Structure-Preserving Symmetry Presolving for Mixed-Binary Linear Problems

Annika Jäger and Marc E. Pfetsch

Department of Mathematics, TU Darmstadt, Germany,
{jaeger|pfetsch}@opt.tu-darmstadt.de

November 12, 2025

Abstract

This paper investigates a presolving method for handling symmetries in mixed-binary programs, based on inequalities computed from so-called Schreier-Sims tables. We show that an iterative application of this method together with merging variables will produce 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. The result is then a changed instance of the same type such that existing black-box solvers can directly be applied. We showcase our approach on stable set problems via computational experiments. For a mixed set of graphs, the presolving method significantly outperforms a black-box branch-and-cut method, while solving the same number of instances as when applying state-of-the-art symmetry handling methods. For Johnson and coding theory graphs, we significantly outperform the stand-alone solver with and without its integrated symmetry handling. A similar speed-up can be observed when running the maximum clique solver CliSAT on the Johnson graphs.

Keywords: Symmetry handling, Schreier-Sims table, Stable set problem

1 Introduction

Symmetry handling for mixed-integer programs is one important component of modern branch-and-cut 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 [19, 20, 21], fundamental domains [7, 28], orbitopes [14, 13, 3], orbital branching [22, 23, 24, 17], Schreier-Sims inequalities [26], to newer methods [9, 8]; see also the overview article [18] and the computational evaluation in [25]. 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 structure, see, e.g., [11, 9], with the exception of particular structures like orbitopes. 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 mixed-binary problems. We build on the inequalities based on Schreier-Sims tables, which have been proposed by Liberti and Ostrowski [16] and Salvagnin [26]. 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 previously used variables; 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. This, 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 problem classes, including packing and covering integer programs as well as stable set and set covering problems, this procedure does not change the problem structure (Section 4). 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 [10]. The resulting stable set problem can then be solved with any suitable solver. As such, one can use black-box solvers on the presolved graphs.

We also use the stable set problem as an example to highlight the computational performance. We test an implementation of the presolving method together with a branch-and-cut solver on a mixed testset of graphs collected from many sources, on Johnson graphs, and graphs arising from coding theory (Section 5). 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 a bit worse than the one of the solver applying state-of-the-art symmetry handling, as implemented in SCIP [4]. For Johnson and coding theory graphs, the presolving method is vastly superior by factors around 2 to 6. We also observe a strong increase of performance of the maximum clique solver CliSAT [27] when applied on the presolved and complemented Johnson graphs.

2 Preliminaries

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

$$\max \left\{ c^\top x \,:\, Ax \leq b,\ x_i \in \{0,1\} \text{ for } i \in I,\ x \in [0,1]^n \right\}, \tag{P}$$

where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$ and $I \subseteq [n] := \{1, \dots, n\}$. The kth row of A is denoted by a_k.

We consider the formulation symmetry of the problem (P) as defined in [25], which will simply refer to as *symmetry* in the following. The symmetry group Γ of (P) is a subgroup of the full symmetric group $\mathcal{S}_n$ of $[n]$ acting on $x \in \mathbb{R}^n$ as $\gamma(x) := (x_{\gamma^{-1}(1)}, \dots, x_{\gamma^{-1}(n)})$. The group Γ consists of all permutations γ for which there exists a permutation $\delta \in \mathcal{S}_m$ such that the objective and right hand side are invariant, i.e., $\gamma(c) = c$, $\delta(b) = b$ 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

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

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

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

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., [19], that stores a sequence of stabilized indices in L:

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

(A1) choose $\ell \in [n] \setminus L$ and compute $O_\ell = \mathrm{orb}(\ell, \Gamma')$;

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

(A3) repeat steps A1 and A2 until $\Gamma' = \{\mathrm{id}_{[n]}\}$.

The SHIs obtained from this algorithm are the so-called *SST-cuts* $x_\ell \geq x_f$ for all $\ell \in L$ and all $f \in O_\ell \setminus \{\ell\}$, where ℓ is called *leader* and f *follower* of ℓ. Liberti and Ostrowski [16] and Salvagnin [26] 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, define $S(L) \coloneqq \{(\ell, f) : \ell \in L,\ f \in O_\ell \setminus \{\ell\}\}$ to be the set of all SST-pairs, which we just denote by S if the corresponding L is clear. As discussed in [10], the set of SST-pairs can be computed in polynomial time.

Salvagnin [26] 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 without symmetry is to iteratively compute symmetry and add SST-cuts. This approach does not seem to have been investigated so far. We show 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 ith column of A, from $A_{\star,i}$, to $A_{\star,i} + A_{\star,j}$.

Lemma 2.1. *Consider a MIP on variables $x_1, \dots, x_n$ and iteratively compute the formulation 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. This procedure terminates after finitely many iterations with a problem with trivial symmetry group.*

Proof. Assume this is false. There are only finitely many different SST-cuts that could possibly be added and variables that can be merged. Thus, 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. More precisely, there exists a symmetry $\gamma \in \Gamma$ with $\gamma(i) \neq i$ for some $i \in [n]$ and all inequalities $x_i \geq x_{\gamma^k(j)}$ with $\gamma^k(i) \neq i$ are already part of the problem formulation. Since each orbit is cyclic under a single element, there exists some $p \in \mathbb{N}$ such that $\gamma^p(i) = i$, which also means that $\gamma^{p-1}(i) \neq i$ because $\gamma(i) \neq i$. Thus, $x_i \geq x_{\gamma^{p-1}(j)}$ is part of the problem formulation. Since γ is a symmetry, the constraint $x_{\gamma(i)} \geq x_i$ obtained by applying γ is also part of the problem formulation. But this contradicts that we computed the SST-pair $(i, \gamma(i))$ and no merging is applicable. □

Therefore, iteratively adding SST-cuts and merging is a presolving method that removes all non-trivial symmetry from a MIP. This works especially for problem classes that allow the addition of such constraints and merging without structural change like the SAT problem (where $x_i \geq x_j$ can be expressed as $\overline{x}_j \vee x_i$). For SAT, the use of SST clauses and the addition of binary symmetry breaking clauses have been investigated in [6] and [1], respectively, where they were not added iteratively but as part of an extensive procedure to reduce symmetry.

