Efficient Propagation Techniques for Handling Cyclic Symmetries in Binary Programs

Jasper van Doornmalen and Christopher Hojny

Combinatorial Optimization Group, Eindhoven University of Technology
email: {m.j.v.doornmalen, c.hojny}@tue.nl

Abstract

The presence of symmetries of binary programs typically degrade the performance of branch-and-bound solvers. In this article, we derive efficient variable fixing algorithms to discard symmetric solutions from the search space based on propagation techniques for cyclic groups. Our algorithms come with the guarantee to find all possible variable fixings that can be derived from symmetry arguments, i.e., one cannot find more variable fixings than those found by our algorithms. Since every permutation symmetry group of a binary program has cyclic subgroups, the derived algorithms can be used to handle symmetries in any symmetric binary program. In experiments we also provide numerical evidence that our algorithms handle symmetries more efficiently than other variable fixing algorithms for cyclic symmetries.

2020 Mathematics Subject Classification: 90C09, 90C27, 90C57
Keywords: symmetry handling, cyclic group, propagation, branch-and-bound

1 Introduction

We consider binary programs $\max\{c^\top x : Ax \leq b,\ x \in \{0,1\}^n\}$, with $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, and $c \in \mathbb{R}^n$ for some positive integers m and n. A standard method to solve binary programs is branch-and-bound, which iteratively explores the search space by splitting the initial binary program into subproblems, see Land and Doig [18]. Although branch-and-bound can solve binary programs with thousands of variables and constraints rather efficiently, the performance of branch-and-bound usually degrades drastically if symmetries are present because it unnecessarily explores symmetric subproblems. Such a *symmetry* is a permutation γ of $[n] \coloneqq \{1, \dots, n\}$ that acts on a vector $x \in \mathbb{R}^n$ by permuting its coordinates, i.e., $\gamma(x) \coloneqq (x_{\gamma^{-1}(1)}, \dots, x_{\gamma^{-1}(n)})$, and that adheres to the following two properties: ***(i)*** it maps feasible solutions to feasible solutions, i.e., $Ax \leq b$ if and only if $A\gamma(x) \leq b$, and ***(ii)*** it preserves the objective value, i.e., $c^\top x = c^\top \gamma(x)$. Two solutions x and y are symmetric if there exists a symmetry γ such that $y = \gamma(x)$.

Various methods to remove symmetric parts from the search space have been proposed in the literature, ranging, among others, from cutting planes, variable fixing or branching rules, or propagation methods, see below for references. The common ground of many of these methods is to impose a lexicographic order on the solution space and to exclude solutions that are not lexicographically maximal in their symmetry class. This approach removes all symmetric copies of a solution, and thus, handles all symmetries. However, deciding whether a solution is lexicographically maximal in its symmetry class is coNP-complete [1]. This makes lexicographic order based methods often computationally expensive, since no generally applicable polynomial-time algorithms for such methods exist, unless P = coNP. For this reason, one typically weakens the requirement of removing all symmetric copies, or investigates symmetry handling methods for particular groups Γ for which methods exist that run in polynomial time.

In this article, we follow the latter approach by investigating propagation techniques, whose idea is as follows. If we are given a subproblem, some of the variables might have been fixed, e.g., due to branching decisions. For a symmetry γ, a propagation algorithm looks for further variables that need to be fixed to guarantee that a solution x that adheres to the fixings of the subproblem is not lexicographically smaller than the permuted vector $\gamma(x)$. Of course, if we are given a set of permutations Π, then this propagation step can be carried out for every $\gamma \in \Pi$. Since symmetry groups may have size $2^{\Omega(n)}$, however, blindly applying propagation for each individual permutation is computationally intractable.

Although the full symmetric group has exponential order, polynomial time propagation algorithms for certain actions of full symmetric groups have been designed [2, 13]. To the best of our knowledge, however, it seems that no efficient propagation algorithms for cyclic groups, i.e., groups generated by a single permutation, are known. At first glance, finding algorithms for cyclic groups seems to be trivial as cyclic shifts have a very simple structure. But despite the simplicity of cyclic shifts, we have no understanding of the structure of binary points being lexicographically maximal with respect to cyclic group actions. In fact, the structure of these points is rather complicated and does not seem to follow an obvious pattern, see [23, Chap. 3.2.2]. It has been an open problem for at least ten years to gain further insights into the structure of lexicographically maximal points for cyclic groups.

We believe that this is an important gap, because every permutation group Γ has cyclic subgroups. Thus, instead of applying propagation for individual permutations, we can apply propagation for corresponding cyclic subgroups to find stronger reductions. In particular, knowledge on cyclic groups can be used for *every* symmetric binary program, whereas algorithms for symmetric groups need assumptions on Γ. We emphasize that, although cyclic groups Γ are generated by a single permutation γ, they also might have superpolynomial size[1]. That is, efficient algorithms for cyclic groups are not immediate.

Literature Review Handling symmetries in binary programs via propagation is not a novel technique. It originates from constraint programming, and symmetry handling techniques in this context are discussed, among others, in [4, 15, 16, 29]. For binary programs, Bendotti et al. [2] describe an algorithm to find variable fixings for certain actions of symmetric groups. Further fixings can be found if the variables affected by the symmetric group are contained in set packing or partitioning constraints, see Kaibel et al. [13]. Moreover, if instead of an entire group only the action of a single permutation is considered, propagation algorithms for so-called symresacks can be used [3, 10]. These algorithms are *complete* in the sense that they find all possible symmetry-based variable fixings derivable from a set of fixed variables. In contrast to this, orbital fixing [26, 31] can be used for arbitrary groups, however, without any guarantee on completeness. Margot [25, 26] presents isomorphism pruning, a propagation technique to prune nodes of a branch-and-bound tree that do not contain lexicographically maximal solutions.

