green.

5 Bounds on the Number of Presolving Rounds

Since Proposition 3.10 shows that only finitely many applications of SST-presolving are necessary to fully handle symmetry, the natural question to ask is in which range this number lies. Unfortunately the argument in the proof of Proposition 3.10 just leads to an exponential upper bound. Anyhow, in the computational results for the stable set problem presented in Section 4.2, only few rounds of SST-presolving are needed to handle the symmetry completely, suggesting that the exponential upper bound is far from being tight. We will therefore investigate stronger bounds on the number of iterations of SST-presolving in this section.

5.1 Lower Bound

Consider the instance of the stable set problem (MSSP) illustrated in Figure 1; see Lemma 4.1 for the corresponding presolving steps. Let graph $G = (V, E)$ be a chain with an even number n of nodes. We assume the nodes to be labeled such that the neighbors of node $i \in \{2, \dots, n-1\}$ are $i-1$ and $i+1$. If the length of the chain is at least 4, there clearly is no merge operation applicable, since no two nodes share exactly the same neighbors. The symmetry group of G consists of one non-trivial element which is the reflection automorphism permuting i and $n-i+1$ for all $i \in [n/2]$. Assume the first leader in L is $\ell_1 = n/2$. Then we have $\text{orb}(\ell_1, \Gamma) = \{n/2, n/2+1\}$. Since $\{n/2, n/2+1\} \in E$, the deletion operation deletes node $n/2+1$. Therefore, because all followers of ℓ_1 were deleted, no addition operation is applicable. Further, since the only automorphism in Γ stabilizing ℓ_1 is the identity, we have $L = (\ell_1)$. So after one round of SST-presolving, the graph decomposes into one chain of length $n/2$ and one chain of length $n/2-1$.

Assume that $n = 2^k$, $k \in \mathbb{N}$. Then after one round of SST-presolving, an even chain of length 2^{k-1} and an odd chain of length $2^{k-1}-1$ remains. It is easy to check that applying SST-presolving to an odd chain of length at least 7 with leader ℓ_1 being a neighbor of the central node in the chain will always lead to a graph with trivial symmetry containing a cycle after one SST-presolving round. The cycle is created by an addition operation as displayed in Figure 1. In particular, this graph is not isomorphic to some chain graph and is thus not interfering the even chain. Then the above argument can be applied iteratively on the even subchains of G that appear during the presolving process. Since then G decomposes k times into an even and an odd chain, we obtain a lower bound for the SST-presolving rounds of $\log(n) = k$.

5.2 Upper Bound

To derive an upper bound on the number of SST-presolving applications, we first prove the following.

Proposition 5.1. *If $c_i \neq 0$ for all $i \in [n]$, then whenever there is an SST-pair computed for the second time since the beginning of the iterative mixed-integer or binary SST-presolving routine, it is valid to merge the variables in the SST-pair.*

Proof. Consider a presolving round in which the SST-pair (i, j) is computed. Assume the SST-pair (i, j) gets computed again in a later presolving round. This means there exists a symmetry γ such that $\gamma(i) = j$. As already argued in the proof of Lemma 2.1, there exists some $p \in \mathbb{N}$ such that $\gamma^p(i) = i$. Note that x_i and x_j being symmetric implies that $c_i = c_j$. Then after the application of addition presolving, we know by Lemma 3.3 and 3.4 that $x_i^* \geq x_j^*$ for every optimal solution x^* . Since symmetries map an optimal solution x^* onto optimal solutions, the inequalities

$$x_{\gamma^k(i)}^* = (\gamma^k)^{-1}(x^*)_i \geq (\gamma^k)^{-1}(x^*)_j = x_{\gamma^k(j)}^* = x_{\gamma^{k+1}(i)}^*,$$

also have to hold for every $k \in \mathbb{N}$. So we obtain the chain of inequalities

$$x_i^* \geq x_j^* = x_{\gamma(i)}^* \geq x_{\gamma^2(i)}^* \geq \dots \geq x_{\gamma^p(i)}^* = x_i^*$$

proving that $x_i^* = x_j^*$ for all optimal solutions x^* . □

Corollary 5.2. *If $c_i \neq 0$ for all $i \in [n]$ and the merge operation is extended by the merging proposed in Proposition 5.1, mixed-integer as well as binary SST-presolving is performed at most n^2 times until it terminates with a problem with trivial symmetry.*

Proof. In each presolving round except for the last, at least one SST-pair gets computed. So there can be at most $n \cdot (n - 1)$ presolving rounds in which the sets of SST-pairs are pairwise disjoint. In all other rounds, SST-pairs that have been generated before have to reappear. By Proposition 5.1 this leads to a reduction of unfixed variables. Since there are n variables at the beginning, we obtain $n \cdot (n - 1) + n = n^2$ as an upper bound on the SST-presolving iterations. □

6 Relation between SST-cuts, SST-presolving and Orbital Branching

SST-presolving being derived from SST-cuts suggests a strong connection between the addition of SST-cuts and the application of SST-presolving as symmetry handling techniques. It further turns out that these two techniques are related to orbital branching. In the following, we are comparing the corresponding optimal solutions as well as branch-and-bound trees derived from these symmetry handling techniques. For this, we will restrict ourselves to binary problems of the form (BP), since they are the usual use case of orbital branching.

6.1 Optimal Solutions after SST-cut and SST-presolving Application

We denote by (P_{SST}) the problem we obtain if we add all SST-cuts corresponding to SST-pairs in $S(L)$ to (BP) and denote the problem we obtain after one round of SST-presolving to (BP) for $S(L)$ by (P_{presol}) . For this section, we redefine SST-presolving to starting with Step 2, the symmetry computation, followed by Step 3, Step 4 and finally Step 1. Further, we assume that merging is only applied for SST-pairs. We do this in order to make the presolving and the addition of SST-cuts comparable. If we would add the SST-cuts computed after merging in the presolving routine to (BP), the cuts might not correspond to symmetry in the initial problem.

Proposition 6.1. *If $c > 0$ or $c < 0$, the set of optimal solutions of (P_{SST}) and (P_{presol}) coincide.*

Proof. By Lemma 3.3 and 3.4, we know that all optimal solutions of (P_{presol}) are feasible for (P_{SST}) . Thus, we only have to show the converse. Let x^* be an optimal solution of (P_{SST}) . Assume it is not feasible for (P_{presol}) . We distinguish the three cases that can cause the infeasibility:

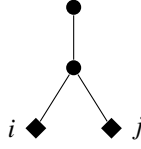


Figure 2: Stable set instance where diamond shaped nodes have objective value 0 and circular nodes have objective value 1.

Case 1: A constraint was added by an addition operation that is violated by x^* . But this is impossible, because in the proof of Lemma 3.2 we showed that all constraints added by the addition operation are satisfied by all optimal solutions of (P_{SST}) .

Case 2: Variables x_i and x_j are merged by presolving but $x_i^* \neq x_j^*$. But the conditions required to hold for the application of the merge operation imply that $x_i^* = x_j^*$ as it is shown in the proof of Lemma 3.6.

Case 3: The variable x_i is fixed to 0 by a deletion operation in the SST-presolving, but $x_i^* = 1$ or it is fixed to 1 by a fixation operation in the SST-presolving while $x_i^* = 0$. But this again is impossible, since the proofs of Lemmas 3.12 and 3.13 show that all feasible solutions of (P_{SST}) have to agree with the fixations of the deletion and fixation operations. This is even true if the required constraint for the operation is not part of (P_{SST}) . The inequality only has to hold for all feasible solutions of (P_{SST}) and all constraints added to (P_{presol}) hold in (P_{SST}) as explained in Case 1.

Hence, x^* is feasible for (P_{presol}) which finishes the proof, because the objective values of a particular solution in both problems agree by construction. $\square$

Note that Proposition 6.1 also holds for the mixed-integer case, where in particular, mixed-integer SST-presolving is applied on (P) for one round to obtain (P_{presol}) . Then the statement can be proven analogously to the binary case except that Case 3 of the case distinction can be removed.

Remark 6.2. If the objective c contains 0s, it might happen that an optimal solution of (P_{SST}) is removed from (P_{presol}) : The stable set instance in Figure 2 has $c_i = c_j = 0$. Computing symmetry leads to either $S = \{(i, j)\}$ or $S = \{(j, i)\}$. Hence, there still exists an optimal solution with $x_i \neq x_j$. But since the merge operation is applicable for x_i and x_j , this solution is not feasible in (P_{presol}) . Thus, the set of optimal solutions of (P_{SST}) does not have to be a subset of the set of optimal solutions of (P_{presol}) if the objective can be 0.

6.2 Comparison to Orbital Branching

An obvious question is whether it is more beneficial to break symmetry completely before starting the branch-and-bound process or if symmetry handling throughout the solving process is the better choice. Since there is no established hierarchy among different methods of symmetry breaking, it is not clear how to theoretically classify the effectiveness of symmetry handling techniques. Therefore, we are going to compare our approaches to orbital branching [26, 28] by comparing the branch-and-bound trees generated using the three different symmetry handling techniques.

We will show that after applying one round of SST-presolving or respectively adding all SST-cuts computed in the first round to (P) , the branch-and-bound tree resembles the branch-and-bound tree we obtain using Orbital Branching (OB). To do so, we first clarify how the branch-and-bound trees are generated.

We denote by T_{presol} (resp. T_{SST}) the variable disjunction branch-and-bound tree for the problem (P_{presol}) (resp. (P_{SST})) where the beginning of the branching order is given by the leader sequence $(\ell_1, \dots, \ell_k)$ and after that some fixed global order. We additionally demand that the representative of two merged variables appears earlier in the branching order than the corresponding non-representative variable. From now on, we assume that the variables are labeled according to this branching order. We denote by F_a^1 (resp. F_a^0) the set of indices of variables branched to 1 (resp. 0) on the path from the root to node a in the corresponding branch-and-bound tree.

We further denote by T_{OB} the branch-and-bound tree for problem (BP) where OB is used as a general branching rule with the same branching order used to generate T_{presol} or T_{SST} . Generally, we assume trees generated by basic branch-and-bound, i.e., no additional propagators, heuristics or further cutting planes are used during the generation of T_{presol} , T_{SST} or T_{OB} . As common in the literature, we assume that OB uses the stabilizer of F_a^1 with respect to the initial symmetry group. Thus, we do not recompute the local symmetry information. If the binary disjunction for OB is

$$x_\ell = 1 \vee \sum_{f \in O_\ell} x_f = 0, \quad (1)$$

we say that x_ℓ is the branching variable and the variables x_f with $f \in O_\ell \setminus \{\ell\}$ are fixed variables.

Further, we additionally demand that if a variable x_i in T_{SST} is branched to 0 and $(i, j) \in S$, the variable x_j is fixed to 0 because the SST-cut $x_i \geq x_j$ is part of the problem formulation of (P_{SST}) .

Proposition 6.3. *The trees T_{SST} and T_{OB} are isomorphic, if nodes are only pruned based on infeasibility.*

Proof. We claim that for every node a in T_{SST} there exists a node b in T_{OB} such that $F_a^0 = F_b^0$ and $F_a^1 = F_b^1$ and vice versa.

We first show that the claim proves the statement. Since the branching order is fixed, the sets F_a^0 and F_a^1 uniquely define a node a in T_{SST} as well as T_{OB} . So by the claim, we have a bijective map from the nodes of T_{SST} to the nodes of T_{OB} . To show that this map preserves the tree structure, let a' be a child node of a in T_{SST} . Then either $F_{a'}^0 = F_a^0 \cup \{i\}$ and $F_{a'}^1 = F_a^1$ or $F_{a'}^0 = F_a^0$ and $F_{a'}^1 = F_a^1 \cup \{i\}$ for some i where i is the largest element in $F_{a'}^0 \cup F_{a'}^1$. By the claim, there exists b in T_{OB} with the same branching variables as a and b' in T_{OB} with the same branching variables as a' . This means in particular that $F_{b'}^0 = F_b^0 \cup \{i\}$ and $F_{b'}^1 = F_b^1$ or $F_{b'}^0 = F_b^0$ and $F_{b'}^1 = F_b^1 \cup \{i\}$ with i being the largest element in $F_{b'}^0 \cup F_{b'}^1$. Hence, b' is a child node of b . This argument can analogously be applied for the opposite direction and therefore shows that the map preserves the tree structure. Consequently, having an adjacency preserving bijective map from T_{SST} to T_{OB} , the trees have to be isomorphic.

We now prove the claim by induction on the tree depth d of the nodes a and b in T_{SST} and T_{OB} respectively. The induction start is correct, since the root node of T_{SST} and the root node of T_{OB} both have ℓ_1 as branching variable by definition. Moreover, since so far no branching has been performed, no variable can possibly be fixed when the first variable is branched on. The induction hypothesis is that the statement is true for all nodes at depth d of T_{SST} and for all nodes at depth d of T_{OB} .

First, let a' be a node in T_{SST} at depth of $d+1$. Then a' is a child of some node a at depth d in T_{SST} . By the induction hypothesis, there exists a node b in T_{OB} such that $F_a^0 = F_b^0$ and $F_a^1 = F_b^1$. Let x_i be the branching variable at a .