3 Presolving for Mixed-Binary Linear Problems

The central step in the presolving routine for (P) is adding constraints that enforce the implication of the SST-pair. In particular, for an SST-pair (i, j) over binary variables, we want to add constraints such that $x_j = 1$ implies $x_i = 1$. To achieve this goal, we add constraints that cause all constraints containing x_i (resp. x_j) to be fulfilled independently of the value of x_i (resp. x_j) if x_j (resp. x_i) is fixed to 0 (resp. 1). Then to obtain optimality, the variable has to be 0 or 1 depending on the signs of c_i and c_j. Throughout this section, we want to argue that certain modifications of problem (P) cannot cut off all optimal solutions. We then say that the operation is *valid* for (P). We further say that some

constraint holds for (P) if it is satisfied by all feasible solutions. For $N \coloneqq [n] \setminus \{i,j\}$ and $x \in [0,1]^n$, we use the notation

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

Definition 3.1. 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

$$\langle a_k, x \rangle_N + (a_{k,i} + a_{k,j}) x_j \leq b_k. \qquad (\text{AO}^+(k,i,j))$$

- 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

$$\langle a_k, x \rangle_N + (a_{k,i} + a_{k,j}) x_i \leq b_k. \qquad (\text{AO}^-(k,i,j))$$

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 have to check that the constraints added by the addition operation are satisfied by an optimal solution $x^\star$ of (P) that satisfies the SST-cut $x^\star_i \geq x^\star_j$ associated to the SST-pair $(i,j) \in S$. This implies

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

Hence, the validity of the constraint $a_k x \leq b_k$ guarantees the validity of the constraints ($\text{AO}^+(k,i,j)$) and ($\text{AO}^-(k,i,j)$) added by addition operations. □

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. Next, we want to show that the operation indeed yields the desired implication.

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). If $c_i > 0$, for every optimal solution $x^\star$, one has $x^\star_i \geq x^\star_j$. In particular, if the variables are binary, $x^\star_i = 0$ implies $x^\star_j = 0$.*

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

$$a_k \tilde{x} = \langle a_k, x^\star \rangle_N + (a_{k,i} + a_{k,j}) x^\star_j \leq b_k,$$

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

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). If $c_j < 0$, for every optimal solution $x^\star$, one has $x^\star_i \geq x^\star_j$. In particular, if the variables are binary, $x^\star_i = 0$ implies $x^\star_j = 0$.*

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

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

Definition 3.5. Let x_i and x_j be variables in (P). Further, let either $c \geq 0$ 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 $c \leq 0$ 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 $c \geq 0$ and, for all $k \in [m]$, let the constraints $(\mathrm{AO}^+(k,i,j))$ and $(\mathrm{AO}^+(k,j,i))$ with $a_{k,i} > 0$ and $a_{k,j} > 0$, respectively, hold in (P). Then if c_i, $c_j > 0$, by Lemma 3.3 all optimal solutions $x^\star$ satisfy $x_i^\star \geq x_j^\star$ as well as $x_i^\star \leq x_j^\star$, implying that $x_i^\star = x_j^\star$. We claim that if $c_i = 0$ or $c_j = 0$, there still exists at least one optimal solution $x^\star$ 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 $\tilde{x}_i = \tilde{x}_j$ and $\tilde{x} \geq x^\star$. 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 $c \leq 0$, we can argue analogously and generally get $x_i^\star = x_j^\star$ for at least one optimal solution $x^\star$ 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. □

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

$$a_k x = \langle a_k, x\rangle_N + a_{k,i}x_i + a_{k,j}x_i = \langle a_k, x\rangle_N + (a_{k,i} + a_{k,j})x_i \leq b_k,$$

which agrees with the constraint $(\mathrm{AO}^+(k,j,i))$ if $a_{k,j} > 0$, $(\mathrm{AO}^-(k,i,j))$ 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 $(\mathrm{AO}^+(k,i,j))$ and $(\mathrm{AO}^-(k,i,j))$ exist in (P), respectively.

We claim that for a row a_k with $a_{k,j} > 0$, $(\mathrm{AO}^+(k,j,i))$, i.e., $\langle a_k, x\rangle_N + (a_{k,i} + a_{k,j})x_i \leq b_k$, holds for all feasible x of (P). Since constraint $a_k x = \langle a_k, x\rangle_N + a_{k,i}x_i + a_{k,j}x_j \leq b_k$ is in (P), there exits some row $\ell \in [m]$ such that $a_\ell x = \langle a_k, \gamma(x)\rangle_N + a_{k,i}x_{\gamma^{-1}(i)} + a_{k,j}x_i \leq b_\ell = b_k$. Since $a_{k,j} > 0$ and $(i, \gamma^{-1}(i)) \in S$, we have by assumption that the constraint $\langle a_k, \gamma(x)\rangle_N + (a_{k,i} + a_{k,j})x_{\gamma^{-1}(i)} \leq b_k$ is in the formulation of (P), because it corresponds to $(\mathrm{AO}^+(\ell, i, \gamma^{-1}(i)))$. Since γ^{-1} is a symmetry, we obtain that the constraint $\langle a_k, x\rangle_N + (a_{k,i} + a_{k,j})x_i \leq b_k$ is in (P) which is the desired constraint.

We can analogously show that the constraints of the type $(\mathrm{AO}^-(k,j,i))$ 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. □

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.

It is possible that the addition operation adds constraints that enforce the fixing of a certain binary 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.

For $i \in [n]$ we define $S^i \coloneqq \{p \in [n] : (i,p) \in S\}$, the set of followers of i in S and $S_i \coloneqq \{p \in [n] : (p,i) \in S\}$, the set of leaders having follower i and denote by $\overline{S}_i$ its complement $[n] \setminus S_i$. Further, we define $\mathcal{A}_k^i \coloneqq \{p \in [n] \setminus \{i\} : a_{k,p} < 0\}$ and denote by $a_k(I)$ the sum $\sum_{i \in I} a_{k,i}$.

Definition 3.9. Let a_k be a row in A such that

$$a_k(S_j) + a_{k,j} + a_k(\overline{S}_j \cap \mathcal{A}_k^j) > b_k. \qquad (\mathrm{DO}(k,j))$$

Then, if x_j is a binary variable, we fix it to 0 and call it **deletion operation**.
Let a_k be a row in A such that

$$a_k(\overline{S^i} \cap \mathcal{A}_k^i) > b_k. \qquad (\mathrm{FO}(k,i))$$

Then, if x_i is a binary variable, we fix it to 1 and call it **fixation operation**.

Lemma 3.10. *The deletion operation is a valid presolving step for* (P).