Besides propagation, further methods for handling symmetries in binary programs exist such as cutting planes [7, 9, 10, 14, 19, 20, 21, 22, 35], branching rules [30, 31], or model reformulations [6].

Contribution The aim of this article is to devise efficient propagation algorithms for cyclic groups. To this end, we derive an auxiliary result for arbitrary sets Π of permutations first. As mentioned above, we can find variable fixings by propagating each individual permutation $\gamma \in \Pi$ using symresack propagation. As mentioned in [3], a single symresack can be propagated in $O(n)$ time. Thus, we can find all fixings derived from all individual permutations in $\mathcal{O}(n^2|\Pi|)$ time, because there are n potential fixings and each might trigger another round of propagating Π. In Section 2, we improve this running time to $\mathcal{O}(n|\Pi|)$ by introducing suitable data structures and a careful analysis of dependencies among the permutations in Π. This result forms the basis for our efficient algorithms for cyclic groups that we derive in Section 3. To this end, we provide a novel characterization of lexicographically maximal elements in certain cyclic groups. This characterization is then used to derive our efficient algorithms for a broad class of cyclic groups. We in particular show that our algorithms find all possible variable fixings, i.e., they are as strong as possible. In Section 4, we report on numerical results on a broad

[1] For example, if γ has k disjoint cycles of mutually distinct prime lengths.

test set containing both instances with many cyclic symmetries and general benchmark instances. If cyclic symmetries are the dominant type of symmetries, these experiments show that our methods outperform the immediate approach of handling all permutations individually. For ease of presentation, we skip most proofs in the discussion of Section 2; the missing proofs are then provided in Appendix A.

Basic Definitions and Notation Throughout this article, we assume that n is a positive integer. Given $k \in [n+1] \coloneqq \{1,\dots,n+1\}$ and vectors $x, y \in \mathbb{R}^n$, we say that $x =_k y$ if and only if $x_i = y_i$ for all $i \in [k-1]$. To decide whether x and y can be distinguished *up to position* k, we write $x \succ_k y$ if and only if there exists $i \in [k-1]$ such that $x =_i y$ and $x_i > y_i$. The relation $x \succeq_k y$ holds if $x =_k y$ or $x \succ_k y$. These relations define the *partial lexicographic order up to* k. When $k = n+1$, we write $=$, $\succ$, and $\succeq$ instead of $=_k$, $\succ_k$, and $\succeq_k$, respectively. In this case, we say that x is *equal to*, *lexicographically greater*, and *lexicographically not smaller* than y, respectively.

Let $\mathcal{S}_n$ be the *symmetric group* on $[n]$. For $\gamma \in \mathcal{S}_n$, the set of all binary vectors that are lexicographically not smaller than their images $\gamma(x)$ is denoted by $\mathcal{X}_\gamma \coloneqq \{x \in \{0,1\}^n : x \succeq \gamma(x)\}$. Moreover, for $\Pi \subseteq \mathcal{S}_n$, denote $\mathcal{X}_\Pi \coloneqq \bigcap_{\gamma\in\Pi} \mathcal{X}_\gamma$. Analogously, we define $\mathcal{X}_\gamma^{(k)}$ and $\mathcal{X}_\Pi^{(k)}$ if we use the relation $\succeq_k$ instead of $\succeq$. If the set Π defines a group, we typically use the symbol Γ to denote this.

If the generating permutations of Γ are $\gamma_1,\dots,\gamma_m$ for some $m \in \mathbb{N}$, then this is denoted with angle brackets $\Gamma \coloneqq \langle \gamma_1,\dots,\gamma_m\rangle$. If permutations are defined explicitly, we always use the cycle representation. For disjoint sets $I_0, I_1 \subseteq [n]$, we define $F(I_0, I_1) \coloneqq \{x \in \{0,1\}^n : x_i = 0 \text{ for } i \in I_0 \text{ and } x_i = 1 \text{ for } i \in I_1\}$. The sets I_0 and I_1 thus define the indices of binary (solution) vectors that are fixed to 0 and 1, respectively. The situation where the entry x_i, $i \in [n]$, of a vector x is fixed to a value $b \in \{0,1\}$ is called a *fixing*, and we denote this by a tuple $f = (i,b) \in [n] \times \{0,1\}$. By a slight abuse of terminology, we say in the following that entry i gets fixed rather than entry x_i of vector x to keep notation short. The *converse fixing* of $f = (i,b)$ is denoted by $\bar{f} \coloneqq (i, 1-b)$. A set of fixings $C \subseteq [n] \times \{0,1\}$ is called a *conjunction*, and we define $V(C) \coloneqq \{x \in \{0,1\}^n : x_i = b \text{ for } (i,b) \in C\}$ as the set of binary vectors respecting the fixings in conjunction C.