We will show that x_i is also the branching variable at b and distinguish two cases.

Case 1: The branching variable at b is x_h with $h < i$. Since $h \notin F_b^0 \cup F_b^1 = F_a^0 \cup F_a^1$ and h is not the branching variable at a , it had to be fixed on the path from root to a . Here we use that in both trees the same branching order is applied. Since a variable in T_{SST} can only be fixed due to an SST-pair, there exists $(g, h) \in S$ with $g < h$ and $g \in F_a^0$. But this is equivalent to the existence of $\gamma \in \text{stab}(\Gamma, \{1, \dots, g-1\})$ with $\gamma(g) = h$ and $g \in F_b^0$, where we use $F_a^0 = F_b^0$ from the induction hypothesis. Consequently, at an ancestor of b in T_{OB} the branching disjunction (1) for branching variable g fixes x_h , contradicting that it is the branching variable at b .

Case 2: The branching variable at b is x_j with $i < j$. Again, because of the same branching order in T_{SST} and T_{OB} , this can only be the case if x_i is fixed at b . We use the same equivalence as in Case 1 to show that this implies that x_i is fixed at a , contradicting that it is the branching variable at a . Consequently, the child node of b where x_i is branched to the same value as in a' , gives us the desired b' .

For the converse direction, let b' be a node in T_{OB} at depth of $d+1$ and b its parent node. Again, by the induction hypothesis, there exists a node a in T_{OB} such that $F_a^0 = F_b^0$ and $F_a^1 = F_b^1$. Let x_i be the branching variable at b . If x_i is not also the branching variable at a , we either have that the branching variable at a is larger than i which agrees with Case 1 or that the branching variable at a is smaller than i which agrees with Case 2. Since both cases lead to contradictions, we obtain that the child node of a where x_i is branched to the same value as in b' , gives us the desired a' . This finishes the proof of the claim. $\square$

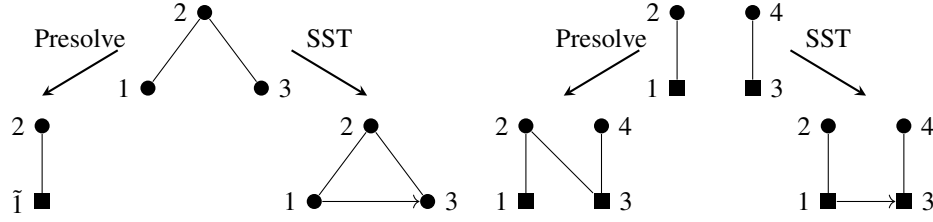


Figure 3: Stable set instances where square shaped nodes have double the objective value of circular shaped nodes and SST-cuts are displayed as directed edges.

Remark 6.4. The LP computed at each node of the according branch-and-bound tree is potentially stronger for (P_{SST}) than for (BP) , because the set of constraints of (BP) is a subset of the constraints of (P_{SST}) . Hence, if the trees are not only pruned by infeasibility but also by the LP value, it can happen that a node in T_{SST} is pruned while the corresponding node in T_{OB} is not. Consequently, in this case T_{SST} is an induced subgraph of T_{OB} .

Remark 6.5. Note that for T_{SST} we only demand to fix followers to 0 whenever one of their leaders is branched to 0. One could demand to fix a leader to 1 whenever one of its followers is branched to 1. This, however, is never applicable because of the branching order – when a follower is branched on, the leader is already fixed.

In the following we consider the branch-and-bound tree T_{presol} . While we know from Proposition 6.1 that the trees T_{presol} and T_{SST} produce the same optimal solutions, their sets of all feasible solutions can be different. Figure 3 displays example instances of the stable set problem where the solution $(1, 0, 0)^\top$ is feasible for (P_{SST}) but not for (P_{presol}) for the instance on the left and $(0, 0, 1, 0)^\top$ is a feasible solution for (P_{presol}) but not for (P_{SST}) for the instance on the right.

When there is an SST-cut between two nodes for which the merge operation is applicable, SST-presolving excludes all solutions where the merged nodes have different values while the SST-cut only enforces one inequality. This difference can be removed for the iterative approaches by using Proposition 5.1. Further, a difference in feasible solutions arises from SST-cuts excluding stable sets containing a follower but not its leaders. This is not directly enforced by the addition operation but this difference can be removed by using the following fixing rule.

Definition 6.6. Let a be a node in T_{presol} . Let x_i, x_j be variables in (P_{presol}) with $c_i, c_j \geq 0$ or $c_i, c_j \leq 0$ such that for the pair (i, j) no addition operation is applicable regarding the constraints at node a in T_{presol} . *Dominance fixing* fixes variable x_j to 0 if x_i is branched or fixed to 0 at a .

Lemma 6.7. *Dominance fixing is valid for (P_{presol}) .*

Proof. If $c_i > 0$, the inequality $x_i \geq x_j$ has to hold for all optimal solutions at a by Lemma 3.3. This is also the case if $c_j < 0$ by Lemma 3.4. If $c_i = 0$ we can still apply the proof of Lemma 3.3, but since increasing x_i does not increase the objective value, it then only shows that from every optimal solution we can derive an optimal solution that satisfies the inequality $x_i \geq x_j$. The same can be shown for $c_j = 0$ and Lemma 3.4. So in any case, $x_i \geq x_j$ holds for at least one optimal solution as long as the problem at a is feasible. Consequently, if x_i is branched or fixed to 0, $x_j = 0$ has to hold for at least one optimal solution at a , if feasible. Thus, it is valid to fix x_j to 0. $\square$

Note that for the stable set problem, the condition for dominance fixing is that $N_a(u) \subseteq N_a(v)$ for $u, v \in V$, where $N_a(u)$ is the neighborhood of u in the induced subgraph of G at node a . Hence to apply dominance fixing, only neighborhoods of nodes have to be checked. For other problems this condition might be harder to check.

We define Fix_a^0 and Fix_a^1 as the index-sets of variables fixed to 0 (resp. 1) in the SST-presolving or in the path from root to a in T_{presol} . Note that this also includes the variables that were merged in SST-presolving and whose representative variable gets branched on or fixed.

Lemma 6.8. *Let a be a node in T_{presol} and let $(i, j) \in S$ with $i < j$. If $i \in F_a^0 \cup \text{Fix}_a^0$, then $j \in \text{Fix}_a^0$.*

Proof. We can w.l.o.g. assume that i is a representative regarding merging. Otherwise, since we demand merging to only be applied for SST-pairs, there exists a representative $h < i$ with $(h, i) \in S$ and therefore also $(h, j) \in S$ and $i \in F_a^0 \cup \text{Fix}_a^0$. So we can argue about h instead of i .

Let $i \in F_a^0 \cup \text{Fix}_a^0$. First assume that x_i was branched or fixed to 0 after SST-presolving. Then in particular, no addition operation for $(i, j) \in S$ is applicable anymore when x_i is branched on or fixed. Hence, the dominance fixing condition for x_j regarding x_i is satisfied. For this note that $(i, j) \in S$ implies that $c_i = c_j$. Therefore, $j \in \text{Fix}_a^0$ by dominance fixing if it was not fixed already in SST-presolving. The variable x_j being fixed by SST-presolving would only contradict the lemma's statement if x_j was fixed to 1 by a fixation operation in SST-presolving. We claim that this can only be the case if also x_i was fixed to 1 in the same way contradicting $i \in F_a^0 \cup \text{Fix}_a^0$. If x_j was fixed to 1, there exists some k such that $a_k(\bar{S}^j \cap \mathcal{A}_k^j) > b_k$ by definition. Moreover, $(i, j) \in S$ implies that $S^j \subsetneq S^i$ and therefore, $\bar{S}^j \supsetneq \bar{S}^i$. Hence,

$$a_k(\bar{S}^i \cap \mathcal{A}_k^i) \geq a_k(\bar{S}^j \cap \mathcal{A}_k^j) > b_k$$

implies the claim. Thus, x_j cannot have been fixed to 1 by SST-presolving.

Finally, assume that x_i was fixed to 0 by SST-presolving. Then there exists some k such that

$$a_k(S_i) + a_{k,i} + a_k(\bar{S}_i \cap \mathcal{A}_k^i) > b_k.$$

On the one side, let $p \in S_i \cup \{i\} \cup (\bar{S}_i \cap \mathcal{A}_k^i)$ with $a_{k,p} > 0$. Then $p \in S_i \cup \{i\}$. But since $i \in S_j$ and $S_i \subsetneq S_j$, we also have $p \in S_j \subseteq S_j \cup \{j\} \cup (\bar{S}_j \cap \mathcal{A}_k^j)$. This shows that $a_k(S_j) + a_{k,j} + a_k(\bar{S}_j \cap \mathcal{A}_k^j)$ contains all positive terms of $a_k(S_i) + a_{k,i} + a_k(\bar{S}_i \cap \mathcal{A}_k^i)$.

On the other side, let $p \in S_j \cup \{j\} \cup (\bar{S}_j \cap \mathcal{A}_k^j)$ with $a_{k,p} < 0$. Assume that $p \notin S_i$, i.e., $p \in \bar{S}_i$. This implies that $p \in \{i\} \cup (\bar{S}_i \cap \mathcal{A}_k^i)$ because $a_{k,p} < 0$. So generally, we have $p \in S_i \cup \{i\} \cup (\bar{S}_i \cap \mathcal{A}_k^i)$ showing that $a_k(S_i) + a_{k,i} + a_k(\bar{S}_i \cap \mathcal{A}_k^i)$ contains all negative summands of $a_k(S_j) + a_{k,j} + a_k(\bar{S}_j \cap \mathcal{A}_k^j)$.

Both cases together show that $a_k(S_j) + a_{k,j} + a_k(\bar{S}_j \cap \mathcal{A}_k^j) > b_k$ holds, proving that x_j is fixed to 0 by a deletion operation in SST-presolving. $\square$

Proposition 6.9. *If in the generation of T_{presol} dominance fixing is used whenever applicable, the tree T_{presol} can be obtained by contracting some edges of T_{SST} . So in particular, this means that T_{presol} is a graph minor of T_{SST} , i.e., it is isomorphic to some graph obtained by contracting edges of a subgraph of T_{SST} .*

Proof. We show the result by defining an embedding of T_{presol} into T_{SST} . Let a be a node in T_{presol} . Then a is completely defined by the sets F_a^0 and F_a^1 . We map the node a to a node b in T_{SST} satisfying the following properties

$$\begin{aligned} F_a^0 &\subseteq F_b^0 \subseteq F_a^0 \cup \text{Fix}_a^0, \\ F_a^1 &\subseteq F_b^1 \subseteq F_a^1 \cup \text{Fix}_a^1. \end{aligned} \tag{2}$$

In particular, we choose a node b that is minimal with these properties, i.e., there exists no node β in T_{SST} with $F_b^0 \supsetneq F_\beta^0$ and $F_b^1 \supsetneq F_\beta^1$ with at least one inclusion being proper. We have to show that this mapping exists, is injective and preserves the ancestor-descendant relations.

That the mapping exists means that for any node a in T_{presol} there exists an image node b in T_{SST} . We construct such an image inductively on the depth of a . If the depth is 0, a is the root of T_{presol} which is mapped to the root of T_{SST} , because at this point the set of branched variables in both trees is empty. The induction hypothesis is that at depth d of T_{presol} all nodes have an image in T_{SST} as described. Let a' be a node in T_{presol} at depth $d+1$ and let a be the parent node of a' . Then, by the induction hypothesis, for a there exists an image b in T_{SST} with the described properties. Let x_j be the branching variable at b and x_i be the branching variable at a .

Assume that $j > i$. Then we have $i \in F_b^0 \cup F_b^1$ or there exists $h < i$ with $h \in F_b^0$ and $(h, i) \in S$. In the first case, since $F_b^0 \cup F_b^1 \subseteq F_a^0 \cup \text{Fix}_a^0 \cup F_a^1 \cup \text{Fix}_a^1$, $i \in F_b^0 \cup F_b^1$ contradicts the fact that x_i is the branching variable at a . In

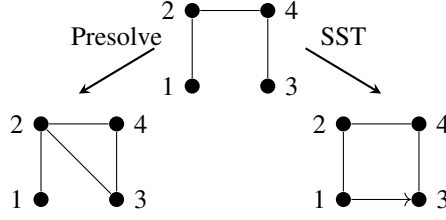


Figure 4: Stable set instance after one round of SST-presolving and after the addition of the SST-cuts displayed as directed edges.

the second case, if $h \in F_b^0 \subseteq F_a^0 \cup \text{Fix}_a^0$ and $(h, i) \in S$, we obtain by Lemma 6.8 that $i \in \text{Fix}_a^0$ leading to the same contradiction. Thus, the assumption is wrong and we have $j \leq i$.