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

$$a_k\tilde{x} = a_k(S_j) + a_{k,j} + \sum_{p \in \overline{S}_j \setminus \{j\}} a_{k,p}\,\tilde{x}_p \geq a_k(S_j) + a_{k,j} + a_k(\overline{S}_j \cap \mathcal{A}_k^j) > b_k$$

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

Lemma 3.11. *The fixation operation is a valid presolving step for* (P).

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

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

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

We propose the following presolving routine:

Definition 3.12 (Presolving routine)**.**

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.

Proposition 3.13. *After finitely many applications of the presolving routine of Definition 3.12, 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 the presolving routine in which the problem is modified. Clearly, there can only be finitely many presolving rounds in which merge, deletion or fixation presolving is active, since all of them reduce the number of 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.

By definition of addition presolving, all new constraints use a basis constraint from the initial problem and update coefficients by sums of the initial coefficients. Thus, there are at most n former coefficients that get distributed among the n variables to generate new coefficients for the derived added constraints. Consequently, there are at most $m \cdot n^n - m$ constraints that can possibly be added by addition presolving. □

Observe that added constraints can maximally have as many nonzero entries as the constraint they were derived from and two constraints with the same right hand sides and corresponding rows of A that are permutations of each other lead to the same set of potentially added constraints. Therefore, we can make the upper bound on the presolving rounds until all symmetry is handled indicated in the proof of Proposition 3.13 more precise, i.e., it is $\tilde{m} \cdot n^{\tilde{n}} + n - m$ where $\tilde{n}$ is the maximum number of nonzero entries in a row of A and $\tilde{m}$ is the maximum number of constraints in the problem formulation such that from no two of them the same constraints can potentially be derived.

Anyhow, the upper bound is still very large and the hope is that it is not tight. In fact, in the computational results for the stable set problem presented in Section 4.2, only few rounds of the presolving routine are needed to handle the symmetry completely.

4 Structure Preserving Presolving

In this section, we discuss examples where the general presolving method of Section 3 leaves the particular structure invariant.

4.1 Packing and Covering Integer Programs

Let $I = [n]$. A *Packing Integer Program* arises from Problem (P) 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, the presolving routine keeps the structure invariant, i.e., the coefficients stay nonnegative and nonpositive, respectively. Then, for instance, approximation algorithms can be applied, see, e.g., [15].

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

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

A special feature of the stable set problems 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 [10], but without finiteness proof.

Lemma 4.1. *If the presolving routine in Definition 3.12 is applied on* (MSSP)*, the presolving steps can be translated for the stable set setting in the following way:*

- Merge: *Let u, $v \in V$ such that $N(u) = N(v)$. Fix $x_v = 0$, delete all edges incident to v from E and update c_u to $c_u + c_v$.*
- Deletion: *Let $\{u, v\}$ 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 $a_k(S_v) + a_{k,v} > 1$. So 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 $(\mathrm{AO}^+(k, u, v))$ 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 $(\mathrm{AO}^+(k, u, v))$ just replaces x_u by x_v, i.e., it corresponds to the edge $\{u, \ell\}$. 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 $(\mathrm{AO}^+(k, u, v))$ with $a_{k,u} > 0$ and $(\mathrm{AO}^+(k, v, u))$ with $a_{k,v} > 0$ hold in (P) for all $k \in [m]$. First assume that $\{u, v\} \in E$. Then the constraint $(\mathrm{AO}^+(k, u, v))$ is $2x_v \leq 1$. This constraint cannot be part of the initial formulation of (MSSP), since G is assumed to be a simple graph, or implied by edge constraints or 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 $(\mathrm{AO}^+(k, u, v))$ with $a_{k,u} > 0$ correspond to edges between v and neighbors of u (see translation of the addition operation). More precisely, the merge condition that all constraints $(\mathrm{AO}^+(k, u, v))$ 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 $\{v, w\}$. □

The upper bound on the number of presolving rounds of Section 3 corresponds to $n^2 + n - m$ in the special case of the stable set problem. No constraint with coefficient larger than 1 can be added by an addition operation, because of the prior deletion operation. Therefore, we can even reduce the upper bound to $n \cdot (n-1) + n - m$, i.e., no two coefficients that are nonzero can be assigned to the same variable. This makes sense, since at most $n \cdot (n-1)$ non-parallel edges can be added to a simple graph with n nodes. The next section 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 [11] arising from the product of a graph with a complete graph with k nodes. Here, the presolving routine 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 set of nodes and let P be a family of sets $\sigma \subseteq U$. The weighted set cover problem with set weights $c \in \mathbb{R}^P_+$ is defined by

$$\min \left\{ c^\top x \,:\, \sum_{u \in \sigma \in P} x_\sigma \geq 1 \quad \forall\, u \in U,\ x \in \{0,1\}^P \right\}. \tag{SSCP}$$

The presolving steps of Definition 3.12 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 node $\tilde{u}$ to U and add it to σ as well as all sets in $P \setminus \{\tau\}$ that contain u.
- *Merge:* Let σ, $\tau \in P$ such that for all $u \in \sigma$ with $u \notin \tau$ there exists $u' \in \tau$ such that $\{\rho \in P \,:\, u \in \rho\} \setminus \{\sigma\} = \{\rho \in S \,:\, u' \in \rho\} \setminus \{\tau\}$ and the other way around. Then fix $x_\tau = 0$, delete all nodes $u \in \tau$ from U and update c_σ to $c_\sigma + c_\tau$.
- *Deletion:* Not applicable.
- *Fixation:* Let $u \in U$ such that $u \in \sigma$ and $(\sigma, \tau) \in S$ for all τ with $u \in \tau$, $\tau \neq \sigma$. Then the variable x_v is fixed to 0.

As in the case of the stable set problem, the addition operation can possibly add constraints with coefficients larger than 1. But since all variables are binary in (SSCP), the constraints $2x_\sigma + \sum x_\tau \geq 1$ and $x_\sigma + \sum x_\tau \geq 1$ are equivalent. Therefore it is equally valid to add the second constraint and the problem structure remains intact after applying the presolving routine.

5 Computational Results

We demonstrate the effect of our presolving method on the example of the stable set problem. To this end, we use the branch-and-cut solver BACS (the source code is available on github), because it allows to apply the state-of-the-art symmetry handling methods of the framework SCIP [4]. 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.

Table 1: Average indicators of the presolving routine on different testsets.