To handle symmetries, the main goal of this article is to find, given a set of initial fixings I_0 and I_1 larger sets I'_0 and I'_1 with $\mathcal{X}_\Pi \cap F(I'_0, I'_1) = \mathcal{X}_\Pi \cap F(I_0, I_1)$. Once we have identified such sets I'_0 and I'_1, the variables in $I'_0 \setminus I_0$ and $I'_1 \setminus I_1$ can be fixed to 0 and 1, respectively. Thus, we can derive variable fixings based on symmetry considerations. To obtain the strongest effect, we are interested in sets I'_0 and I'_1 being as large as possible. Note that the largest sets, denoted $I^\star_0$ and $I^\star_1$, are unique: Let C be the conjunction representing the fixings (I_0, I_1). Let $X_\Pi(C) \coloneqq \{x \in \mathcal{X}_\Pi : x_i = b \text{ for } (i,b) \in C\}$ be the set of binary vectors in $\mathcal{X}_\Pi$ respecting the fixings in C. For a set $X \subseteq \{0,1\}^n$, let $\mathcal{C}(X) \coloneqq \{(i,b) \in [n] \times \{0,1\} : x_i = b \text{ for all } x \in X\}$ be the set of fixings in X. Then $\mathcal{C}(X_\psi(C))$ is the unique largest conjunction of fixings for initial fixings C with respect to $\mathcal{X}_\Pi$, from which we derive $I^\star_0$ and $I^\star_1$.

For a subgroup Γ of $\mathcal{S}_n$, denoted $\Gamma \leq \mathcal{S}_n$, we frequently use stabilizer subgroups. Given a set $I \subseteq [n]$, the *pointwise stabilizer* is $\mathrm{STAB}(I,\Gamma) \coloneqq \{\gamma \in \Gamma : \gamma(i) = i,\ i \in I\}$. The *setwise stabilizer* is $\mathrm{stab}(I,\Gamma) \coloneqq \{\gamma \in \Gamma : \gamma(i) \in I,\ i \in I\}$. For singleton sets, we write $\mathrm{STAB}(i,\Gamma)$ and $\mathrm{stab}(i,\Gamma)$ instead of $\mathrm{STAB}(\{i\},\Gamma)$ and $\mathrm{stab}(\{i\},\Gamma)$, respectively. The *orbit* of a solution x with respect to a group Γ is $\{\gamma(x) : \gamma \in \Gamma\}$. Last, for a permutation $\gamma \in \mathcal{S}_n$, the *restriction of* γ *to* I, $\delta = \mathrm{restr}(\gamma, I)$, satisfies $\delta(i) = \gamma(i)$ for $i \in I$ and $\delta(i) = i$ for $i \notin I$. For groups $\Gamma \leq \mathcal{S}_n$, we denote $\mathrm{restr}(\Gamma, I) \coloneqq \{\mathrm{restr}(\gamma, I) : \gamma \in \Gamma\}$. Note that $\delta \in \mathcal{S}_n$ if and only if $\gamma(I) = I$, and that $\mathrm{restr}(\Gamma, I) \leq \mathcal{S}_n$ if and only if I corresponds to the union of orbits of elements from I.

2 Propagation of Individual Permutations In a Set

The main goal of this article is to devise efficient propagation algorithms that enforce a solution to be lexicographically maximal in its orbit with respect to a cyclic group. As we will see in the next section, the main workhorse of these algorithms is an efficient

Algorithm 1: Determine the complete set of fixings for each individual constraint $x \succeq \gamma(x)$ for all $\gamma \in \Pi$.

input : set of permutations $\Pi \subseteq \mathcal{S}_n$, and initial set of fixings (I_0, I_1)
output: INFEASIBLE if an empty inf-conjunction for some $\gamma \in \Pi$ is found by the algorithm, or FEASIBLE and the set of fixings that is complete for each individual permutation in Π.

if $F(I_0, I_1) = \emptyset$ **then** **return** INFEASIBLE;
$t \leftarrow 0$; $(I_0^t, I_1^t) \leftarrow (I_0, I_1)$;
foreach $\gamma \in \Pi$ **do** $i_\gamma \leftarrow 1$;
while *there is a* $\gamma \in \Pi$ *not satisfying sufficient conditions for completeness* **do**
  $i_\gamma \leftarrow i_\gamma + 1$; $t \leftarrow t + 1$;
  repeat
    if *there is a* $\delta \in \Pi$ *with* i_δ*-inf-conjunction* $\emptyset$ **then** **return** INFEASIBLE;
    else if *there is a* $\delta \in \Pi$ *with* i_δ*-inf-conjunction* $\{(i,b)\}$ *and* $i \notin I_{1-b}^t$ **then**
      Apply fixing $(i, 1-b)$: $(I_b^{t+1}, I_{1-b}^{t+1}) \leftarrow (I_b^t, I_{1-b}^t \cup \{i\})$; $t \leftarrow t + 1$;
    else **break** repeat-loop ;
return FEASIBLE, (I_0^t, I_1^t);

subroutine that, for a given set of permutations Π, propagates $x \succeq \gamma(x)$ for all $\gamma \in \Pi$. To make this statement precise, we introduce the following terminology and notation.