To obtain an image b' of a' , we start with b and follow the branchings on variables x_k with $k \in \{j, j+1, \dots, i\}$ to 0 if $k \in F_{a'}^0 \cup \text{Fix}_a^0$ and to 1 if $k \in F_{a'}^1 \cup \text{Fix}_a^1$ if the variable is not already fixed. We have to check that in the construction of b' no variable in $(x_{j+1}, \dots, x_i)$ gets fixed to 0 even though its index is in $F_{a'}^1 \cup \text{Fix}_a^1$ and x_i is not fixed at all. Note that as discussed before, no 1 fixing is done in T_{SST} . Therefore, it suffices to check that x_i is not fixed to 0, instead of checking that it is not fixed at all. If some x_k with $j < k \leq i$ is fixed to 0 in T_{SST} , there exists x_h with $h < k$ that is branched to 0 on the path from root to b , i.e., $h \in F_b^0$, and $(h, k) \in S$. By the induction hypothesis, $h \in F_a^0 \cup \text{Fix}_a^0$. Further, since $(h, k) \in S$, the application of the addition operation in SST-presolving makes dominance fixing applicable implying that x_k is fixed to 0 in T_{presol} contradicting that $k \in F_{a'}^1 \cup \text{Fix}_a^1$ or that x_i is unfixed at a . This shows that we can always construct an image of a' with the described properties. Note that for this result we also use that no ancestor node β of b' in the path from b to b' in T_{SST} is infeasible. By the construction of b' we have $F_\beta^0 \subseteq F_a^0 \cup \text{Fix}_a^0$ and $F_\beta^1 \subseteq F_a^1 \cup \text{Fix}_a^1$. Therefore, if β was infeasible, a would be infeasible as well, contradicting a having a child node.

We next show injectivity of the mapping. Let a_1 and a_2 be distinct nodes in T_{presol} and images b_1 and b_2 in T_{SST} , respectively. Assume there exists $i \in F_{a_1}^0 \cap F_{a_2}^1$ or $i \in F_{a_1}^1 \cap F_{a_2}^0$. Then by construction $i \in F_{b_1}^1 \cap F_{b_2}^0$ or $i \in F_{b_1}^0 \cap F_{b_2}^1$ both showing that $b_1 \neq b_2$. Thus, we can assume that there exists a smallest index $i \in F_{a_1}^0 \cup F_{a_2}^1$ and $i \notin F_{a_2}^0 \cup F_{a_2}^1$.

If $i \in F_{a_1}^0$, by (2) we have $i \in F_{b_1}^0$. Assume that $i \in F_{b_2}^0$. Then $i \in \text{Fix}_{a_2}^0$, because $F_{b_2}^0 \subseteq F_{a_2}^0 \cup \text{Fix}_{a_2}^0$ holds because of (2) and $i \notin F_{a_2}^0$. But $i \in \text{Fix}_{a_2}^0$ is a consequence of SST-presolving and the variable branchings before i . Hence, by the minimality of i , $i \in \text{Fix}_{a_2}^0$ implies $i \in \text{Fix}_{a_1}^0$ contradicting that $i \in F_{a_1}^0$. Consequently, $i \notin F_{b_2}^0$ showing that $b_1 \neq b_2$. In the case where $i \in F_{a_1}^1$, we can show analogously that $b_1 \neq b_2$ proving injectivity.

Finally, the ancestor-descendant relations are preserved by the inductive construction of the images. Thus, this finishes the proof showing that the branch-and-bound tree T_{presol} is embedded in T_{SST} . $\square$

Remark 6.10. The proof of Proposition 6.9 actually shows the slightly stronger result that one can embed T_{presol} into T_{SST} such that each image of a node is placed with nodewise minimal distance to the root. In this end, one can also show uniqueness of the images for the defined embedding.

The left example in Figure 3 shows that T_{presol} can indeed be a proper minor of T_{SST} . This can happen for several reasons, for example the merge operation in the above example. Another reason is the application of dominance fixing on variables that initially were not symmetric. An example where such a fixing appears is displayed in Figure 4. On the one side, since in the SST-presolved graph we have $N(1) \subseteq N(3)$ as well as $N(1) \subseteq N(4)$, branching x_1 to 0 causes dominance fixing to fix x_3 and x_4 to 0. On the other side, branching x_1 to 0 for the graph with SST-cuts only implies that $x_3 = 0$ potentially causing T_{presol} to be smaller than T_{SST} .

Corollary 6.11. *If in the generation of T_{presol} dominance fixing is used whenever applicable, the tree T_{presol} is a graph minor of T_{OB} .*

Proof. This is a direct consequence of Proposition 6.3 and Proposition 6.9. $\square$

7 Computational Results

In the following, we demonstrate the effect of binary SST-presolving on three use cases. The first two apply to the maximum stable set/clique problem but two different solvers and the last one to the set cover problem.

7.1 Branch-and-Cut for the Stable Set Problem

In our first use case, we apply the branch-and-cut solver BACS (<https://github.com/dopt-TUDa/bacs>) for solving the maximum stable set problem, because it allows to apply the state-of-the-art symmetry handling methods of the framework SCIP [11]. The BACS implementation features a clique separator based on a greedy algorithm and, if unsuccessful, a combinatorial algorithm for the maximal weight clique problem implemented in SCIP. All specialized presolving/propagation methods of BACS are turned off.

SST-presolving (Definition 3.9, Lemma 4.1) is performed on the original graph before the optimization process starts. After the symmetry is computed in the second step of SST-presolving, BACS first checks whether there are symmetric components and possibly merges them. Since nodes that get merged during this operation do not satisfy the merge condition (otherwise they would have been merged in the first step of SST-presolving), we do not count them towards the merge operation even if the action applied on them is the same. SST-presolving is run iteratively until no further modifications of the graph are made, which means that the graph has trivial symmetry by Proposition 3.10. Instead of computing stabilizers for the leaders and followers, we filter the generators and therefore only operate on a subgroup of the actual symmetry group of the current graph for performance reasons. Anyhow, because of the iterative approach, we can still guarantee the presolved graph to have trivial symmetry.

Alternatively to SST-presolving, orbital branching can be used within the branch-and-bound tree generated by BACS. Here, we do not use a fixed branching order as in Section 6, but pick an orbit of largest size and branch on the variable with smallest index. Note that it uses filtering throughout the branch-and-bound tree and does not compute exact stabilizers. Moreover, orbital fixing is applied as described in [29].

To detect automorphisms of graphs, we apply bliss [16] and sassy [3]. To highlight the effect on the dual bound, we initialized the runs with a cutoff using the best primal value. All primal heuristics are turned off. We use a stringent leader choice as motivated by [12], which roughly means that we prioritize choosing leaders from maximum orbits as long as they fulfill a structure-preserving condition that is described in the following definition:

Definition 7.1. Let $L = (\ell_1, \dots, \ell_k)$. A family of SST-cuts $S(L)$ is called *stringent* if for all $i \in [k]$ we have $\ell_i \notin \{\ell_1, \dots, \ell_{i-1}\}$ and

$$N(u) \cap \left[\left(\bigcup_{j=1}^{i-1} O_j \right) \setminus \{\ell_1, \dots, \ell_{i-1}\} \right] \subseteq \{v \in V : \exists j < i \text{ with } u, v \in O_j\}$$

for all $u \in O_i$.

The computational experiments were carried out on three different testsets: the *mixed-testset*, consisting of 127 instances from various sources, the *Johnson-testset*, which includes 252 Johnson graphs of adequate complexity with potentially three copies for each Johnson graph with differently permuted node labels, and the *binary-code-testset*, containing 27 graphs processing Hamming distances between binary code words also of adequate complexity. Detailed information on the testsets can be found in the appendix.

Table 1 summarizes the structural effect that SST-presolving has on the instances of the three different testsets. More specifically, the arithmetic mean of the number of presolving rounds, the number of variables merged or deleted or edges added by SST-presolving over all instances in each testset is displayed. The same information is given for the individual instances in Tables 7, 8 and 9 in the appendix. Further, the average of the SST-presolving time in seconds as well as the total number of leaders $|L|$ and followers $|S(L)|$ computed is listed in the table.

Each operation of SST-presolving becomes active, while the presolving time is small compared to the overall solving time (see Tables 2, 3 and 4). The number of followers and therefore the number of SST-pairs strongly

Table 1: Analysis of iterative SST-presolving on different testsets.

Testset	#rounds	#merge	#del	#add	time	L	S(L)
Mixed	1.2	1079.9	72.3	2421.9	1.93	66.2	226.5
Johnson	2.7	4.9	339.4	40 121.5	0.46	8.4	2728.3
Binary-code	2.5	30.6	456.7	35 796.9	1.10	69.7	26 343.9

Table 2: Comparison of different symmetry handling techniques on the mixed-testset.

Setting	all (127)			all optimal (112)	
	#opt	time	#nodes	time	#nodes
base	112	27.12	2863.5	12.38	2011.3
base-sym	115	28.86	2435.8	14.06	1704.5
OB	119	19.58	2054.5	10.81	1718.8
presol	115	28.37	2596.3	13.65	1657.1
presol-dom	116	28.34	2586.5	13.68	1647.1
iter presol	115	28.38	2516.8	13.66	1616.7
iter presol-dom	116	28.27	2502.0	13.65	1604.3

increases from the mixed- to the Johnson-testset and binary-code-testset. Accordingly, the number of deletion and addition operations increases from testset to testset. This effect cannot be observed for the merge operation due to three outliers in the mixed-testset. The maximal number of iterations of SST-presolving is 6, meeting the expectation that the upper bound of n^2 is far from being tight (see Tables 7, 8 and 9).

Tables 2, 3 and 4 summarize the overall solution performance after SST-presolving was iteratively applied until no non-trivial symmetry was left (setting *iter presol*) and compares this to only applying one round of SST-presolving (setting *presol*) as well as default settings with SCIP symmetry handling (setting *base-sym*) and without SCIP symmetry handling (setting *base*). Further, the performance is examined where (iterated) SST-presolving is used and additionally dominance fixing is applied as a propagator in the branch-and-bound tree (settings *iter presol-dom* and *presol-dom*). Lastly, the setting *OB* describes the application of BACS with orbital branching, while all SCIP symmetry handling is turned off. For all three testsets, the number of instances solved to optimality within two hours, the shifted geometric means over all instances of the solution time and the number of nodes in the branch-and-bound tree are listed. Beyond that, the shifted geometric means over all instances of the solution time and the number of nodes in the branch-and-bound tree are also given on only the instances that were solved by all settings. For detailed information on the solution times see again Tables 7, 8 and 9.

Table 3: Comparison of different symmetry handling techniques on the Johnson-testset.

Setting	all (252)			all optimal (101)	
	#opt	time	#nodes	time	#nodes
base	102	913.36	5080.4	42.81	1462.0
base-sym	145	221.09	1451.6	6.55	108.4
OB	206	51.09	320.6	2.49	57.6
presol	157	159.46	2271.6	2.84	180.7
presol-dom	159	157.66	2267.7	2.73	169.6
iter presol	157	112.49	1478.8	2.17	111.6
iter presol-dom	160	109.49	1464.5	2.17	113.4

Table 4: Comparison of different symmetry handling techniques on the binary-code-testset.

Setting	all (27)			all optimal (14)	
	#opt	time	#nodes	time	#nodes
base	14	374.05	2206.0	23.12	321.3
base-sym	19	80.98	434.9	4.78	4.6
OB	23	22.29	151.3	1.26	3.0
presol	20	23.68	350.6	0.67	12.4
presol-dom	20	23.19	372.7	0.67	12.4
iter presol	20	19.92	284.3	0.52	0.1
iter presol-dom	20	19.52	297.2	0.53	0.1

It is important to highlight that the instances in ‘all optimal’ are often rather easy as can be seen by the very small average computational times. The reason is that the slowest method *base* is only able to solve such instances. Thus, the corresponding information should be analyzed with care.

The comparison of *iter presol* and *presol* shows that there is a rather small computational benefit in iterating SST-presolving and thus, handle symmetry completely. However, the speedup effect seems to correlate with the number of SST-presolving rounds needed (displayed in Tables 7, 8 and 9) since *iter presol* always performs a final check that no symmetry is left, while this computation is never done in the *presol* setting. Thus, for instances where only one presolving round is necessary, *presol* has to be a bit faster than *iter presol*. Further, the application of dominance fixing seems beneficial, since it slightly reduces the solution time and branch-and-bound nodes, while potentially solving more instances on all testsets. Thus, among the four SST-presolving applications *iter presol-dom* achieves the best results by a small margin.

For the mixed-testset, *iter presol-dom* solves four more instances than *base* with a solving time increase of around 4 %. Compared to *base-sym*, the solving time is slightly reduced by *iter presol-dom*, while still solving one more instance. For the other two testsets, however, the performance of *iter presol* is clearly superior to both *base* and *base-sym* with a reduction of about 51–95 % of solving time, while solving 15 more Johnson graphs and one more binary code graph than *base-sym*. On the instances solved to optimality by all settings, very similar tendencies can be seen.

For the stable set problem, among all variants, orbital branching clearly solves the largest number of instances – more than twice as many Johnson graphs as *base*. Further, significant time reductions can be observed. On the mixed-testset, *OB* only uses about 2/3 of the running time used in the other settings. On the Johnson graphs *OB* only takes about half of the time of *iter presol-dom*, but the latter is slightly faster than *OB* on ‘all optimal’. Only for the binary-code-testset, the solution time is larger than the one of *iter presol-dom*, but still three more instances are solved using *OB*.