Testset	#merge	#del	#add	time	$\|L\|$	$\|S(L)\|$
Mixed	1032.8	160.8	18 766.1	4.78	31.4	576.5
Johnson	2.5	270.0	38 428.5	0.32	7.8	2973.2
Binary-code	2.5	441.7	42 022.4	0.34	30.5	3249.1

Our presolving routine (Definition 3.12, Lemma 4.1) is performed on the original graph before the optimization process starts. After the symmetry is computed in the second step of the presolving routine, BACS first checks whether there are symmetric components and possibly merges them. Since nodes that get merged during this operation do not meet the merge condition (otherwise they would have been merged by a merge operation in the first step of the presolving routine), we do not count them towards the merge operation even if the action applied on them is the same. The presolving routine is run iteratively until no further modifications of the graph are made, which means that no symmetry remains by Proposition 3.13. 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.

To detect automorphisms of graphs, we apply bliss [12] and sassy [2]. 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 [10], which roughly means we prioritize choosing leaders from maximum orbits as long as they fulfill a structure-preserving condition that is described in the following definition:

Definition 5.1. 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

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

for all $u \in O_i$ where $N(u)$ is the neighborhood of u in G.

The computational experiments were carried out on three different testsets: the *mixed-testset*, which is a collection of 73 instances from various sources, the *Johnson-testset*, which includes 62 Johnson graphs of adequate complexity, and the *binary-code-testset*, containing 23 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 the presolving routine has on the instances of the three different testsets. More specifically, the arithmetic mean of the number of variables deleted or edges added by the presolving routine over all instances in each testset is displayed. The same information is given for the individual instances in Tables 3, 4 and 5 in the appendix. Further, the average of the presolving time as well as the total number of leaders $|L|$ and followers $|S(L)|$ computed is listed in the table.

Each operation of the presolving routine becomes active, while the presolving time is small compared to the overall solving time (see Table 2). While for the mixed-testset the largest number of leaders is used on average, the number of followers and therefore the number of SST-pairs strongly increases from the mixed- to the Johnson-testset and then again a bit to the binary-code-testset. Comparable to the increase of SST-pairs is also the increase of deletion and addition operations. This effect cannot be observed for the merge operation due to two outliers in the mixed-testset. The maximal number of iterations of the presolving routine is 6, meeting the expectation that the upper bound of $O(n^2)$ is far from being tight (see Tables 3, 4 and 5).

Table 2 summarizes the overall solution performance after the presolving routine was iteratively applied until no symmetry was left (setting *presol*) and compares this to default settings with SCIP

Table 2: Comparison of presolving performance on different testsets.

	Mixed (73)			Johnson (62)			Binary-code (23)		
Setting	#opt	time	#nodes	#opt	time	#nodes	#opt	time	#nodes
base	42	334.74	3339.2	28	630.08	3232.3	10	689.46	2058.8
base-sym	47	136.95	1475.6	41	131.16	661.7	15	181.08	665.6
presol	47	201.82	2392.5	41	68.16	660.8	16	23.74	282.1

symmetry handling (setting *base-sym*) and without SCIP symmetry handling (setting *base*). For all three testsets, the number of instances solved to optimality within one hour, the shifted geometric means over all instances of the solution time and the number of nodes in the branch-and-bound tree are listed. For detailed information on the solution times see again Tables 3, 4 and 5. For the mixed-testset, the presol setting solves five more instances than the base setting with a solving time reduction of around $40\,\%$. However, base-sym dominates presol in solving the same number of instances while taking about $32\,\%$ less time. For the other two testsets, however, the performance of presol is clearly superior to both base and base-sym with around 48–$87\,\%$ reduction of solving time.

5.1 Computational Results for CliSAT

CliSAT [27, 5] 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 it contains weighted graphs and CliSAT does not allow weighted instances. Therefore, we exemplarily only performed computations on the Johnson-testset. That CliSAT does not support weighted instances is also an issue if the presolving applies merge operations, since they modify node weights. To avoid this, we ran the presolving routine without the merge operation. The results of Table 4 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 presolving and complementing is not included in the results.

From the results listed in Table 6 in the appendix, we conclude that after presolving, CliSAT can solve 7 more instances in around one third of the time compared to CliSAT without 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. Moreover, note that CliSAT cannot solve weighted instances, such that we did not perform merging and only ran tests on one testset.

References

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

[2] M. Anders, P. Schweitzer, and J. Stieß, "Engineering a preprocessor for symmetry detection," in *21st International Symposium on Experimental Algorithms (SEA 2023)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), L. Georgiadis, Ed., vol. 265. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023, pp. 1:1–1:21. 10.4230/LIPIcs.SEA.2023.1

[3] P. Bendotti, P. Fouilhoux, and C. Rottner, "Orbitopal fixing for the full (sub-)orbitope and application to the unit commitment problem," *Mathematical Programming*, vol. 186, pp. 337–372, 2021. 10.1007/s10107-019-01457-1

[4] K. Bestuzheva, M. Besançon, W.-K. Chen, A. Chmiela, T. Donkiewicz, J. van Doornmalen, L. Eifler, O. Gaul, G. Gamrath, A. Gleixner, L. Gottwald, C. Graczyk, K. Halbig, A. Hoen, C. Hojny, R. van der Hulst, T. Koch, M. Lübbecke, S. J. Maher, F. Matter, E. Mühmer, B. Müller, M. E. Pfetsch, D. Rehfeldt, S. Schlein, F. Schlösser, F. Serrano, Y. Shinano, B. Sofranac, M. Turner, S. Vigerske, F. Wegscheider, P. Wellner, D. Weninger, and J. Witzig, "Enabling research through the SCIP Optimization Suite 8.0," *ACM Trans. Math. Softw.*, vol. 49, no. 2, 2023, article 22. 10.1145/3585516

[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] J. Devriendt, B. Bogaerts, M. Bruynooghe, M. Denecker, N. Creignou, and D. LeBerre, "Improved static symmetry breaking for sat," in *Proceedings of Theory and Applications of Satisfiability Testing – SAT 2016 – 19th International Conference, Bordeaux, France*, ser. LNCS, vol. 9710. Springer, 2016, pp. 104–122.