Let $\Pi \subseteq \mathcal{S}_n$, and $I_0, I_1 \subseteq [n]$ be disjoint. Our aim is to find larger sets I_0', I_1' with $\mathcal{X}_\Pi \cap F(I_0, I_1) = \mathcal{X}_\Pi \cap F(I_0', I_1')$ by iteratively *applying valid fixings* for the constraints $x \succeq \gamma(x)$ for each $\gamma \in \Pi$. A *fixing* is a tuple $(i, b) \in [n] \times \{0, 1\}$ that encodes the situation where the value of entry i is fixed to b. We say that a fixing is *valid* for a permutation $\gamma \in \Pi$ and a disjoint set of fixings $I_0, I_1 \subseteq [n]$ if all $x \in \mathcal{X}_\gamma \cap F(I_0, I_1)$ have $x_i = b$. Such a fixing (i, b) is *applied* if entry i is added to the index set I_b. This way, the simple Observation 2.1 below shows how additional fixings can be found. If no further valid fixing can be found by considering any individual constraint $x \succeq \gamma(x)$ for $\gamma \in \Pi$, then this is a *complete set of fixings for each permutation in* Π, denoted by I_0', I_1'. We want to stress that these do not need to be the complete set of fixings for $\mathcal{X}_\Pi \cap F(I_0, I_1)$: more fixings could exist, as we will demonstrate in Example 3.1.

Using this terminology, this section's goal is to find an efficient algorithm to determine the complete set of fixings for all $\gamma \in \Pi$. As mentioned in the introduction, a trivial running time of such an algorithm is $\mathcal{O}(n^2|\Pi|)$. By introducing suitable data structures and implications among the different permutations in Π, however, we show that the running time can be reduced to $\mathcal{O}(n|\Pi|)$. To develop our algorithm, we make use of the following simple observation.

Observation 2.1. *Let* $\Pi \subseteq \mathcal{S}_n$ *and* $I_0, I_1 \subseteq [n]$ *be disjoint. Suppose we want to propagate* $x \succeq \gamma(x)$ *for all* $\gamma \in \Pi$. *Then,* $i \in [n] \setminus (I_0 \cup I_1)$ *can be added to* I_0 *(resp.* I_1*) if and only if every* $x \in F(I_0, I_1)$ *with* $x_i = 1$ *(resp.* $x_i = 0$*) satisfies* $x \prec \gamma(x)$ *for some* $\gamma \in \Pi$.

Consequently, if $\mathcal{X}_\Pi \cap F(I_0, I_1) \neq \emptyset$ and $\mathcal{X}_\Pi \cap F(I_0 \cup \{i\}, I_1) = \emptyset$, we know that i can be fixed to 1 (and analogously we can argue for 0-fixings). Since adding i to I_0 makes the latter set empty, we refer to such a fixing as an *infeasibility* fixing. To algorithmically exploit Observation 2.1, we are thus interested in finding infeasibility fixings (i, b) as $(i, 1-b)$ is then a valid fixing. For our algorithm, it will turn out that also considering sets of fixings that lead to infeasibility, if applied simultaneously, are of importance. As mentioned in the introduction, these sets are referred to as *conjunctions*. *Inf-conjunctions* are sets of fixings that yield infeasibility if all fixings of the set are applied. Moreover, we specify special types of inf-conjunctions. Let $k \in [n+1]$ and $x \in \{0,1\}^n$. Note that $x \prec_k \gamma(x)$ implies $x \prec \gamma(x)$, and that equivalence holds if $k = n + 1$. A *k-inf-conjunction* is a conjunction $C \subseteq [n] \times \{0, 1\}$ such that all $x \in F(I_0, I_1)$ with $x_i = b$ for $(i, b) \in C$ have $x \prec_k \gamma(x)$. Note that C is also an inf-conjunction for γ.

Algorithm 1 describes how additional fixings can be found. To simplify the analysis,

$$
\begin{array}{c}
\begin{array}{ccc} x & \gamma_1(x) & \gamma_2(x) \end{array} \\
\begin{bmatrix}
{}_{(1)}\,\_ & {}_{(5)}\,1 & {}_{(5)}\,1 \\
{}_{(2)}\,\_ & {}_{(7)}\,\_ & {}_{(6)}\,0 \\
{}_{(3)}\,\_ & {}_{(3)}\,\_ & {}_{(1)}\,\_ \\
{}_{(4)}\,0 & {}_{(8)}\,\_ & {}_{(2)}\,\_ \\
{}_{(5)}\,1 & {}_{(2)}\,\_ & {}_{(4)}\,0 \\
{}_{(6)}\,0 & {}_{(1)}\,\_ & {}_{(3)}\,\_ \\
{}_{(7)}\,\_ & {}_{(4)}\,0 & {}_{(7)}\,\_ \\
{}_{(8)}\,\_ & {}_{(6)}\,0 & {}_{(8)}\,\_
\end{bmatrix}
\end{array}
\overset{x_1 \leftarrow 1}{\rightsquigarrow}
\begin{array}{c}
\begin{array}{ccc} x & \gamma_1(x) & \gamma_2(x) \end{array} \\
\begin{bmatrix}
{}_{(1)}\,1 & {}_{(5)}\,1 & {}_{(5)}\,1 \\
{}_{(2)}\,\_ & {}_{(7)}\,\_ & {}_{(6)}\,0 \\
{}_{(3)}\,\_ & {}_{(3)}\,\_ & {}_{(1)}\,1 \\
{}_{(4)}\,0 & {}_{(8)}\,\_ & {}_{(2)}\,\_ \\
{}_{(5)}\,1 & {}_{(2)}\,\_ & {}_{(4)}\,0 \\
{}_{(6)}\,0 & {}_{(1)}\,1 & {}_{(3)}\,\_ \\
{}_{(7)}\,\_ & {}_{(4)}\,0 & {}_{(7)}\,\_ \\
{}_{(8)}\,\_ & {}_{(6)}\,0 & {}_{(8)}\,\_
\end{bmatrix}
\end{array}
\overset{x_7 \leftarrow 0}{\rightsquigarrow}
\begin{array}{c}
\begin{array}{ccc} x & \gamma_1(x) & \gamma_2(x) \end{array} \\
\begin{bmatrix}
{}_{(1)}\,1 & {}_{(5)}\,1 & {}_{(5)}\,1 \\
{}_{(2)}\,\_ & {}_{(7)}\,0 & {}_{(6)}\,0 \\
{}_{(3)}\,\_ & {}_{(3)}\,\_ & {}_{(1)}\,1 \\
{}_{(4)}\,0 & {}_{(8)}\,\_ & {}_{(2)}\,\_ \\
{}_{(5)}\,1 & {}_{(2)}\,\_ & {}_{(4)}\,0 \\
{}_{(6)}\,0 & {}_{(1)}\,1 & {}_{(3)}\,\_ \\
{}_{(7)}\,0 & {}_{(4)}\,0 & {}_{(7)}\,0 \\
{}_{(8)}\,\_ & {}_{(6)}\,0 & {}_{(8)}\,\_
\end{bmatrix}
\end{array}
$$