That *OB* produces smaller branch-and-bound trees than *iter presol-dom* for solving the stable set problem is in contrast to Corollary 6.11. The reason for this difference can be the (dynamic) choice of the branching order used for orbital branching or the application of orbital fixing. Especially for the Johnson graphs, *OB* strongly outperforms all other methods on ‘all’ and solves more instances. As observed in [12], the effect of the order in which the leaders are chosen behaves differently for Johnson graphs than for large classes of other graphs. To reduce this effect, we compiled all Johnson graphs in three differently labeled versions for the Johnson-testset. Anyhow, this did not reduce the performance gap between *OB* and the other settings. Thus, we expect the overall leader sequence not to be the key performance driver.

7.2 Computational Results for CliSAT

Our second use case illustrates the possibility to perform iterative SST-presolving and then apply a “black-box” solver. Here, we use CliSAT [31, 5], which is a combinatorial solver for the maximum clique problem. Since the stable set problem is equivalent to the maximum clique problem on the complemented graph, we can apply CliSAT on the complemented graphs of our testset. The mixed-testset is not suitable for CliSAT, because

Table 5: Analysis of SST-presolving on Steiner Triple System set covering Instances.

Testset	#rounds	#merge	#fix	#add	time	$ L $	$ S(L) $
default	0.0	0.0	0.0	0.0	0.00	0.0	0.0
nosym	0.0	0.0	0.0	0.0	0.00	0.0	0.0
OB	0.0	0.0	0.0	0.0	0.00	0.0	0.0
presolve-addall	1.5	0.0	1.0	439.4	0.11	2.0	62.2
presolve-addonly	1.0	0.0	0.0	51.6	0.06	1.5	51.6
presolve	1.5	0.0	1.0	10.6	0.10	2.0	62.2

it contains weighted graphs and CliSAT does not allow weighted instances. Therefore, we exemplarily only performed computations on a subset of the Johnson-testset. That CliSAT does not support weighted instances is also an issue if SST-presolving applies merge operations, since they modify node weights. To avoid this, we ran SST-presolving without the merge operation. The results of Table 8 show that only a fraction of the Johnson graphs are influenced by the merge operation and therefore we expect the loss caused by omitting this operation being small.

We ran CliSAT on the same machine as the above computations with a time limit of one hour. Due to the three graphs Johnson_14/15/16_6_0 being very sparse, their complements are too large for CliSAT to properly run on these instances. Therefore they are not listed in the results table. Also note that the time for SST-presolving and complementing is not included in the results.

From the results listed in Table 10 in the appendix, we conclude that after iterative SST-presolving, CliSAT can solve 7 more instances in around one third of the time compared to CliSAT without SST-presolving. Note that only a binary executable of CliSAT is available, such that it can be seen as a true black-box solver for us.

7.3 Computational Results for Set Covering

In our third use case, we apply the binary SST-presolving steps for set covering as explained in Section 4.3. The set covering instance is read, the corresponding 0/1 matrix is detected and then iterative SST-presolving is performed. The matrix is then used for setting up a MIP in SCIP, which is then solved. Our implementation and test instances are available at <https://github.com/dopt-TUDa/setcovering>.

As instances we use 22 matrices formed by the incidence matrices of Steiner triple systems, as already done by [22] and proposed by Fulkerson et al. [9].

We use the following variants:

- *base*: turn off all symmetry handling in SCIP
- *base-sym*: SCIP default settings
- *OB*: apply orbital branching
- *presol*: apply SST-presolving with only one addition for each SST-pair
- *presol-addonly*: same as presolve, but only perform the addition operation
- *presol-addall*: apply SST-presolving with all possible addition operations

Table 5 shows the average number of presolving rounds (#rounds), average number of presolving operations (#merge, #fix, #add), the average time of SST-presolving in seconds, the average total number of leaders ($|L|$), and the average total number of SST-pairs ($|S(L)|$). As can be seen, the merge operation is never applied. The fixation operation is applied exactly once per instance, which is essential for good performance: the variant without fixation (*presol-addonly*) performs much worse (see below). The addition operation performs most work, where *presol-addall* adds a large number of rows. The running time of presolving is negligible, even for

Table 6: Comparison of SST-presolving performance on Steiner Triple System set covering instances.

Setting	all (22)			all optimal (11)	
	#opt	time	#nodes	time	#nodes
base	15	688.04	582101.4	162.79	159945.1
base-sym	14	597.55	553401.3	138.59	148634.0
OB	13	744.23	772409.5	170.43	204512.9
presol	15	547.76	441933.7	116.59	104215.4
presol-addall	15	608.37	539158.5	135.00	134238.0
presol-addonly	11	1038.56	1212834.6	299.11	406710.9

presol-addall. The number of leaders is only around 2 on average, but the number of SST-pairs is between 51.6 and 62.2.

Table 6 shows the number of instances solved to optimality (#opt) as well as the shifted geometric means of the number of nodes and time (in seconds). The right part shows the 11 instances that have been solved by all variants. We apply a time limit of one hour and initialize the runs using the best known primal solutions to avoid noise from finding primal solutions.

The results show that performing orbital branching (*OB*) is not very successful, while *nosym* solves one more instance than the default. Among the SST-presolving variants, *presolve* improves upon the default by about 18 % on ‘all optimal’. The other presolving variants are not much faster than the default (*presol-addall*) or much slower (*presol-addonly*). Thus, the basic SST-presolving version *presol* enables a significant speed-up with very limited amount of implementation effort and additional running time.

8 Conclusions

In this paper, we introduced presolving methods that, when applied iteratively, allow to remove all non-trivial symmetry. One of these methods allows to preserve problem structure. We investigated this presolving technique in three case studies.

As already mentioned in the introduction, one main question is whether a presolving method that produces an instance without non-trivial symmetry is advantageous in comparison to dynamic symmetry handling methods that allow to adapt symmetry handling to the solving process, e.g., through the choice of branching decisions. A complete answer to this answer is still open, but to obtain a partial answer, we proved that the branch-and-bound tree produced after SST-presolving is not larger than the tree of orbital branching with a fixed branching variable order and if no other methods are used. The computational results for the stable set problem show the dominance of orbital branching (as a dynamic technique and including orbital fixing), while presolving seems to be beneficial on the set covering instances. Thus, a more thorough investigation could be helpful. Moreover, finding more use cases for structure-preserving SST-presolving would be interesting.

It would also be interesting to close the gap between the lower and upper bounds on the number of presolving rounds in order to only leave non-trivial symmetry.

References

- [1] Anders, M., Brenner, S., Rattan, G.: Satsuma: Structure-Based Symmetry Breaking in SAT. In: S. Chakraborty, J.H.R. Jiang (eds.) 27th International Conference on Theory and Applications of Satisfiability Testing (SAT 2024), *Leibniz International Proceedings in Informatics (LIPIcs)*, vol. 305, pp. 4:1–4:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2024). DOI 10.4230/LIPIcs.SAT.2024.4

- [2] Anders, M., Codel, C., Heule, M.J.H.: Simplify, order, break, repeat. In: A. Ignatiev, S. Szeider (eds.) 29th International Conference on Theory and Applications of Satisfiability Testing (SAT 2026), vol. 377, pp. 4:1–4:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik (2026). DOI 10.4230/LIPIcs.SAT.2026.4
- [3] Anders, M., Schweitzer, P., Stieß, J.: Engineering a preprocessor for symmetry detection. In: L. Georgiadis (ed.) 21st International Symposium on Experimental Algorithms (SEA 2023), *Leibniz International Proceedings in Informatics (LIPIcs)*, vol. 265, pp. 1:1–1:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2023). DOI 10.4230/LIPIcs.SEA.2023.1
- [4] Bendotti, P., Fouilhoux, P., Rottner, C.: Orbitopal fixing for the full (sub-)orbitope and application to the unit commitment problem. *Mathematical Programming* **186**, 337–372 (2021). DOI 10.1007/s10107-019-01457-1
- [5] Instances for CliSAT – an efficient state-of-the-art exact algorithm for the maximum clique problem. Available at <https://github.com/psanse/CliSAT>
- [6] Colbourn, C., Dinitz, J. (eds.): *Handbook of Combinatorial Designs*, 2nd edn. Chapman and Hall/CRC (2006). DOI 10.1201/9781420010541
- [7] Devriendt, J., Bogaerts, B., Bruynooghe, M., Denecker, M.: Improved static symmetry breaking for SAT. In: N. Creignou, D. Le Berre (eds.) *Proceedings of Theory and Applications of Satisfiability Testing – SAT 2016 – 19th International Conference, LNCS*, vol. 9710, pp. 104–122. Springer (2016)
- [8] Friedman, E.J.: Fundamental domains for integer programs with symmetries. In: A. Dress, Y. Xu, B. Zhu (eds.) *Combinatorial Optimization and Applications, LNCS*, vol. 4616, pp. 146–153. Springer (2007). DOI 10.1007/978-3-540-73556-4_17
- [9] Fulkerson, D.R., Nemhauser, G.L., Trotter, L.E.: Two computationally difficult set covering problems that arise in computing the 1-width of incidence matrices of Steiner triple systems. In: M.L. Balinski (ed.) *Approaches to Integer Programming*, pp. 72–81. Springer, Berlin, Heidelberg (1974). DOI 10.1007/BFb0120689
- [10] Hojny, C.: Packing, partitioning, and covering symresacks. *Discrete Applied Mathematics* **283**, 689–717 (2020). DOI 10.1016/j.dam.2020.03.002
- [11] Hojny, C., Besançon, M., Bestuzheva, K., Borst, S., Chmiela, A., Dionísio, J., Eifler, L., Ghannam, M., Gleixner, A., Göß, A., Hoen, A., van der Hulst, R., Kamp, D., Koch, T., Kofler, K., Lentz, J., Maher, S.J., Mexi, G., Mühmer, E., Pfetsch, M.E., Pokutta, S., Serrano, F., Shinano, Y., Turner, M., Vigerske, S., Walter, M., Weninger, D., Xu, L.: The SCIP Optimization Suite 10.0. Technical report, Optimization Online (2025). URL <https://optimization-online.org/2025/11/the-scip-optimization-suite-10-0/>
- [12] Hojny, C., Jäger, A., Pfetsch, M.E., Verschae, J.: The impact of symmetry handling for the stable set problem via Schreier-Sims cuts. Preprint, Optimization Online (2026). URL <https://optimization-online.org/?p=24730>
- [13] Hojny, C., Pfetsch, M.E.: Polytopes associated with symmetry handling. *Mathematical Programming* **175**, 197–240 (2019). DOI 10.1007/s10107-018-1239-7
- [14] Jäger, A., Pfetsch, M.E.: Structure-preserving symmetry presolving for mixed-binary linear problems. In: S.S. Dey, M. Di Summa, D. Salvagnin (eds.) *Integer Programming and Combinatorial Optimization*, pp. 457–473. Springer, Cham (2026). DOI 10.1007/978-3-032-28691-8_30
- [15] Januschowski, T., Pfetsch, M.E.: Branch-cut-and-propagate for the maximum k -colorable subgraph problem with symmetry. In: T. Achterberg, J.C. Beck (eds.) *Proc. 8th International Conference, CPAIOR 2011, Lecture Notes in Computer Science*, vol. 6697, pp. 99–116. Springer (2011). DOI 10.1007/978-3-642-21311-3_11