[7] E. J. Friedman, "Fundamental domains for integer programs with symmetries," in *Combinatorial Optimization and Applications*, ser. LNCS, A. Dress, Y. Xu, and B. Zhu, Eds. Springer, 2007, vol. 4616, pp. 146–153. 10.1007/978-3-540-73556-4_17

[8] C. Hojny, "Packing, partitioning, and covering symresacks," *Discrete Applied Mathematics*, vol. 283, pp. 689–717, 2020. 10.1016/j.dam.2020.03.002

[9] C. Hojny and M. E. Pfetsch, "Polytopes associated with symmetry handling," *Mathematical Programming*, vol. 175, pp. 197–240, 2019. 10.1007/s10107-018-1239-7

[10] C. Hojny, M. E. Pfetsch, and J. Verschae, "The impact of symmetry handling for the stable set problem via Schreier-Sims cuts," arXiv, Tech. Rep. 2311.06085, 2023. [Online]. Available: https://arxiv.org/abs/2311.06085

[11] T. Januschowski and M. E. Pfetsch, "Branch-cut-and-propagate for the maximum k-colorable subgraph problem with symmetry," in *Proc. 8th International Conference, CPAIOR 2011, Berlin*, ser. Lecture Notes in Computer Science, T. Achterberg and J. C. Beck, Eds., vol. 6697. Springer, 2011, pp. 99–116. 10.1007/978-3-642-21311-3_11

[12] T. Junttila and P. Kaski, "bliss: A tool for computing automorphism groups and canonical labelings of graphs," https://users.aalto.fi/~tjunttil/bliss/, 2012.

[13] V. Kaibel, M. Peinhardt, and M. E. Pfetsch, "Orbitopal fixing," *Discrete Optimization*, vol. 8, no. 4, pp. 595–610, 2011. 10.1016/j.disopt.2011.07.001

[14] V. Kaibel and M. E. Pfetsch, "Packing and partitioning orbitopes," *Mathematical Programming*, vol. 114, no. 1, pp. 1–36, 2008. 10.1007/s10107-006-0081-5

[15] S. G. Kolliopoulos and N. E. Young, "Approximation algorithms for covering/packing integer programs," *Journal of Computer and System Sciences*, vol. 71, no. 4, pp. 495–505, 2005. 10.1016/j.jcss.2005.05.002

[16] L. Liberti and J. Ostrowski, "Stabilizer-based symmetry breaking constraints for mathematical programs," *Journal of Global Optimization*, vol. 60, pp. 183–194, 2014. 10.1007/s10898-013-0106-6

[17] J. Linderoth, J. Núñez Ares, J. Ostrowski, F. Rossi, and S. Smriglio, "Orbital conflict: Cutting planes for symmetric integer programs," *INFORMS Journal on Optimization*, vol. 3, no. 2, pp. 139–153, 2021. 10.1287/ijoo.2019.0044

[18] F. Margot, "Symmetry in integer linear programming," in *50 Years of Integer Programming*, M. Jünger, T. M. Liebling, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi, and L. A. Wolsey, Eds. Springer, 2010, pp. 647–686.

[19] ——, "Pruning by isomorphism in branch-and-cut," *Mathematical Programming*, vol. 94, no. 1, pp. 71–90, 2002. 10.1007/s10107-002-0358-2

[20] ——, "Exploiting orbits in symmetric ILP," *Mathematical Programming*, vol. 98, no. 1–3, pp. 3–21, 2003. 10.1007/s10107-003-0394-6

[21] ——, "Small covering designs by branch-and-cut," *Mathematical Programming*, vol. 94, no. 2, pp. 207–220, 2003. 10.1007/s10107-002-0316-z

[22] J. Ostrowski, "Symmetry in integer programming," Ph.D. dissertation, Lehigh University, 2008.

[23] J. Ostrowski, M. F. Anjos, and A. Vannelli, "Modified orbital branching for structured symmetry with an application to unit commitment," *Mathematical Programming*, vol. 150, no. 1, pp. 99–129, 2015. 10.1007/s10107-014-0812-y

[24] J. Ostrowski, J. Linderoth, F. Rossi, and S. Smriglio, "Orbital branching," *Mathematical Programming*, vol. 126, no. 1, pp. 147–178, 2011. 10.1007/s10107-009-0273-x

[25] M. E. Pfetsch and T. Rehn, "A computational comparison of symmetry handling methods for mixed integer programs," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 37–93, 2019. 10.1007/s12532-018-0140-y

[26] D. Salvagnin, "Symmetry breaking inequalities from the Schreier-Sims table," in *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, W.-J. van Hoeve, Ed. Cham: Springer, 2018, pp. 521–529. 10.1007/978-3-319-93031-2_37

[27] P. San Segundo, F. Furini, D. Álvarez, and P. M. Pardalos, "CliSAT: A new exact algorithm for hard maximum clique problems," *European Journal of Operational Research*, vol. 307, no. 3, pp. 1008–1025, 2023. 10.1016/j.ejor.2022.10.028

[28] J. Verschae, M. Villagra, and L. von Niederhäusern, "On the geometry of symmetry breaking inequalities," *Mathematical Programming*, vol. 197, no. 2, pp. 693–719, 2023. 10.1007/s10107-022-01819-2

A Further Information on Computational Results

We use a developer version of SCIP 9.2.4 (githash: 919eba6) and CPLEX 12.10 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 and with a time limit of one hour.

A.1 Instances

Table 1 of [10] 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 computations. Then we ran BACS in default-mode with disabled symmetry handling of SCIP 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 the root node or in less than 10 seconds. All remaining 73 instances form the *mixed-testset*.

The *Johnson-testset* consists of 62 Johnson graphs. These graphs were selected by first generating all Johnson graphs with 100 to $10\,000$ nodes and 1000 to $1\,000\,000$ edges. Then again, we ran BACS in default-mode without SCIP symmetry handling. We removed all instances that were too easy like we did for the mixed-testset and additionally removed all instances that appeared too hard. This means all instances were removed for which the gap between primal and dual bound was still larger than $200\,\%$ after one hour of computation.

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 in the exact same way like we did for the Johnson-testset and this way obtained 23 instances.

A.2 Detailed Results

In Tables 3, 4 and 5 results of the computations described in Section 5 for each instance in the three different testsets are displayed. The solution time and the time used for 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 $p(n)/n$ contain the ratios of nodes and edges of the initial graph the presolved graph consists of. The columns #merge, #del and #add summarize the quantities of nodes fixed or edges added by each presolving method. Further, the column #rounds displays the number of presolving rounds applied before the procedure terminated.

A.3 Computational Results for CliSAT

Table 6 contains information about the solving performance of CliSAT for each instance in the Johnson-testset. It shows the number of nodes generated and the solving time in seconds first on instances where presolving was applied prior to the CliSAT run and next for instances that were solely complemented. The last two lines contain the ratios of nodes and time where the numbers without presolving are divided by the numbers with presolving.