Figure 1: Figure for Example 2.2. Each matrix shows the state of the algorithm at a point. The columns of the matrix correspond to the vectors x, and the permuted vectors $\gamma_1(x)$ and $\gamma_2(x)$, and for each entry in the matrix the variable index is written left to it. The left matrix is the initial state. The second matrix is after application of $x_1 \leftarrow 1$, and the third matrix is after application of $x_7 \leftarrow 0$.

we maintain a timestamp t, starting at 0. Also, for each permutation $\gamma \in \Pi$, the index until which the partial lexicographic is considered is i_γ, which is initialized at 1. If a time-specification is needed, the value of i_γ at time t is denoted by i_γ^t. The set of fixings at this time is denoted by I_0^t and I_1^t. The idea of our algorithm is to iterate over permutations from Π for which we can potentially find further variable fixings. It checks whether there exists a permutation γ in this list that admits an inf-conjunction consisting of at most a single element: If there is an empty inf-conjunction for γ, then $\mathcal{X}_\gamma \cap F(I_0, I_1) = \emptyset$ and the algorithm terminates since infeasibility has been determined. Otherwise, for all inf-conjunctions $\{(i, b)\}$ that can be found for one permutation in the list, the algorithm applies the fixing $(i, 1-b)$. To be able to find inf-conjunctions efficiently, the algorithm does not check for the existence of arbitrary inf-conjunctions. Instead, only inf-conjunctions are checked that certify infeasibility for a partial lexicographic order. To make this precise, we introduce the following terminology.

Note that Algorithm 1 is not practically applicable yet, because it does not specify details on how to execute it. In the remainder of this section, we provide these missing details. In particular, we derive structural properties of inf-conjunctions and develop efficient data structures that allow us to execute the algorithm in $\mathcal{O}(n|\Pi|)$ time. Before doing so, we provide an example that illustrates the execution of this algorithm, and prove that this algorithm is correct if it terminates.

Example 2.2. Let $\gamma_1 = (1, 6, 8, 4, 7, 2, 5)$, $\gamma_2 = (1, 3, 6, 2, 4, 5)$, $\Pi = \{\gamma_1, \gamma_2\}$, and let the initial fixings be $I_0 = \{4, 6\}$ and $I_1 = \{5\}$ encoded by $x = (\_, \_, \_, 0, 1, 0, \_, \_)$, where _ represents an unfixed entry. We execute a few steps of the algorithm, and the fixing updates are shown in Figure 1. More precisely, we discuss which permutations are selected at each iteration in Line 4, and which of the cases of Lines 7–10 applies. Later we specify how the selection conditions work algorithmically, and how k-inf-conjunctions can be detected.

In the first iteration, we select γ_1, set $i_{\gamma_1} \leftarrow 2$. There is a 2-inf-conjunction $\{(1, 0)\}$ for γ_1, since choosing $x_1 \leftarrow 0$ yields $x \prec_2 \gamma_1(x)$. Hence, we apply fixing $(1, 1)$, which fixes entry 1 to value 1. Any remaining i_δ-inf-conjunction for $\delta \in \Pi$ needs at least two fixings, so we continue with the next iteration. Again, select γ_1 and set $i_{\gamma_1} \leftarrow 3$. Since x_2, x_7 are both unfixed, no 3-inf-conjunction of cardinality less than two exists. Set $i_{\gamma_1} \leftarrow 4$, and we encounter a fixed point 3. Set $i_{\gamma_1} \leftarrow 5$, we have $(x_4, x_8) = (0, \_)$. If the value of x_2 and x_7 is the same, then x_8 must become 0, as well. Set $i_{\gamma_1} \leftarrow 6$, we encounter $(x_5, x_2) = (1, \_)$. In this case, if the columns x and $\gamma_1(x)$ are equal up to entry 5, and $x_2 = 0$, then no 6-inf-conjunction for γ_1 with cardinality 1 can be found. Otherwise, if $x_2 = 1$, we can continue. Choose $i_{\gamma_1} \leftarrow 7$. Then, $(x_6, x_1) = (0, 1)$, which means that $x \prec_7 \gamma_1(x)$ if for all entries $i < 6$ we have that the value of x_i is the same as $\gamma_1(x)_i$. If $x_7 = 1$, to ensure $x \succeq_6 \gamma_1(x)$, we must have $x_2 = 1$ and $x_8 = 0$, but in that case $x \prec_7 \gamma(x)$, so $\{(7, 1)\}$ is a 7-inf-conjunction for γ_1. Hence, apply fixing $(7, 0)$.