- [16] Junttila, T., Kaski, P.: bliss: A tool for computing automorphism groups and canonical labelings of graphs. <https://users.aalto.fi/~tjunttil/bliss/> (2012)
- [17] Kaibel, V., Peinhardt, M., Pfetsch, M.E.: Orbitopal fixing. *Discrete Optimization* **8**(4), 595–610 (2011). DOI 10.1016/j.disopt.2011.07.001
- [18] Kaibel, V., Pfetsch, M.E.: Packing and partitioning orbitopes. *Mathematical Programming* **114**(1), 1–36 (2008). DOI 10.1007/s10107-006-0081-5
- [19] Kolliopoulos, S.G., Young, N.E.: Approximation algorithms for covering/packing integer programs. *Journal of Computer and System Sciences* **71**(4), 495–505 (2005). DOI 10.1016/j.jcss.2005.05.002
- [20] Liberti, L., Ostrowski, J.: Stabilizer-based symmetry breaking constraints for mathematical programs. *Journal of Global Optimization* **60**, 183–194 (2014). DOI 10.1007/s10898-013-0106-6
- [21] Linderöth, J., Núñez Ares, J., Ostrowski, J., Rossi, F., Smriglio, S.: Orbital conflict: Cutting planes for symmetric integer programs. *INFORMS Journal on Optimization* **3**(2), 139–153 (2021). DOI 10.1287/ijoo.2019.0044
- [22] Margot, F.: Pruning by isomorphism in branch-and-cut. *Mathematical Programming* **94**(1), 71–90 (2002). DOI 10.1007/s10107-002-0358-2
- [23] Margot, F.: Exploiting orbits in symmetric ILP. *Mathematical Programming* **98**(1–3), 3–21 (2003). DOI 10.1007/s10107-003-0394-6
- [24] Margot, F.: Small covering designs by branch-and-cut. *Mathematical Programming* **94**(2), 207–220 (2003). DOI 10.1007/s10107-002-0316-z
- [25] Margot, F.: Symmetry in integer linear programming. In: M. Jünger, T.M. Liebling, D. Naddef, G.L. Nemhauser, W.R. Pulleyblank, G. Reinelt, G. Rinaldi, L.A. Wolsey (eds.) *50 Years of Integer Programming*, pp. 647–686. Springer (2010)
- [26] Ostrowski, J.: Symmetry in integer programming. Ph.D. thesis, Lehigh University (2008)
- [27] Ostrowski, J., Anjos, M.F., Vannelli, A.: Modified orbital branching for structured symmetry with an application to unit commitment. *Mathematical Programming* **150**(1), 99–129 (2015). DOI 10.1007/s10107-014-0812-y
- [28] Ostrowski, J., Linderöth, J., Rossi, F., Smriglio, S.: Orbital branching. *Mathematical Programming* **126**(1), 147–178 (2011). DOI 10.1007/s10107-009-0273-x
- [29] Pfetsch, M.E., Rehn, T.: A computational comparison of symmetry handling methods for mixed integer programs. *Mathematical Programming Computation* **11**(1), 37–93 (2019). DOI 10.1007/s12532-018-0140-y
- [30] Salvagnin, D.: Symmetry breaking inequalities from the Schreier-Sims table. In: W.J. van Hoeve (ed.) *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pp. 521–529. Springer, Cham (2018). DOI 10.1007/978-3-319-93031-2_37
- [31] San Segundo, P., Furini, F., Álvarez, D., Pardalos, P.M.: CliSAT: A new exact algorithm for hard maximum clique problems. *European Journal of Operational Research* **307**(3), 1008–1025 (2023). DOI 10.1016/j.ejor.2022.10.028
- [32] Verschae, J., Villagra, M., von Niederhäusern, L.: On the geometry of symmetry breaking inequalities. In: M. Singh, D.P. Williamson (eds.) *Integer Programming and Combinatorial Optimization*, pp. 73–88. Springer, Cham (2021)
- [33] Verschae, J., Villagra, M., von Niederhäusern, L.: On the geometry of symmetry breaking inequalities. *Mathematical Programming* **197**(2), 693–719 (2023). DOI 10.1007/s10107-022-01819-2

A Further Information on Computational Results

We use a developer version of SCIP 10.0.4 (GitHash: 36ba424) and CPLEX 22.1.2.0 as LP solver. The experiments were run on a Linux cluster with 3.5 GHz Intel Xeon E5-1620 Quad-Core CPUs, having 32 GB main memory and 10 MB cache each. All computations were run single-threaded. The experiments for the stable set instances use a time limit of two hours, while the ones for CliSAT and set covering use a time limit of one hour.

A.1 Stable Set Instances

Table 1 of [12] contains a collection of sources of interesting instances for the stable set or maximum clique problem. The instances generated for the maximum clique problem were complemented before we started the computations. Then we ran BACS with all settings considered in Section 7 on all of the instances listed in the table. We removed all instances with trivial symmetry. Moreover, we removed all instances that appeared too easy which means instances that were solvable in less than one second for all of the 7 different settings. All remaining 127 instances form the *mixed-testset*.

The *Johnson-testset* consists of 252 Johnson graphs. These graphs were selected by first generating all Johnson graphs with 100 to 10000 nodes and 1000 to 1000000 edges. Then, the graphs were randomly permuted with two different seeds such that each graph appears in three differently labeled versions. Then again, we ran BACS with all the 7 considered settings. We removed all instances that were too easy like we did for the *mixed-testset*.

For the *binary-code-testset*, we generated graphs from coding theory, denoted by `BinaryCode_n_d`. Each node represents a binary code word of length n and an edge between two nodes exists if the Hamming distance of the two nodes is maximally d . For the selection of instances for the testset we proceeded exactly as for the *Johnson-testset* and this way obtained 27 instances.

The Johnson- as well as the binary-code-testset is available on github (<https://github.com/dopt-TUDa/SymGraphs>).

A.2 Set Covering Instances

A *Steiner Triple System (STS)* consists of a groundset S of size $n \in \mathbb{N}$, and a family of triples $(S_1, \dots, S_m)$ of S such that every pair of items in S is contained in exactly one of the triples. It is well-known that an STS can only exist for $n \equiv 1, 3 \pmod{6}$. The set covering instances are then formed by using the incidence vectors of the triples as the rows of the 0/1-matrix A , where its columns are indexed by S .

For each $n \equiv 1, 3 \pmod{6}$ several STS can exist and we use the same construction as the one in Sage (see <https://github.com/sagemath/sage/blob/develop/src/sage/combinat/designs/bibd.py>). For instance, for $n \equiv 3 \pmod{6}$ the following produces an STS, see, e.g., [6, Chapter II.2]: for odd $k \in \mathbb{N}$, let $V := \{0, \dots, k-1\}$, $S := V \times \mathbb{Z}_3$ (thus $n := 3k$) and define the triples as

$$\{(x, 0), (x, 1), (x, 2)\} : x \in V \cup \{(x, i), (y, i), ((x+y) \bmod k, (i+1) \bmod 3)\} : x, y \in V, x < y, i \in \mathbb{Z}_3\}.$$

A.3 Detailed Results

In Tables 7, 8 and 9 results of the computations described in Section 7 for each instance in the three different testsets are displayed. The solution time and the time used for SST-presolving are shown in the first two columns after the instance names. By $p(n)$ and $p(m)$ we denote the number of nodes and edges respectively in the presolved graph. This means the columns with titles $p(n)/n$ and $p(m)/m$ contain the ratios of nodes and edges of the presolved graph vs. the initial graph, respectively. The columns #merge, #del and #add summarize the number of nodes fixed or edges added by each presolving method. Further, the column #rounds displays the number of SST-presolving rounds applied before the procedure terminated.

Table 7: Analysis of Presolving for Mixed-testset.

name	time	pre time	$p(n)/n$	$p(m)/m$	#merge	#del	#add	#rounds
109	1.06	0.88	0.96	0.90	0	167	0	1
118	3.59	2.83	0.97	0.91	0	250	0	1
103	1.54	0.85	0.98	0.96	0	73	0	1
112	5.81	4.45	0.96	0.89	0	363	0	1
108	2.12	1.61	0.95	0.88	0	258	0	1
114	8.37	7.18	0.92	0.80	0	659	0	1
107	1.88	1.48	0.95	0.87	0	276	0	1
116	1.74	1.44	0.96	0.89	0	238	0	1
101	1.51	0.89	0.98	0.94	0	102	0	1
104	1.51	0.98	0.97	0.94	0	123	0	1
113	2.63	1.91	0.98	0.96	0	125	0	1
111	5.87	5.14	0.96	0.87	0	390	0	1
117	1.61	1.29	0.96	0.89	0	201	0	1
120	5.40	3.24	0.97	0.91	0	260	0	1
110	1.23	0.92	0.99	0.97	0	64	0	1
115	2.04	1.59	0.94	0.85	0	292	0	1
119	2.29	1.10	0.98	0.95	0	89	0	1
105	1.48	1.20	0.95	0.86	0	246	0	1
vc-exact_038.gr-compl	5.50	0.01	0.97	0.99	4	19	0	2
ref-25-30-02-compl-weigh	52.38	0.02	0.99	0.98	0	4	0	1
ref-25-35-00-compl-weigh	346.64	0.01	0.99	0.99	0	2	0	1
ref-20-30-04-compl-weigh	18.41	0.01	1.00	0.99	0	1	0	1
ref-25-25-03-compl-weigh	22.03	0.01	0.99	0.98	0	4	0	1
ref-25-40-05-compl-weigh	230.49	0.01	0.99	0.99	0	2	0	1
ref-25-30-01-compl-weigh	43.81	0.01	0.99	0.98	0	3	0	1
ref-25-25-01-compl-weigh	28.34	0.01	0.97	0.94	0	10	0	1
ref-20-30-08-compl-weigh	0.99	0.01	0.99	0.99	0	2	0	1
ref-25-25-07-compl-weigh	4.90	0.01	0.98	0.96	0	8	0	1
ref-30-30-01-compl-weigh	620.61	0.01	0.99	0.98	0	4	0	1
ref-25-40-06-compl-weigh	32.54	0.01	1.00	0.99	0	1	0	1
ref-20-30-00-compl-weigh	3.10	0.01	1.00	0.99	0	1	0	1
ref-20-30-02-compl-weigh	2.76	0.01	1.00	0.99	0	1	0	1
ref-25-35-04-compl-weigh	67.77	0.01	0.99	0.99	0	2	0	1
ref-25-35-03-compl-weigh	29.13	0.01	1.00	1.00	0	1	0	1
ref-20-50-09-compl-weigh	8.75	0.00	0.96	0.93	0	12	0	1
ref-25-30-08-compl-weigh	12.48	0.02	0.99	0.98	0	4	0	1
ref-25-30-03-compl-weigh	40.68	0.01	0.99	0.99	0	2	0	1
ref-20-30-05-compl-weigh	2.81	0.01	0.99	0.98	0	2	0	1
ref-25-25-06-compl-weigh	48.78	0.01	0.98	0.96	0	8	0	1
ref-30-30-08-compl-weigh	257.95	0.01	1.00	0.99	0	1	0	1
ref-30-30-06-compl-weigh	444.17	0.01	0.99	0.98	0	4	0	1
ref-25-30-09-compl-weigh	47.62	0.01	0.98	0.96	0	7	0	1
ref-25-25-04-compl-weigh	22.58	0.01	1.00	1.00	0	1	0	1
ref-30-30-02-compl-weigh	158.75	0.02	1.00	0.99	0	2	0	1
ref-25-35-08-compl-weigh	18.65	0.01	0.98	0.96	0	7	0	1
ref-25-25-09-compl-weigh	9.13	0.01	0.98	0.96	0	8	0	1
ref-25-35-07-compl-weigh	90.04	0.01	0.99	0.99	0	2	0	1
ref-30-30-03-compl-weigh	141.63	0.01	1.00	1.00	0	1	0	1
ref-25-30-07-compl-weigh	56.17	0.01	0.99	0.99	0	2	0	1
ref-20-30-03-compl-weigh	4.51	0.01	1.00	0.99	0	1	0	1
ref-25-35-05-compl-weigh	4.91	0.01	0.99	0.99	0	2	0	1
ref-25-35-02-compl-weigh	74.73	0.01	1.00	0.99	0	1	0	1
ref-30-30-04-compl-weigh	344.27	0.01	0.99	0.98	0	4	0	1
ref-30-30-05-compl-weigh	428.27	0.01	1.00	0.99	0	2	0	1
ref-25-25-00-compl-weigh	13.07	0.01	0.98	0.96	0	8	0	1
ref-25-25-02-compl-weigh	15.72	0.01	0.99	0.98	0	3	0	1
ref-20-30-07-compl-weigh	0.24	0.01	0.99	0.98	0	2	0	1
ref-20-30-01-compl-weigh	2.97	0.01	1.00	0.99	0	1	0	1
ref-30-40-06-compl-weigh	162.55	0.01	1.00	0.99	0	2	0	1
ref-30-30-00-compl-weigh	38.62	0.01	1.00	0.99	0	2	0	1
ref-30-30-07-compl-weigh	240.11	0.01	0.99	0.98	0	5	0	1