Table 3: Analysis of presolving for mixed-testset.

name	time	pre time	$p(n)/n$	$p(m)/m$	#merge	#del	#add	#rounds
vc-exact_038.gr-compl	8.07	0.01	0.97	0.99	4	19	0	2
ref-25-35-00-compl-weigh	0.48	0.01	0.99	0.99	0	2	0	1
ref-25-30-01-compl-weigh	0.57	0.01	0.99	0.98	0	3	0	1
a265032_2dc.2048	3599.61	0.42	1.00	1.00	0	1	220	1
a265032_1zc.2048	3599.94	0.06	0.97	1.21	0	61	10 479	1
a265032_1zc.4096	3599.93	0.19	0.98	1.31	0	66	31 217	1
a265032_1tc.1024	17.75	0.01	0.62	0.65	1	1	27	2
a265032_1tc.2048	3599.97	0.03	0.50	0.50	1	0	0	1
a265032_1et.2048	3599.92	0.08	0.49	0.52	3	15	628	6
a265032_1zc.512	110.50	0.01	0.92	1.11	0	41	1783	1
a265032_2dc.512	19.81	0.04	1.00	1.00	0	1	144	1
a265032_1zc.1024	3599.97	0.03	0.96	1.25	0	45	5506	2
a265032_1dc.512	4.31	0.01	1.00	1.00	0	1	32	1
a265032_1dc.1024	753.86	0.01	1.00	1.00	0	1	36	1
a265032_1et.1024	23.34	0.04	0.61	0.72	3	15	828	4
a265032_2dc.1024	2975.23	0.12	1.00	1.00	0	1	180	1
a265032_1dc.2048	3599.97	0.04	1.00	1.00	0	1	40	1
relational_2	63.00	62.99	0.00	0.03	24 749	0	1225	1
relational_3	0.26	0.16	0.01	0.01	0	0	0	1
Alchemy_11	20.17	0.02	0.98	1.00	39	0	0	1
relational_5	2.45	2.45	0.00	0.00	49 999	0	0	1
relational_4	62.57	61.93	0.01	0.21	99	0	9900	1
117-compl-weigh	3.05	1.72	0.96	0.90	0	184	0	1
112-compl-weigh	11.40	6.02	0.96	0.89	0	347	0	1
MANN_a45comp	231.22	0.01	1.00	1.36	0	1	709	1
MANN_a27comp	10.97	0.00	0.99	1.81	0	4	578	2
keller4comp	1.51	0.00	0.88	0.95	0	21	966	2
keller5comp	2915.79	0.07	0.94	1.09	0	43	14 862	2
keller6comp	3597.82	2.19	0.97	1.09	0	116	165 635	3
hamming10-4comp	3599.85	0.15	0.79	1.01	0	220	32 827	2
MANN_a81comp	3599.91	0.09	1.00	1.94	0	6	6135	2
monoton-9-compl	3599.96	0.04	1.00	1.01	0	3	486	1
monoton-8-compl	3599.98	0.02	0.99	1.01	0	3	418	1
monoton-7-compl	51.82	0.01	0.99	1.02	0	3	350	1
12-20-6-6-compl-weigh	3599.46	0.54	1.00	0.99	0	7	0	1
01-11-4-4-compl-weigh	6.00	0.00	0.88	1.03	1	17	464	2
04-14-6-6-compl-weigh	235.27	0.19	0.88	0.95	0	94	21 488	2
15-22-10-10-compl-weigh	3553.55	46.60	0.98	0.97	0	144	19 355	2
13-20-8-10-compl-weigh	1177.08	2.58	0.99	0.98	0	21	722	1
03-14-4-7-compl-weigh	301.35	0.00	0.99	1.05	0	2	306	1
08-17-6-6-compl-weigh	217.51	0.06	0.87	0.86	0	70	6331	2
11-20-6-5-compl-weigh	3599.52	0.53	0.96	0.93	0	55	7654	1
14-21-10-9-compl-weigh	413.93	41.06	0.29	0.10	0	3605	133 446	3
02-12-4-6-compl-weigh	4.74	0.00	0.93	1.21	0	17	1379	2
10-19-8-8-compl-weigh	3598.04	1.97	0.94	0.89	0	130	23 074	2
09-19-4-6-compl-weigh	17.60	0.00	0.96	1.06	0	10	672	1
06-16-8-8-compl-weigh	880.43	3.28	0.64	0.41	0	802	15 792	2
1-Insertions_5	10.67	0.00	0.99	1.01	0	2	50	2
12-20-6-6	3599.68	0.33	1.00	1.01	0	0	2158	1
02-12-4-6	0.85	0.02	0.61	0.38	8	81	209	2
15-22-10-10	3596.44	3.57	1.00	1.02	0	27	81 926	1
04-14-6-6	146.55	0.22	0.84	0.73	0	132	9802	2
14-21-10-9	3594.66	5.34	0.76	0.68	347	885	379 031	2
06-16-8-8	3599.49	0.51	0.91	1.17	112	95	102 034	1
11-20-6-5	3599.65	0.35	0.98	1.01	0	24	14 393	1
08-17-6-6	31.72	0.09	0.89	0.86	2	57	5431	3
03-14-4-7	64.21	0.02	0.96	0.92	0	9	45	1
10-19-8-8	3599.32	0.68	0.97	1.01	0	62	37 450	1
13-20-8-10	1075.63	0.56	1.00	1.01	0	2	6218	1
15-22-10-10.cmpl	3554.60	48.13	0.98	0.97	0	144	19 355	2
13-20-8-10.cmpl	323.32	2.15	0.99	0.98	0	21	722	1
09-19-4-6.cmpl	11.23	0.01	0.96	1.06	0	10	672	1
11-20-6-5.cmpl	3599.46	0.54	0.96	0.93	0	55	7654	1
02-12-4-6.cmpl	6.55	0.00	0.93	1.21	0	17	1379	2
12-20-6-6.cmpl	3599.49	0.55	1.00	0.99	0	7	0	1
10-19-8-8.cmpl	3597.78	2.22	0.94	0.89	0	130	23 074	2
08-17-6-6.cmpl	178.74	0.06	0.87	0.86	0	70	6331	2
03-14-4-7.cmpl	83.20	0.00	0.99	1.05	0	2	306	1
14-21-10-9.cmpl	95.22	47.15	0.29	0.09	0	3619	134 432	4
04-14-6-6.cmpl	297.97	0.15	0.89	0.97	0	85	21 360	2
ehi-85-297-60-compl	11.29	0.06	0.99	0.99	12	0	0	1
ehi-90-315-40-compl	14.93	0.05	1.00	1.00	6	0	0	1
ehi-90-315-48-compl	17.29	0.07	1.00	1.00	6	0	0	1