Similar steps can be applied to permutation γ_2, but no further fixings can be deduced. Namely, if $x_2 \leftarrow 0$ then $x_3 \leftarrow 1$ and we find $x \succ_6 \gamma_2(x)$, and if $x_2 \leftarrow 1$ then $x \succ_3 \gamma_2(x)$.

We next show that Algorithm 1 indeed finds the complete set of fixings.

Lemma 2.3. *If Algorithm 1 terminates, then it correctly detects infeasibility, or finds the complete set of fixings for each permutation in* Π.

Proof. First of all, we show that the algorithm never adds incorrect fixings. If the algorithm applies a fixing (i, b), then this fixing is due to Line 9, so there is a $\delta \in \Pi$ such that $\{(i, 1-b)\}$ is an i_δ-inf-conjunction. Hence, all $x \in \mathcal{X}_\delta^{(i_\delta)} \cap F(I_0^t, I_1^t) \supseteq \mathcal{X}_\delta \cap F(I_0^t, I_1^t)$ have $x_i = b$, showing that (i, b) is a valid fixing.

If feasibility is returned at time t, then the set of fixings (I_0^t, I_1^t) satisfies sufficient conditions for completeness for all permutations $\gamma \in \Pi$. Together with the first paragraph of this proof, (I_0^t, I_1^t) defines a complete set of fixings for all permutations $\gamma \in \Pi$, and $\mathcal{X}_\Pi \cap F(I_0, I_1) = \mathcal{X}_\Pi \cap F(I_0^t, I_1^t)$.

Otherwise infeasibility is returned. If this is due to the first line, then there does not exist any vector satisfying the initial sets of fixings I_0, I_1, so no valid solution exists for any $\gamma \in \Pi$. Otherwise it is due to Line 7 at time t. Then the empty set is a valid i_δ-inf-conjunction for some $\delta \in \Pi$. Therefore, the empty set is an inf-conjunction, and thus infeasibility is found without applying any fixing. Hence, infeasibility is correctly returned. □

We now proceed with the missing detail to turn Algorithm 1 into a practically applicable algorithm. A crucial step of Algorithm 1 is to identify k-inf-conjunctions. To this end, recall that $x \prec \gamma(x)$ if and only if there exists $j \in [n]$ such that $x_i = \gamma(x)_i$ for all $i \in [j-1]$, and $x_j = 0$ as well as $\gamma(x)_j = 1$. To generate k-inf-conjunctions, we thus also introduce conjunctions ensuring equality up to a certain index. A *k-eq-conjunction* is a conjunction $C \subseteq [n] \times \{0, 1\}$ such that, for all $x \in F(I_0, I_1)$ with $x_i = b$ for $(i, b) \in C$, we have $x \preceq_k \gamma(x)$, and $x =_k \gamma(x)$ holds at least for one such vector. We call it a k-eq-conjunction, because all vectors $x \in F(I_0, I_1)$ satisfying the conjunction C in $\mathcal{X}_\gamma$ (i.e., $x \in \mathcal{X}_\gamma \cap F(I_0, I_1) \cap V(C)$) satisfy $x =_k \gamma(x)$. We use such k-eq-conjunctions of a permutation to determine $(k+1)$-inf-conjunctions, by exploiting that $x \preceq_k \gamma(x)$ for this conjunction, and additionally ensuring $x_k < \gamma(x)_k$. This way, all vectors $x \in F(I_0, I_1)$ respecting the new conjunction either have that $x \prec_k \gamma(x)$, or that $x =_k \gamma(x)$ and $x_k < \gamma(x)_k$. In either case $x \prec_{k+1} \gamma(x)$ holds, so the new conjunction is a $(k+1)$-inf-conjunction. For each permutation, denote the set of inclusionwise minimal i_γ-inf-conjunctions at time t by D_γ^t, and, likewise, the set of inclusionwise minimal i_γ-eq-conjunctions by E_γ^t. Note that considering minimal conjunctions is sufficient to guarantee correctness of Algorithm 1.