continued on next page

name	time	pre time	$p(n)/n$	$p(m)/m$	#merge	#del	#add	#rounds
ref-25-40-00-compl-weigh	1.63	0.00	1.00	0.99	0	1	0	1
ref-30-40-04-compl-weigh	1243.65	0.01	1.00	0.99	0	1	0	1
ref-20-40-02-compl-weigh	15.66	0.01	1.00	0.99	0	1	0	1
ref-30-30-09-compl-weigh	171.37	0.01	0.99	0.98	0	5	0	1
ref-20-50-01-compl-weigh	2.41	0.01	0.99	0.98	0	4	0	1
ref-20-40-00-compl-weigh	2.07	0.01	0.99	0.98	0	2	0	1
ref-25-30-00-compl-weigh	58.51	0.01	0.99	0.98	0	3	0	1
ref-20-50-03-compl-weigh	1.39	0.01	1.00	0.99	0	1	0	1
ref-25-35-01-compl-weigh	89.69	0.01	1.00	1.00	0	1	0	1
ref-25-25-08-compl-weigh	40.14	0.01	0.97	0.94	0	11	0	1
ref-25-30-04-compl-weigh	83.39	0.01	1.00	0.99	0	1	0	1
a265032_2dc.2048	7200.00	0.39	1.00	1.00	0	1	220	1
a265032_1zc.2048	7200.00	0.07	0.97	1.21	0	61	10479	1
a265032_1et.512	0.36	0.01	0.47	0.56	3	12	315	2
a265032_1zc.4096	7200.03	0.20	0.98	1.31	0	66	31217	1
a265032_1tc.1024	11.33	0.01	0.62	0.65	1	1	27	1
a265032_1tc.2048	6389.51	0.02	0.50	0.50	1	0	14	1
a265032_1et.2048	7200.00	0.09	0.49	0.52	3	15	628	6
a265032_1zc.512	156.56	0.01	0.92	1.11	0	41	1783	1
a265032_2dc.512	27.64	0.03	1.00	1.00	0	1	144	1
a265032_1zc.1024	7200.00	0.02	0.96	1.25	0	45	5506	2
a265032_1dc.512	4.16	0.00	1.00	1.00	0	1	32	1
a265032_1dc.1024	783.49	0.01	1.00	1.00	0	1	36	1
a265032_1et.1024	15.14	0.04	0.61	0.72	3	15	828	4
a265032_2dc.1024	3407.70	0.12	1.00	1.00	0	1	180	1
a265032_1dc.2048	7200.01	0.05	1.00	1.00	0	1	40	1
relational_2	66.26	66.25	0.00	0.03	24749	0	1225	1
relational_3	0.36	0.27	0.01	0.01	0	0	0	1
relational_1	8.43	8.43	0.00	0.00	62250	1	0	1
Alchemy_11	19.13	0.03	0.98	1.00	39	0	0	1
relational_5	9.74	9.74	0.00	0.00	49999	0	0	1
relational_4	71.34	70.71	0.01	0.21	99	0	9900	1
116-compl-weigh	2.31	1.53	0.97	0.91	0	185	0	1
105-compl-weigh	3.36	1.43	0.95	0.86	0	236	0	1
104-compl-weigh	2.73	1.07	0.97	0.94	0	123	0	1
107-compl-weigh	3.20	1.86	0.95	0.88	0	265	0	1
102-compl-weigh	0.96	0.53	0.99	0.96	0	51	0	1
101-compl-weigh	1.62	0.92	0.98	0.94	0	101	0	1
115-compl-weigh	3.13	1.94	0.95	0.86	0	264	0	1
117-compl-weigh	3.78	1.44	0.96	0.90	0	184	0	1
112-compl-weigh	58.97	5.29	0.96	0.89	0	347	0	1
120-compl-weigh	6.79	3.58	0.97	0.92	0	250	0	1
114-compl-weigh	13.86	10.74	0.93	0.81	0	582	0	1
109-compl-weigh	2.05	1.00	0.96	0.90	0	165	0	1
119-compl-weigh	2.32	1.07	0.98	0.96	0	80	0	1
106-compl-weigh	1.32	0.62	0.98	0.94	0	87	0	1
110-compl-weigh	2.48	0.97	0.99	0.97	0	61	0	1
103-compl-weigh	1.33	0.87	0.98	0.96	0	73	0	1
108-compl-weigh	3.31	1.98	0.96	0.88	0	247	0	1
111-compl-weigh	10.13	6.26	0.96	0.88	0	358	0	1
118-compl-weigh	7.38	3.20	0.97	0.91	0	243	0	1
113-compl-weigh	6.64	1.93	0.98	0.96	0	113	0	1
MANN_a45comp	140.48	0.01	1.00	1.36	0	1	709	1
MANN_a27comp	2.94	0.00	0.99	1.81	0	4	578	2
keller4comp	1.47	0.00	0.88	0.95	0	21	966	2
keller5comp	2369.87	0.07	0.94	1.09	0	43	14862	2
keller6comp	7200.00	1.97	0.97	1.09	0	116	165635	3
hamming10-4comp	7200.00	0.14	0.79	1.01	0	220	32827	2
MANN_a81comp	7200.00	0.11	1.00	1.94	0	6	6135	2
monoton-9-compl	7200.00	0.03	1.00	1.01	0	3	486	1
monoton-8-compl	7014.61	0.01	0.99	1.01	0	3	418	1
monoton-7-compl	59.38	0.01	0.99	1.02	0	3	350	1
12-20-6-6-compl-weigh	7200.02	0.58	1.00	0.99	0	7	0	1
01-11-4-4-compl-weigh	1.32	0.00	0.85	1.05	1	21	572	2

continued on next page

name	time	pre time	$p(n)/n$	$p(m)/m$	#merge	#del	#add	#rounds
04-14-6-6-compl-weigh	82.19	0.16	0.89	0.97	0	85	21 360	2
05-16-4-5-compl-weigh	1.66	0.00	0.98	1.00	0	3	104	1

Table 8: Analysis of Presolving for Johnson-testset.

name	time	pre time	$p(n)/n$	$p(m)/m$	#merge	#del	#add	#rounds
Johnson_14_3_1	5.43	0.03	0.39	0.18	0	221	1191	3
Johnson_14_4_1	13.09	0.33	0.50	0.27	0	500	5116	2
Johnson_17_4_2	7200.02	1.13	0.64	0.73	0	858	163 299	2
Johnson_19_3_1	134.51	0.26	0.22	0.06	4	750	4618	3
Johnson_14_4_3	0.92	0.04	0.91	1.45	0	91	12 161	2
Johnson_14_3_2	0.35	0.01	0.75	1.07	0	91	2897	2
Johnson_15_6_5	7200.14	0.43	0.98	1.47	0	105	69 268	1
Johnson_16_4_3	3.22	0.10	0.93	1.50	0	120	27 056	2
Johnson_13_5_0	202.77	0.07	0.92	1.41	0	98	20 078	1
Johnson_13_5_2	1005.38	0.61	0.53	0.30	0	602	8891	2
Johnson_18_4_2	7200.01	1.79	0.66	0.81	0	1029	285 016	1
Johnson_14_6_5	7200.00	0.20	0.97	1.45	0	91	36 212	2
Johnson_15_3_1	5.46	0.06	0.30	0.09	3	317	1095	3
Johnson_12_5_0	37.27	0.02	0.95	1.49	0	37	4893	1
Johnson_17_4_3	7200.00	0.13	0.94	1.51	0	136	37 850	1
Johnson_10_4_2	1.74	0.00	0.54	0.31	1	95	157	3
Johnson_17_3_1	118.99	0.13	0.34	0.13	2	450	2165	3
Johnson_18_3_2	22.89	0.04	0.81	1.30	0	153	11 300	2
Johnson_15_4_2	7200.01	0.38	0.57	0.58	0	587	51 707	2
Johnson_14_4_2	3738.91	0.21	0.53	0.49	0	467	25 579	2
Johnson_14_6_0	7200.00	0.12	0.99	1.65	0	38	28 382	1
Johnson_11_4_3	459.63	0.00	0.83	1.23	0	55	2416	1
Johnson_16_4_0	1.17	0.74	0.54	0.39	8	831	70 578	2
Johnson_14_6_1	7200.00	1.08	0.84	1.08	0	482	197 400	1
Johnson_11_4_2	0.46	0.04	0.37	0.20	0	207	1415	1
Johnson_13_4_2	209.20	0.14	0.48	0.40	0	369	11 561	1
Johnson_13_4_3	167.30	0.05	0.89	1.41	0	78	7770	3
Johnson_13_5_4	7200.00	0.05	0.94	1.47	0	78	15 001	2
Johnson_12_5_2	0.56	0.26	0.42	0.22	0	462	7247	3
Johnson_15_4_3	3223.60	0.10	0.92	1.48	0	105	18 650	1
Johnson_20_3_1	263.89	0.37	0.24	0.08	3	861	5562	3
Johnson_14_5_2	7200.00	1.49	0.57	0.33	0	856	9689	2
Johnson_15_5_3	7200.00	1.40	0.76	0.94	0	734	231 013	2
Johnson_10_4_1	0.02	0.01	0.40	0.21	4	122	612	2
Johnson_13_4_0	0.11	0.06	0.70	0.79	5	213	16 791	2
Johnson_11_3_1	0.51	0.00	0.44	0.20	0	92	172	2
Johnson_18_5_4	7200.04	1.39	0.98	1.55	0	153	162 546	3
Johnson_15_5_4	7200.01	0.21	0.97	1.49	0	105	41 823	2
Johnson_15_4_0	0.58	0.35	0.59	0.49	7	559	46 451	2
Johnson_18_4_3	7200.00	0.25	0.95	1.53	0	153	53 382	3
Johnson_14_4_0	0.25	0.15	0.64	0.62	6	359	29 569	2
Johnson_17_5_4	7200.04	0.69	0.98	1.52	0	136	104 644	2
Johnson_19_3_0	0.38	0.31	0.35	0.11	10	619	4879	3
Johnson_16_4_2	7200.00	0.65	0.61	0.66	0	713	95 023	2
Johnson_12_4_1	0.17	0.07	0.52	0.30	0	239	1958	2
Johnson_15_5_0	4.40	0.82	0.87	1.24	0	400	186 452	1
Johnson_13_4_1	0.44	0.17	0.51	0.28	0	351	2936	2
Johnson_17_4_0	2.44	1.46	0.50	0.32	13	1188	105 649	2
Johnson_16_3_1	12.29	0.08	0.32	0.11	2	379	1514	3
Johnson_14_6_4	7200.13	1.28	0.78	0.94	0	659	189 010	1
Johnson_15_4_1	109.14	0.65	0.51	0.26	0	673	5298	2
Johnson_12_5_4	7200.00	0.02	0.92	1.42	0	66	7979	1
Johnson_17_3_0	0.15	0.14	0.32	0.09	29	436	3003	4
Johnson_18_3_0	0.25	0.22	0.35	0.11	32	497	4470	4

continued on next page

name	time	pre time	$p(n)/n$	$p(m)/m$	#merge	#del	#add	#rounds
Johnson_14_5_1	8.31	1.09	0.51	0.38	6	966	82209	2
Johnson_16_4_1	572.00	1.35	0.51	0.26	0	894	8339	2
Johnson_16_6_0	7200.01	2.65	0.96	1.52	0	320	505359	1
Johnson_12_4_3	7200.00	0.01	0.87	1.34	0	66	4610	2
Johnson_14_5_4	7200.00	0.09	0.95	1.50	0	91	26249	1
Johnson_13_5_3	7200.01	0.28	0.65	0.70	0	453	45049	1
Johnson_14_5_0	468.01	0.25	0.89	1.33	0	211	67404	1
Johnson_14_3_0	0.04	0.03	0.24	0.04	37	240	683	5
Johnson_12_5_1	27.56	0.09	0.64	0.61	4	278	16387	2
Johnson_12_5_3	3073.71	0.11	0.57	0.54	0	338	15897	1
Johnson_16_3_2	65.92	0.02	0.79	1.22	0	120	6306	2
Johnson_16_5_4	7200.02	0.34	0.97	1.47	0	120	63005	1
Johnson_20_3_2	257.13	0.06	0.83	1.41	0	190	20233	2
Johnson_13_3_1	3.16	0.02	0.45	0.22	0	156	574	2
Johnson_18_3_1	23.81	0.20	0.22	0.06	3	633	2991	3
Johnson_12_4_2	4.39	0.04	0.43	0.30	0	283	4488	1
Johnson_16_3_0	0.09	0.08	0.34	0.11	28	342	2517	4
Johnson_13_5_1	21.72	0.32	0.57	0.47	5	550	38685	2
Johnson_19_4_3	7200.01	0.36	0.96	1.55	0	171	73550	2
Johnson_10_4_0	0.02	0.00	0.89	1.31	0	24	844	1
Johnson_15_6_0	7200.00	0.63	0.98	1.61	0	123	137706	1
Johnson_17_3_2	7200.01	0.02	0.80	1.27	0	136	8575	3
Johnson_20_3_0	0.56	0.48	0.33	0.09	12	756	5862	3
Johnson_11_3_2	0.17	0.00	0.67	0.83	0	55	694	3
Johnson_11_4_1	0.04	0.03	0.36	0.17	7	203	1300	3
Johnson_10_3_1	0.01	0.00	0.26	0.08	2	87	106	4
Johnson_20_4_3	7200.00	0.47	0.96	1.56	0	190	98564	1
Johnson_14_5_3	7200.02	0.67	0.71	0.84	0	587	112218	2
Johnson_16_6_5	7200.43	0.98	0.99	1.49	0	120	124404	1
Johnson_12_3_1	0.76	0.02	0.44	0.20	1	123	306	3
Johnson_11_4_0	0.03	0.01	0.83	1.28	3	52	3778	2
Johnson_14_6_5_perm123	7200.06	0.25	0.97	1.32	0	77	26166	2
Johnson_14_4_2_perm123	7200.00	0.24	0.59	0.52	0	409	19789	2
Johnson_15_6_0_perm123	7200.00	0.96	0.98	1.61	0	123	137860	3
Johnson_10_3_1_perm123	0.01	0.00	0.25	0.07	2	88	112	5
Johnson_18_3_0_perm123	0.32	0.32	0.06	0.00	144	621	1837	6
Johnson_11_3_2_perm123	0.53	0.00	0.68	0.77	0	52	587	3
Johnson_13_5_0_perm123	590.31	0.08	0.93	1.27	0	86	14651	3
Johnson_16_4_1_perm123	700.02	1.62	0.51	0.26	0	885	5590	2
Johnson_12_4_1_perm123	0.23	0.08	0.54	0.31	0	227	1240	2
Johnson_15_4_1_perm123	106.95	0.80	0.51	0.26	0	672	4866	3
Johnson_11_4_0_perm123	0.09	0.01	0.84	1.24	3	51	3538	2
Johnson_13_3_1_perm123	3.71	0.04	0.46	0.22	1	154	580	4
Johnson_13_5_1_perm123	28.49	0.51	0.57	0.47	5	550	42468	3
Johnson_12_4_2_perm123	24.02	0.06	0.50	0.35	0	249	3824	2
Johnson_12_5_0_perm123	50.56	0.02	0.95	1.49	0	37	4893	2
Johnson_18_4_3_perm123	7200.00	0.33	0.96	1.30	0	111	31678	3
Johnson_14_5_3_perm123	7200.00	0.72	0.74	0.79	0	525	79581	2
Johnson_14_3_0_perm123	0.04	0.03	0.22	0.03	47	237	733	5
Johnson_14_5_2_perm123	7200.00	1.61	0.57	0.34	0	852	10963	3
Johnson_12_5_1_perm123	98.98	0.14	0.67	0.60	4	255	13475	3
Johnson_16_6_0_perm123	7200.01	3.05	0.96	1.35	0	300	357544	2
Johnson_12_5_2_perm123	23.12	0.21	0.56	0.31	0	352	1573	2
Johnson_14_5_0_perm123	6137.20	0.30	0.91	1.17	0	186	44837	3
Johnson_13_4_1_perm123	0.26	0.24	0.35	0.15	7	456	5544	4
Johnson_12_4_3_perm123	7200.00	0.02	0.88	1.22	0	60	3534	3
Johnson_15_4_0_perm123	0.77	0.48	0.58	0.49	7	563	48865	3
Johnson_15_5_0_perm123	7.81	0.93	0.88	1.10	0	357	126632	2
Johnson_17_3_0_perm123	0.20	0.20	0.08	0.00	118	505	1565	8
Johnson_16_4_3_perm123	0.90	0.12	0.95	1.27	0	92	15634	3
Johnson_17_4_0_perm123	3.26	1.96	0.49	0.32	9	1194	110900	3
Johnson_10_4_2_perm123	0.09	0.01	0.33	0.14	0	141	346	2
Johnson_17_3_2_perm123	7200.01	0.04	0.84	1.16	0	106	6102	2
Johnson_15_5_3_perm123	7200.06	2.53	0.75	0.94	0	736	241717	4