Table 4: Analysis of presolving for Johnson-testset.

name	time	pre time	$^{p(n)}/_{n}$	$^{p(m)}/_{m}$	#merge	#del	#add	#rounds
Johnson_11_4_0	0.04	0.01	0.83	1.28	3	52	3778	2
Johnson_13_5_1	23.92	0.33	0.57	0.47	5	550	38 685	2
Johnson_11_3_2	0.21	0.00	0.67	0.83	0	55	694	3
Johnson_15_4_1	115.15	0.65	0.51	0.26	0	673	5298	2
Johnson_14_5_0	275.54	0.24	0.89	1.33	0	211	67 404	1
Johnson_18_5_4	3598.83	1.24	0.98	1.55	0	153	162 546	3
Johnson_11_3_1	0.59	0.01	0.44	0.20	0	92	172	2
Johnson_12_4_3	3599.99	0.01	0.87	1.34	0	66	4610	2
Johnson_14_5_1	6.34	1.08	0.51	0.38	6	966	82 209	2
Johnson_15_5_4	3599.83	0.18	0.97	1.49	0	105	41 823	2
Johnson_14_3_1	8.05	0.04	0.39	0.18	0	221	1191	3
Johnson_14_4_0	0.26	0.15	0.64	0.62	6	359	29 569	2
Johnson_14_4_1	17.08	0.34	0.50	0.27	0	500	5116	2
Johnson_16_6_0	3597.38	2.63	0.96	1.52	0	320	505 359	1
Johnson_14_4_3	2.28	0.04	0.91	1.45	0	91	12 161	2
Johnson_12_5_0	110.10	0.02	0.95	1.49	0	37	4893	1
Johnson_13_4_3	47.28	0.02	0.89	1.41	0	78	7770	3
Johnson_14_6_1	3598.90	1.10	0.84	1.08	0	482	197 400	1
Johnson_19_3_0	0.40	0.32	0.35	0.11	10	619	4879	3
Johnson_20_4_3	3599.56	0.44	0.96	1.56	0	190	98 564	1
Johnson_15_4_0	0.67	0.36	0.59	0.49	7	559	46 451	2
Johnson_17_4_3	3599.87	0.13	0.94	1.51	0	136	37 850	1
Johnson_13_3_1	4.17	0.02	0.45	0.22	0	156	574	2
Johnson_14_6_5	3599.89	0.17	0.97	1.45	0	91	36 212	2
Johnson_17_3_2	3599.97	0.03	0.80	1.27	0	136	8575	3
Johnson_15_4_3	776.58	0.06	0.92	1.48	0	105	18 650	1
Johnson_12_5_1	27.96	0.10	0.64	0.61	4	278	16 387	2
Johnson_15_6_0	3599.40	0.60	0.98	1.61	0	123	137 706	1
Johnson_16_6_5	3599.51	0.82	0.99	1.49	0	120	124 404	1
Johnson_16_5_4	3599.70	0.31	0.97	1.47	0	120	63 005	1
Johnson_14_5_2	3598.48	1.52	0.57	0.33	0	856	9689	2
Johnson_18_4_3	3599.76	0.24	0.95	1.53	0	153	53 382	3
Johnson_18_3_2	21.59	0.04	0.81	1.30	0	153	11 300	2
Johnson_11_4_3	571.32	0.00	0.83	1.23	0	55	2416	1
Johnson_16_3_1	18.69	0.08	0.32	0.11	2	379	1514	3
Johnson_12_4_2	5.61	0.05	0.43	0.30	0	283	4488	1
Johnson_17_3_0	0.14	0.13	0.32	0.09	29	436	3003	4
Johnson_20_3_0	0.56	0.48	0.33	0.09	12	756	5862	3
Johnson_13_5_2	1426.95	0.61	0.53	0.30	0	602	8891	2
Johnson_13_5_0	290.13	0.07	0.92	1.41	0	98	20 078	1
Johnson_14_6_0	3599.89	0.11	0.99	1.65	0	38	28 382	1
Johnson_11_4_1	0.05	0.04	0.36	0.17	7	203	1300	3
Johnson_15_6_5	3599.66	0.38	0.98	1.47	0	105	69 268	1
Johnson_14_3_0	0.04	0.03	0.24	0.04	37	240	683	5
Johnson_10_4_0	0.04	0.00	0.89	1.31	0	24	844	1
Johnson_20_3_2	70.26	0.05	0.83	1.41	0	190	20 233	2
Johnson_10_4_2	1.48	0.01	0.54	0.31	1	95	157	3
Johnson_17_4_0	2.62	1.49	0.50	0.32	13	1188	105 649	2
Johnson_12_5_4	3599.98	0.02	0.92	1.42	0	66	7979	1
Johnson_11_4_2	0.26	0.02	0.37	0.20	0	207	1415	1
Johnson_15_3_1	7.76	0.05	0.30	0.09	3	317	1095	3
Johnson_12_4_1	0.37	0.07	0.52	0.30	0	239	1958	2
Johnson_13_4_0	0.11	0.06	0.70	0.79	5	213	16 791	2
Johnson_16_3_2	6.35	0.02	0.79	1.22	0	120	6306	2
Johnson_19_4_3	3599.68	0.32	0.96	1.55	0	171	73 550	2
Johnson_13_5_4	3599.96	0.04	0.94	1.47	0	78	15 001	2
Johnson_12_3_1	0.77	0.01	0.44	0.20	1	123	306	3
Johnson_10_4_1	0.02	0.01	0.40	0.21	4	122	612	2
Johnson_14_5_4	3599.92	0.09	0.95	1.50	0	91	26 249	1
Johnson_17_5_4	3599.44	0.60	0.98	1.52	0	136	104 644	2
Johnson_16_4_1	607.15	1.36	0.51	0.26	0	894	8339	2
Johnson_12_5_2	0.71	0.22	0.42	0.22	0	462	7247	3

Table 5: Analysis of presolving for binary-code-testset.