To efficiently store and update all information needed for Algorithm 1, we encode D_γ^t and E_γ^t by a tree structure. For each permutation $\gamma \in \Pi$ and time t, we define an *implication tree*, which is a directed rooted tree $\mathcal{T}_\gamma^t = (V_\gamma^t, A_\gamma^t)$ with four types of vertices that partition V_γ^t: $V_{\text{root},\gamma}^t \coloneqq \{r_\gamma\}$, $V_{\text{cond},\gamma}^t$, $V_{\text{necc},\gamma}^t$, and $V_{\text{loose},\gamma}^t$ for the set with the root vertex r_γ, the *conditional fixing vertices*, the *necessary fixing vertices*, and the *loose end vertices*, respectively. Each *fixing vertex* $v \in V_{\text{cond},\gamma}^t \cup V_{\text{necc},\gamma}^t$ has an associated fixing, denoted by $f_v \in [n] \times \{0, 1\}$. We call this structure an implication tree, because it encodes conjunctions by implications of the type "if a set of fixings is applied, then also another fixing must be applied". When walking along a directed rooted path, the vertices in this path describe (dependent) implications as follows: If we encounter a necessary fixing vertex, then the associated fixing must be applied. If we encounter a conditional fixing vertex, then we continue following the path only if that fixing has already been applied. Added to this, if we encounter a loose end vertex on our walk, then all previously applied fixings ensure that $x =_{i_\gamma} \gamma(x)$ is found for all solution vectors x that respect the given fixings, and the fixings on the loose end vertex. When illustrating (parts of) implication trees, we draw conditional fixing vertices as diamonds, necessary fixing vertices as circles, loose end vertices as squares, and no outline is used in case of the root vertex or if the type is not important for the illustration. For fixing vertices, its fixings are written next to the vertex.

In Example 2.2, after applying the fixing $x_1 \leftarrow 1$, we have the following implications for γ_1: If we fix entry 2 to zero, then entries 7 and 8 must be fixed to zero, as well. Likewise, if we fix entry 7 to one, then entry 2 must be one, and entry 8 must be zero, and we find $x =_6 \gamma(x)$ for all solution vectors respecting these fixings. The associated implication tree is shown in Figure 2.

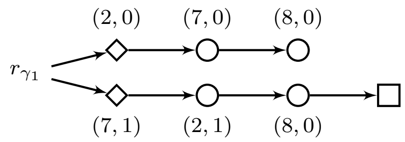


Figure 2: Implication tree of permutation γ_1 in Example 2.2 with $i_{\gamma_1} = 6$ and $x_1 = 1$.

Note that symmetry-based implications can always be encoded by *some* implication tree. In the following, however, we will see that, if using the right encoding and update strategies, the implication trees have a *particular* structure. This structure will allow us to implement Algorithm 1 efficiently. We first list properties on the relation between our particular implication trees and the sets D_γ^t and E_γ^t. Moreover, we give some properties of the implication trees that are maintained by the algorithm. We prove that the relations and properties are maintained using induction. We also show how Algorithm 1 can be executed, and how the data structures are updated consistently such that the mentioned properties indeed hold. Last, the running time of these methods is analyzed.

Property 2.4 (Relation between $\mathcal{T}_\gamma^t$ and E_γ^t)**.** Each eq-conjunction $C \in E_\gamma^t$ corresponds to a loose end vertex $v \in V_{\text{loose},\gamma}^t$ and vice versa. Given a loose end vertex $v \in V_{\text{loose},\gamma}^t$, the fixings of its conditional ancestors correspond to a minimal i_γ^t-eq-conjunction. Symbolically, let $\operatorname{Anc}(v)$ yield all (improper) ancestors of a vertex v (i.e., including v itself). For $v \in V_{\text{loose},\gamma}^t$ then $\{f_u : u \in \operatorname{Anc}(v) \cap V_{\text{cond},\gamma}^t\} \in E_\gamma^t$. This is a bijection, so any $C \in E_\gamma^t$ has a corresponding loose end vertex yielding this eq-conjunction.

We say that a conjunction $C \subseteq [n] \times \{0,1\}$ is *incompatible* with a set of fixings (I_0, I_1) if no vector exists that satisfies the fixings of (I_0, I_1) and C, i.e., $F(I_0, I_1) \cap V(C) = \emptyset$. We distinguish two types of inf-conjunctions: conjunctions $C \in D_\gamma^t$ that are incompatible with the fixings (I_0^t, I_1^t) and those that are not.

Property 2.5 (Relation between $\mathcal{T}_\gamma^t$ and D_γ^t)**.** For $F(I_0^t, I_1^t) \neq \emptyset$, each $C \in D_\gamma^t$ that is not incompatible with the fixings corresponds to a necessary fixing vertex $v \in V_{\text{necc},\gamma}^t$ and vice versa. For $v \in V_{\text{necc},\gamma}^t$, $\{f_u : u \in \operatorname{Anc}(v) \cap V_{\text{cond},\gamma}^t\} \cup \{\bar{f}_v\} \in D_\gamma^t$. In other words, infeasibility follows if the converse of f_v and all fixings associated with the conditional ancestors of v are applied. Regarding the minimal incompatible inf-conjunctions in D_γ^t, these are single conjunctions for each $i \in [n]$: $\{(i,0),(i,1)\}$ if $i \notin I_0^t \cup I_1^t$, or $\{(i,0)\}$ if $i \in I_1^t$, or $\{(i,1)\}$ if $i \in I_0^t$. For the special case where $F(I_0^t, I_1^t) = \emptyset$ (i.e., $D_\gamma^t = \{\emptyset\}$), the tree is marked infeasible.

To show that these properties of implication trees hold, we introduce some notation. Let $C_v := \{f_u : u \in \operatorname{Anc}(v) \cap (V_{\text{necc},\gamma}^t \cup V_{\text{cond},\gamma}^t)\}$ be the conjunction of all fixings found on the path from the root r_γ to v. Also, for fixing $f = (i, b)$, let $\operatorname{Ent}(f) = i$ be the *entry* of fixing f, and for a set of fixings $C \subseteq [n] \times \{0,1\}$, let $\operatorname{Ent}(C) := \{\operatorname{Ent}(f) : f \in C\}$. Using this notation, we show four auxiliary properties that we need to show the former properties.