continued on next page

name	time	pre time	$p(n)/n$	$p(m)/m$	#merge	#del	#add	#rounds
Johnson_18_3_2_perm123	51.42	0.05	0.83	1.22	0	140	9629	3
Johnson_20_3_1_perm123	12.34	0.45	0.17	0.04	3	941	7002	4
Johnson_16_5_4_perm123	7200.00	0.51	0.98	1.27	0	104	37861	2
Johnson_14_5_4_perm123	7200.01	0.14	0.96	1.31	0	76	17217	3
Johnson_16_3_2_perm123	10.02	0.02	0.82	1.15	0	99	4877	2
Johnson_14_6_4_perm123	7200.02	1.80	0.79	0.95	0	645	199811	3
Johnson_18_3_1_perm123	3.41	0.22	0.17	0.04	3	672	3625	3
Johnson_16_4_0_perm123	1.51	0.94	0.54	0.40	8	837	75375	4
Johnson_11_3_1_perm123	0.39	0.01	0.44	0.19	1	92	130	4
Johnson_13_4_2_perm123	3182.78	0.11	0.54	0.44	0	326	9773	2
Johnson_14_5_1_perm123	126.14	1.28	0.55	0.39	6	885	65364	3
Johnson_14_4_0_perm123	0.35	0.20	0.64	0.62	6	359	30024	2
Johnson_17_3_1_perm123	40.57	0.16	0.29	0.10	3	483	2867	5
Johnson_18_5_4_perm123	7200.07	2.02	0.98	1.32	0	134	98366	3
Johnson_12_5_4_perm123	7200.00	0.03	0.93	1.27	0	57	5507	2
Johnson_16_6_5_perm123	7200.40	1.43	0.99	1.34	0	102	87687	2
Johnson_20_3_0_perm123	0.70	0.69	0.13	0.01	93	896	1858	6
Johnson_16_3_0_perm123	0.17	0.13	0.42	0.15	20	307	1921	5
Johnson_16_3_1_perm123	0.75	0.11	0.21	0.05	2	443	2099	5
Johnson_17_4_2_perm123	7200.01	1.35	0.70	0.73	0	704	114395	2
Johnson_20_4_3_perm123	7200.00	0.91	0.97	1.30	0	133	54624	4
Johnson_10_4_1_perm123	0.02	0.01	0.40	0.20	5	122	693	4
Johnson_14_4_1_perm123	18.20	0.38	0.52	0.27	0	483	2252	3
Johnson_12_3_1_perm123	0.56	0.01	0.44	0.20	1	123	325	3
Johnson_11_4_3_perm123	2258.62	0.01	0.85	1.09	0	49	1625	2
Johnson_13_4_0_perm123	0.15	0.09	0.70	0.79	5	213	17158	4
Johnson_15_5_4_perm123	7200.01	0.31	0.97	1.31	0	87	27619	3
Johnson_14_6_1_perm123	7200.00	1.37	0.85	1.00	0	456	147688	2
Johnson_15_6_5_perm123	7200.10	0.61	0.98	1.33	0	88	48602	2
Johnson_14_3_1_perm123	50.27	0.05	0.51	0.27	1	179	655	4
Johnson_14_6_0_perm123	7200.00	0.17	0.99	1.62	0	38	27148	2
Johnson_13_5_3_perm123	7200.00	0.30	0.68	0.68	0	409	34453	2
Johnson_10_4_0_perm123	0.10	0.00	0.90	1.23	0	21	677	2
Johnson_16_4_2_perm123	7200.00	0.76	0.67	0.66	0	605	69052	2
Johnson_15_4_3_perm123	1602.39	0.09	0.94	1.28	0	84	11745	3
Johnson_18_4_2_perm123	7200.01	2.19	0.73	0.79	0	837	196256	2
Johnson_19_3_1_perm123	5.13	0.33	0.16	0.03	4	813	5318	4
Johnson_20_3_2_perm123	160.26	0.08	0.85	1.35	0	166	17358	2
Johnson_13_5_4_perm123	7200.00	0.08	0.94	1.46	0	76	14838	3
Johnson_15_3_1_perm123	0.64	0.07	0.18	0.03	6	369	1508	5
Johnson_19_3_0_perm123	0.52	0.51	0.11	0.01	112	749	1773	6
Johnson_15_4_2_perm123	7200.00	0.45	0.63	0.60	0	500	38695	2
Johnson_17_4_3_perm123	7200.00	0.22	0.96	1.27	0	100	21341	3
Johnson_11_4_2_perm123	0.22	0.02	0.37	0.20	0	207	1847	2
Johnson_17_5_4_perm123	7200.04	1.09	0.98	1.29	0	112	60167	3
Johnson_12_5_3_perm123	7200.01	0.14	0.61	0.55	0	311	12828	2
Johnson_13_5_2_perm123	44.34	0.68	0.41	0.20	0	762	20939	4
Johnson_19_4_3_perm123	7200.00	0.49	0.97	1.28	0	111	38968	2
Johnson_13_4_3_perm123	158.79	0.03	0.91	1.22	0	63	4910	2
Johnson_11_4_1_perm123	0.05	0.04	0.37	0.18	5	203	1341	3
Johnson_18_4_2_perm456	7200.03	2.37	0.72	0.79	0	847	197837	3
Johnson_12_4_3_perm456	7200.01	0.02	0.89	1.16	0	56	2902	2
Johnson_12_5_4_perm456	7200.00	0.03	0.93	1.27	0	58	5633	2
Johnson_16_4_3_perm456	1.20	0.14	0.95	1.28	0	92	16144	3
Johnson_12_3_1_perm456	0.26	0.02	0.28	0.09	2	156	299	4
Johnson_14_3_1_perm456	0.46	0.04	0.20	0.05	4	286	922	4
Johnson_14_3_2_perm456	0.20	0.01	0.78	1.04	0	80	2462	3
Johnson_18_4_3_perm456	7200.00	0.36	0.96	1.29	0	111	30829	3
Johnson_11_4_3_perm456	3960.48	0.00	0.86	1.09	0	47	1585	2
Johnson_14_4_2_perm456	7200.00	0.23	0.59	0.52	0	411	20096	2
Johnson_18_3_0_perm456	0.37	0.32	0.37	0.12	12	500	3217	4
Johnson_13_4_0_perm456	0.15	0.09	0.70	0.79	5	213	17159	2
Johnson_14_3_0_perm456	0.05	0.04	0.19	0.03	52	243	775	6
Johnson_19_3_0_perm456	0.50	0.49	0.18	0.02	51	742	1366	6

continued on next page

name	time	pre time	$p(n)/n$	$p(m)/m$	#merge	#del	#add	#rounds
Johnson_13_5_2_perm456	93.31	0.66	0.44	0.23	0	722	18126	3
Johnson_20_3_0_perm456	0.79	0.71	0.34	0.09	11	746	4615	5
Johnson_13_4_3_perm456	188.68	0.02	0.91	1.29	0	66	5797	1
Johnson_13_3_1_perm456	5.05	0.03	0.48	0.24	1	149	387	3
Johnson_13_4_1_perm456	0.27	0.21	0.38	0.18	7	434	4898	3
Johnson_13_5_1_perm456	198.83	0.42	0.61	0.48	5	502	31108	2
Johnson_11_3_1_perm456	0.48	0.01	0.44	0.19	1	91	132	3
Johnson_15_4_2_perm456	7200.00	0.46	0.63	0.60	0	500	38690	2
Johnson_16_6_0_perm456	7200.01	4.34	0.96	1.52	0	320	504897	4
Johnson_16_3_1_perm456	1.08	0.11	0.20	0.05	3	445	2130	3
Johnson_16_4_2_perm456	7200.00	0.87	0.66	0.67	0	610	73548	3
Johnson_20_3_2_perm456	1464.74	0.09	0.86	1.31	0	161	16185	2
Johnson_12_4_1_perm456	0.11	0.09	0.36	0.16	6	312	2980	4
Johnson_14_5_1_perm456	8.09	1.51	0.51	0.38	6	968	92386	3
Johnson_10_4_2_perm456	0.07	0.01	0.33	0.14	0	141	346	2
Johnson_19_4_3_perm456	7200.00	0.49	0.97	1.29	0	112	39262	3
Johnson_20_3_1_perm456	11.39	0.45	0.17	0.04	5	943	7058	4
Johnson_15_3_1_perm456	0.60	0.06	0.19	0.04	5	363	1438	4
Johnson_20_4_3_perm456	7200.00	0.67	0.98	1.28	0	119	50044	2
Johnson_11_4_0_perm456	0.04	0.01	0.83	1.28	3	52	3827	2
Johnson_16_3_2_perm456	14.38	0.03	0.82	1.13	0	98	4682	2
Johnson_13_5_4_perm456	7200.00	0.07	0.95	1.32	0	70	11038	3
Johnson_10_3_1_perm456	0.01	0.00	0.27	0.07	4	84	110	4
Johnson_18_5_4_perm456	7200.08	2.02	0.99	1.31	0	123	93007	3
Johnson_17_3_1_perm456	3.22	0.16	0.21	0.05	3	537	2943	4
Johnson_12_5_2_perm456	30.11	0.22	0.55	0.31	0	353	1518	3
Johnson_14_6_4_perm456	7200.00	1.47	0.80	0.91	0	598	149674	2
Johnson_12_5_1_perm456	106.47	0.12	0.68	0.59	4	253	12197	2
Johnson_11_4_2_perm456	0.34	0.03	0.39	0.22	0	202	1849	3
Johnson_15_6_0_perm456	7200.00	0.90	0.98	1.58	0	123	132474	3
Johnson_16_5_4_perm456	7200.01	0.50	0.98	1.28	0	97	38092	2
Johnson_14_5_3_perm456	7200.01	0.82	0.74	0.79	0	523	79725	2
Johnson_12_5_0_perm456	47.21	0.01	0.95	1.49	0	37	4893	2
Johnson_16_3_0_perm456	0.13	0.12	0.16	0.02	67	401	1088	6
Johnson_16_4_0_perm456	1.51	0.92	0.53	0.38	20	837	75681	4
Johnson_16_4_1_perm456	5.66	1.82	0.39	0.18	0	1107	28996	3
Johnson_15_5_3_perm456	7200.00	1.66	0.78	0.88	0	651	162233	2
Johnson_10_4_1_perm456	0.02	0.01	0.45	0.24	4	112	544	2
Johnson_19_3_1_perm456	5.28	0.33	0.15	0.03	5	814	5324	5
Johnson_18_3_2_perm456	1795.06	0.05	0.83	1.25	0	138	9850	2
Johnson_13_5_3_perm456	7200.00	0.32	0.68	0.68	0	408	34257	2
Johnson_17_5_4_perm456	7200.04	1.21	0.98	1.29	0	111	59092	4
Johnson_12_5_3_perm456	7200.00	0.14	0.61	0.55	0	307	12752	2
Johnson_18_3_1_perm456	2.38	0.24	0.17	0.04	5	676	3662	4
Johnson_16_6_5_perm456	7200.38	2.03	0.99	1.49	0	120	124656	4
Johnson_14_6_1_perm456	7200.00	1.45	0.85	1.00	0	456	147907	2
Johnson_14_5_2_perm456	7200.00	1.86	0.56	0.33	0	880	18108	4
Johnson_15_4_1_perm456	67.87	0.80	0.49	0.25	0	702	10286	3
Johnson_13_4_2_perm456	1572.01	0.10	0.53	0.44	0	334	10402	1
Johnson_17_3_0_perm456	0.20	0.19	0.21	0.03	37	497	1075	4
Johnson_14_4_0_perm456	0.34	0.20	0.64	0.62	6	359	30080	3
Johnson_10_4_0_perm456	0.10	0.00	0.90	1.20	0	21	621	2
Johnson_12_4_2_perm456	22.77	0.05	0.49	0.35	0	251	3922	2
Johnson_14_4_1_perm456	0.52	0.46	0.34	0.14	8	654	11956	5
Johnson_14_6_0_perm456	7200.00	0.22	0.99	1.65	0	38	28388	3
Johnson_17_3_2_perm456	7200.00	0.04	0.84	1.15	0	109	6178	3
Johnson_17_4_0_perm456	2.94	1.82	0.50	0.32	9	1187	105623	2
Johnson_14_5_0_perm456	7200.00	0.32	0.91	1.17	0	186	44634	3
Johnson_14_6_5_perm456	7200.07	0.31	0.97	1.44	0	87	35210	3
Johnson_15_4_3_perm456	2704.15	0.12	0.93	1.27	0	89	11787	4
Johnson_11_4_1_perm456	0.35	0.04	0.57	0.35	0	143	866	4
Johnson_11_3_2_perm456	0.99	0.00	0.72	0.86	0	47	642	3
Johnson_14_5_4_perm456	7200.01	0.14	0.96	1.31	0	76	17078	2
Johnson_15_6_5_perm456	7200.01	0.76	0.98	1.45	0	100	66552	3