name	time	pre time	$p(n)/n$	$p(m)/m$	#merge	#del	#add	#rounds
BinaryCode_10_3	3599.86	0.14	0.79	1.01	0	220	32 827	2
BinaryCode_8_2	4.50	0.00	0.72	0.95	0	72	1747	2
BinaryCode_10_8	0.56	0.56	0.00	0.00	1	1022	0	3
BinaryCode_9_7	0.11	0.11	0.00	0.00	1	510	0	3
BinaryCode_11_3	3599.55	0.46	0.86	1.18	0	286	99 713	2
BinaryCode_10_4	0.53	0.27	0.33	0.22	0	682	17 546	3
BinaryCode_12_2	3601.41	0.41	0.96	1.60	0	156	107 247	1
BinaryCode_9_6	0.11	0.11	0.00	0.00	1	510	0	3
BinaryCode_8_5	0.02	0.02	0.00	0.00	1	254	0	3
BinaryCode_9_5	0.11	0.11	0.01	0.00	11	498	0	4
BinaryCode_9_4	0.09	0.09	0.02	0.00	14	489	1356	6
BinaryCode_10_2	3599.96	0.04	0.89	1.50	0	110	19 258	1
BinaryCode_11_4	12.36	1.02	0.52	0.51	0	981	114 511	2
BinaryCode_9_2	647.54	0.02	0.82	1.10	0	90	4516	3
BinaryCode_10_5	0.53	0.52	0.11	0.01	14	897	3263	6
BinaryCode_8_6	0.03	0.03	0.00	0.00	1	254	0	3
BinaryCode_9_3	20.02	0.04	0.68	0.71	0	165	6781	2
BinaryCode_13_2	3600.41	1.28	0.98	1.55	0	182	218 977	2
BinaryCode_10_7	0.52	0.52	0.00	0.00	1	1022	0	3
BinaryCode_12_3	3598.73	1.29	0.91	1.32	0	364	291 163	1
BinaryCode_11_2	3599.89	0.14	0.94	1.59	0	132	47 610	1
BinaryCode_8_4	0.02	0.02	0.01	0.00	11	242	0	4
BinaryCode_10_6	0.60	0.60	0.00	0.00	1	1022	0	3

Table 6: Comparison of CliSAT performance with and without presolving.

	with presolving		without presolving			
name	nodes	time	nodes	time	nodes ratio	time ratio
Johnson_15_5_4	10 413 200	3600.1	9 797 200	3600.1	0.94	1.00
Johnson_14_3_1	453	0.5	363 956	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	56 522 700	3600.0	51 291 700	3600.0	0.91	1.00
Johnson_12_5_0	1	0.5	11 922 800	3600.1	Large	Large
Johnson_13_4_3	82 147 700	3600.0	79 088 000	3600.0	0.96	1.00
Johnson_14_6_1	286 000	3600.6	1 480 400	3600.1	5.18	1.00
Johnson_19_3_0	1	0.5	1	0.5	1.00	1.00
Johnson_11_4_0	1	0.5	179	0.5	179.00	1.00
Johnson_20_4_3	6 557 300	3600.1	7 823 300	3600.1	1.19	1.00
Johnson_15_4_0	1	0.5	27	0.5	27.00	1.00
Johnson_17_4_3	19 340 800	3600.0	15 911 200	3600.0	0.82	1.00
Johnson_13_3_1	98	0.5	25 972	0.5	265.02	1.00
Johnson_14_6_5	7 912 600	3600.1	9 736 000	3600.0	1.23	1.00
Johnson_17_3_2	122 548 400	3600.0	113 778 500	3600.0	0.93	1.00
Johnson_15_4_3	40 381 500	3600.1	37 396 000	3600.0	0.93	1.00
Johnson_12_5_1	1	0.5	20 767 200	3600.1	Large	Large
Johnson_16_6_5	2 160 800	3600.2	2 340 200	3600.2	1.08	1.00
Johnson_13_5_1	154	0.5	14 312 507	2247.6	Large	Large
Johnson_16_5_4	6 306 300	3600.1	6 097 300	3600.1	0.97	1.00
Johnson_14_5_2	58 920 694	2117.0	75 657 900	3600.0	1.28	1.70
Johnson_18_4_3	13 946 000	3600.1	12 211 900	3600.1	0.88	1.00
Johnson_18_3_2	109 573 600	3600.0	98 890 700	3600.0	0.90	1.00
Johnson_11_4_3	115 265 515	2713.3	162 187 300	3600.0	1.41	1.33
Johnson_16_3_1	388	0.5	2 759 036	62.4	Large	124.80
Johnson_12_4_2	21	0.5	1 803 397	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	1 196 258	106.2	363.60	212.40
Johnson_11_3_2	193	0.5	2 431 438	28.6	Large	57.20
Johnson_13_5_0	1	0.5	2 716 200	3600.1	Large	Large
Johnson_11_4_1	50	0.5	64	0.5	1.28	1.00
Johnson_15_6_5	4 694 500	3600.1	3 655 600	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	82 834 400	3600.0	71 574 000	3600.0	0.86	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	61 369 400	3600.0	59 947 200	3600.0	0.98	1.00
Johnson_15_4_1	183	0.5	4766	1.4	26.04	2.80
Johnson_11_4_2	1	0.5	12 290	0.5	Large	1.00
Johnson_15_3_1	452	0.5	1 770 076	26.3	Large	52.60
Johnson_12_4_1	1	0.5	93	0.5	93.00	1.00
Johnson_13_4_0	1	0.5	40	0.5	40.00	1.00
Johnson_16_3_2	155 197 200	3600.0	144 259 000	3600.0	0.93	1.00
Johnson_19_4_3	9 729 900	3600.1	8 805 400	3600.1	0.90	1.00
Johnson_13_5_4	36 279 500	3600.1	33 686 100	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	17 723 600	3600.1	18 491 400	3600.1	1.04	1.00
Johnson_14_5_0	1	0.5	1 059 900	3600.5	Large	Large
Johnson_17_5_4	3 435 700	3600.1	3 925 200	3600.1	1.14	1.00
Johnson_16_4_1	171	0.5	21 337	8.2	124.78	16.40
Johnson_12_5_2	1	0.5	4659	0.5	Large	1.00
Johnson_18_5_4	2 136 900	3600.1	2 156 900	3600.1	1.01	1.00
Johnson_11_3_1	3	0.5	1052	0.5	350.67	1.00
Johnson_12_4_3	115 520 100	3600.0	106 809 100	3600.0	0.92	1.00
Johnson_14_5_1	1	0.5	11 121 400	3600.1	Large	Large