continued on next page

name	time	pre time	$p(n)/n$	$p(m)/m$	#merge	#del	#add	#rounds
Johnson_15_4_0_perm456	0.74	0.45	0.58	0.49	8	563	48953	3
Johnson_15_5_0_perm456	7.98	1.05	0.88	1.11	0	357	128772	3
Johnson_15_5_4_perm456	7200.01	0.30	0.97	1.32	0	85	27542	3
Johnson_17_4_3_perm456	7200.00	0.21	0.96	1.28	0	101	21945	3
Johnson_17_4_2_perm456	7200.00	1.42	0.70	0.73	0	723	122130	3
Johnson_13_5_0_perm456	1632.39	0.07	0.93	1.23	0	86	13240	2

Table 9: Analysis of Presolving for Binary-Code-testset.

name	time	pre time	$p(n)/n$	$p(m)/m$	#merge	#del	#add	#rounds
BinaryCode_10_6	0.47	0.47	0.00	0.00	1	1022	0	3
BinaryCode_9_2	439.84	0.02	0.82	1.10	0	90	4516	3
BinaryCode_9_6	0.14	0.14	0.00	0.00	1	510	0	3
BinaryCode_11_3	7200.00	0.46	0.86	1.18	0	286	99713	2
BinaryCode_10_9	0.03	0.03	0.00	0.00	512	511	0	1
BinaryCode_9_7	0.10	0.10	0.00	0.00	1	510	0	3
BinaryCode_10_10	19.76	19.76	0.00	0.00	0	1023	0	1
BinaryCode_10_5	0.45	0.44	0.11	0.01	14	897	3263	6
BinaryCode_9_8	0.01	0.01	0.00	0.00	256	255	0	1
BinaryCode_9_3	11.11	0.05	0.68	0.71	0	165	6781	2
BinaryCode_9_9	2.34	2.34	0.00	0.00	0	511	0	1
BinaryCode_12_3	7200.12	1.26	0.91	1.32	0	364	291163	1
BinaryCode_10_4	0.46	0.26	0.33	0.22	0	682	17546	3
BinaryCode_8_2	3.21	0.00	0.72	0.95	0	72	1747	2
BinaryCode_7_5	0.00	0.00	0.01	0.00	1	126	0	3
BinaryCode_8_4	0.03	0.03	0.01	0.00	11	242	0	4
BinaryCode_10_2	7200.00	0.05	0.89	1.50	0	110	19258	1
BinaryCode_10_8	0.54	0.54	0.00	0.00	1	1022	0	3
BinaryCode_10_3	7200.00	0.13	0.79	1.01	0	220	32827	2
BinaryCode_11_4	13.74	1.01	0.52	0.51	0	981	114511	2
BinaryCode_11_2	7200.01	0.13	0.94	1.59	0	132	47610	1
BinaryCode_12_2	7200.01	0.44	0.96	1.60	0	156	107247	1
BinaryCode_10_7	0.50	0.50	0.00	0.00	1	1022	0	3
BinaryCode_8_5	0.02	0.02	0.00	0.00	1	254	0	3
BinaryCode_13_2	7203.20	1.43	0.98	1.55	0	182	218977	2
BinaryCode_9_5	0.09	0.09	0.01	0.00	11	498	0	4
BinaryCode_9_4	0.08	0.08	0.02	0.00	14	489	1356	6

A.4 Computational Results for CliSAT

Table 10 contains information about the solving performance of CliSAT for each of the selected Johnson instances. It shows the number of nodes generated and the solving time in seconds first on instances where iterative SST-presolving was applied prior to the CliSAT run and next for instances that were solely complemented. The last two columns contain the ratios of nodes and time, where the numbers without SST-presolving are divided by the numbers with SST-presolving.

A.5 Computational Results for Set Covering

Table 11 provides detailed results on the STS set covering instances. The columns give the number of variables (#var), number of rows (#rows), total time in seconds (time), total number of branch-and-bound nodes (#nodes), total number of presolving rounds (#rounds), total number of generators (#gen), total number of variables fixed by the fixation operation (#fixed), total number of rows added in addition operations (#add), time for SST-presolving in seconds (SST time), total number of leaders ($|L|$) and total number of SST-pairs ($|S(L)|$).

Table 10: Comparison of CliSAT performance with and without SST-presolving.

name	with SST-presolving		without SST-presolving		nodes ratio	time ratio
	nodes	time	nodes	time		
Johnson_15_5_4	10436800	3600.1	9797200	3600.1	0.94	1.00
Johnson_14_3_1	453	0.5	363956	4.3	803.43	8.60
Johnson_14_4_0	1	0.5	47	0.5	47.00	1.00
Johnson_14_4_1	162	0.5	1087	0.5	6.71	1.00
Johnson_14_4_3	56576000	3600.0	51291700	3600.0	0.91	1.00
Johnson_12_5_0	1	0.5	11922800	3600.1	Large	Large
Johnson_13_4_3	84417000	3600.0	79088000	3600.0	0.94	1.00
Johnson_14_6_1	285600	3600.1	1480400	3600.1	5.18	1.00
Johnson_19_3_0	1	0.5	1	0.5	1.00	1.00
Johnson_20_4_3	6513700	3600.1	7823300	3600.1	1.20	1.00
Johnson_11_4_0	1	0.5	179	0.5	179.00	1.00
Johnson_15_4_0	1	0.5	27	0.5	27.00	1.00
Johnson_17_4_3	19262900	3600.0	15911200	3600.0	0.83	1.00
Johnson_13_3_1	98	0.5	25972	0.5	265.02	1.00
Johnson_14_6_5	7917300	3600.1	9736000	3600.0	1.23	1.00
Johnson_17_3_2	119486500	3600.0	113778500	3600.0	0.95	1.00
Johnson_15_4_3	40313300	3600.0	37396000	3600.0	0.93	1.00
Johnson_12_5_1	1	0.5	20767200	3600.1	Large	Large
Johnson_16_6_5	1998600	3600.1	2340200	3600.2	1.17	1.00
Johnson_16_5_4	6304300	3600.1	6097300	3600.1	0.97	1.00
Johnson_14_5_2	58920694	2117.1	75657900	3600.0	1.28	1.70
Johnson_13_5_1	154	0.5	14312507	2247.6	Large	Large
Johnson_18_4_3	13977400	3600.1	12211900	3600.1	0.87	1.00
Johnson_18_3_2	109292600	3600.0	98890700	3600.0	0.90	1.00
Johnson_11_4_3	115265515	2623.6	162187300	3600.0	1.41	1.37
Johnson_16_3_1	388	0.5	2759036	62.4	Large	124.80
Johnson_12_4_2	21	0.5	1803397	57.6	Large	115.20
Johnson_17_3_0	1	0.5	1	0.5	1.00	1.00
Johnson_20_3_0	1	0.5	1	0.5	1.00	1.00
Johnson_13_5_2	3290	0.5	1196258	106.2	363.60	212.40
Johnson_13_5_0	1	0.5	2716200	3600.1	Large	Large
Johnson_11_4_1	50	0.5	64	0.5	1.28	1.00
Johnson_11_3_2	193	0.5	2431438	28.6	Large	57.20
Johnson_15_6_5	4699200	3600.1	3655600	3600.0	0.78	1.00
Johnson_14_3_0	1	0.5	1	0.5	1.00	1.00
Johnson_10_4_0	1	0.5	51	0.5	51.00	1.00
Johnson_20_3_2	82258400	3600.0	71574000	3600.0	0.87	1.00
Johnson_10_4_2	37	0.5	2975	0.5	80.41	1.00
Johnson_17_4_0	1	0.5	14	0.5	14.00	1.00
Johnson_12_5_4	62911700	3600.0	59947200	3600.0	0.95	1.00
Johnson_11_4_2	1	0.5	12290	0.5	Large	1.00
Johnson_15_3_1	452	0.5	1770076	26.3	Large	52.60
Johnson_12_4_1	1	0.5	93	0.5	93.00	1.00
Johnson_15_4_1	183	0.5	4766	1.4	26.04	2.80
Johnson_13_4_0	1	0.5	40	0.5	40.00	1.00
Johnson_16_3_2	166534900	3600.0	144259000	3600.0	0.87	1.00
Johnson_19_4_3	9750100	3600.1	8805400	3600.1	0.90	1.00
Johnson_13_5_4	36340300	3600.0	33686100	3600.0	0.93	1.00
Johnson_12_3_1	2	0.5	1178	0.5	589.00	1.00
Johnson_10_4_1	1	0.5	99	0.5	99.00	1.00
Johnson_14_5_4	17648900	3600.0	18491400	3600.1	1.04	1.00
Johnson_17_5_4	3412100	3600.1	3925200	3600.1	1.15	1.00
Johnson_16_4_1	171	0.5	21337	8.2	124.78	16.40
Johnson_12_5_2	1	0.5	4659	0.5	Large	1.00
Johnson_14_5_0	1	0.5	1059900	3600.5	Large	Large
Johnson_18_5_4	2146400	3600.1	2156900	3600.1	1.00	1.00
Johnson_11_3_1	3	0.5	1052	0.5	350.67	1.00
Johnson_12_4_3	115465900	3600.0	106809100	3600.0	0.93	1.00
Johnson_14_5_1	1	0.5	11121400	3600.1	Large	Large

Table 11: Detailed results for STS set covering instances with variant *presolve*.

name	#var	#rows	time	#nodes	#rounds	#gen	#fixed	#add	SST time	L	S(L)
sts51	52	425	25.73	21513	2	6	1	15	0.00	3	80
sts55	55	495	28.54	21043	1	1	1	0	0.04	1	2
sts57	58	532	50.39	47587	2	4	1	17	0.01	3	90
sts61	61	610	58.22	50041	1	1	1	0	0.09	1	2
sts63	64	651	108.40	117461	2	6	1	11	0.01	3	84
sts67	67	737	59.60	39866	1	1	1	0	0.10	1	2
sts69	70	782	192.46	213089	2	6	1	21	0.02	3	110
sts73	73	876	119.53	88297	1	1	1	0	0.31	1	2
sts75	76	925	399.71	443437	2	4	1	19	0.02	3	112
sts79	79	1027	456.02	425025	1	1	1	0	0.07	1	2
sts81	82	1080	872.67	872377	2	4	1	17	0.03	3	114
sts85	85	1190	935.81	814351	1	1	1	0	0.12	1	2
sts87	88	1247	1788.90	1614619	2	4	1	27	0.05	3	140
sts91	91	1365	974.62	728179	1	1	1	0	0.20	1	2
sts93	94	1426	3600.00	3025012	2	6	1	29	0.08	3	150
sts97	97	1552	2417.70	1710467	1	1	1	0	0.12	1	2
sts99	100	1617	3600.00	2779073	2	6	1	19	0.10	3	136
sts103	103	1751	3600.00	2531162	1	1	1	0	0.18	1	2
sts105	106	1820	3600.00	2507582	2	6	1	23	0.10	3	150
sts109	109	1962	3600.00	2379535	1	1	1	0	0.21	1	2
sts111	112	2035	3600.00	2249201	2	4	1	35	0.13	3	180
sts115	115	2185	3600.00	2127851	1	1	1	0	0.27	1	2