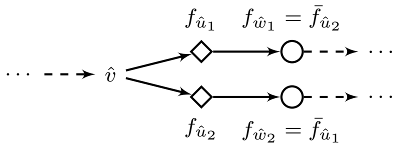
Property 2.6. Loose end vertices will occur only as leaves of the tree.

Property 2.7. If a loose end vertex $v \in V_{\text{loose},\gamma}^t$ exists, $\operatorname{Ent}(C_u) \subseteq \operatorname{Ent}(C_v)$ for all $u \in V_\gamma^t$. Also, any loose end vertex $v \in V_{\text{loose},\gamma}^t$ satisfies $\operatorname{Ent}(C_v) = \bigcup_{i \in [i_\gamma^t - 1]} \{i, \gamma^{-1}(i)\} \setminus (I_0^t \cup I_1^t)$.

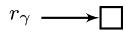
Property 2.8. Any vertex $\hat{v}$ in $\mathcal{T}_\gamma^t$ with outdegree larger than one has outdegree two, and its children $\hat{u}_1, \hat{u}_2$ are conditional vertices. In turn, $\hat{u}_1$ and $\hat{u}_2$ have outdegree one, and their children (resp. $\hat{w}_1$ and $\hat{w}_2$) are necessary fixing vertices with $f_{\hat{u}_1} = \bar{f}_{\hat{w}_2}$ and $f_{\hat{u}_2} = \bar{f}_{\hat{w}_1}$, where fixings $f_{\hat{u}_1}$ and $f_{\hat{u}_2}$ are on different entries. This is depicted in Figure 3a.

Property 2.9. In any rooted path P in the implication tree $\mathcal{T}_\gamma^t$, the fixings of all fixing vertices are on different entries, and no entry is in I_0^t or I_1^t.

These properties ensure that if a conjunction (either an k-inf-conjunction or k-eq-conjunction) is encoded by an implication tree, then no inclusionwise smaller or larger conjunction of the same type is represented by the same implication tree.



(a) Implication tree $\mathcal{T}^t_\gamma$ around vertex v with outdegree two.



(b) The initial implication tree $\mathcal{T}^t_\gamma$.

Figure 3: Illustrations for Property 2.8 and the initial state.

Lemma 2.10. *Suppose that Property 2.8 and 2.9 hold for $\gamma \in \Pi$ at time t. For all distinct $v, v' \in V^t_{\text{necc},\gamma}$,*

$$\{f_u : u \in \text{Anc}(v) \cap V^t_{\text{cond},\gamma}\} \cup \{\bar{f}_v\} \not\subsetneq \{f_u : u \in \text{Anc}(v') \cap V^t_{\text{cond},\gamma}\} \cup \{\bar{f}_{v'}\}.$$

In other words, the conjunctions implied by v and v' are inclusionwise not contained in each other.

Proof. Suppose $v, v' \in V^t_{\text{necc},\gamma}$, respectively representing conjunctions C, C' with $C \subsetneq C'$. Let u be the first common ancestor of v and v'. By Property 2.9, $u \notin \{v, v'\}$, so by Property 2.8, u is a vertex of outdegree 2. Without loss of generality, identify u with $\hat{v}$ in the property, and let v be in the subtree of $\hat{u}_1$ and v' be in the subtree of $\hat{u}_2$. Then $f_{\hat{u}_1} \in C$, so by Property 2.9 and $C \subsetneq C'$ we must have that $v' = \hat{w}_2$. But then $|C'| \leq |C|$, contradicting $C \subsetneq C'$. □

Lemma 2.11. *Suppose that Property 2.6, 2.8 and 2.9 hold for $\gamma \in \Pi$ at time t. For all distinct $v, v' \in V^t_{\text{loose},\gamma}$, $\{f_u : u \in \text{Anc}(v) \cap V^t_{\text{cond},\gamma}\} \not\subsetneq \{f_u : u \in \text{Anc}(v') \cap V^t_{\text{cond},\gamma}\}$. In other words, the conjunctions implied by v and v' are inclusionwise not contained in each other.*

Proof. Suppose $v, v' \in V^t_{\text{loose},\gamma}$, respectively representing conjunctions C, C' with $C \subsetneq C'$. Vertices v and v' are in different subtrees, since they are both leaves due to Property 2.6. Let u be the first common ancestor of v and v', so by Property 2.8, u is a vertex of outdegree two. Without loss of generality, identify u with $\hat{v}$ in the property, and let v be in the subtree of $\hat{u}_1$ and v' in the subtree of $\hat{u}_2$. Then $f_{\hat{u}_1} \in C$. But vertex $\hat{w}_2$ has fixing $\bar{f}_{\hat{u}_1}$, so Property 2.9 yields $f_{\hat{u}_1} \notin C'$. This contradicts $C \subsetneq C'$. □

We now proceed by induction to show that the aforementioned properties hold. To this end, we show that they hold at initialization, and that they are maintained during any step of the algorithm.

Initial state At initialization (i.e., $t = 0$), the index i_γ is set to one for all $\gamma \in \Pi$. Because $=_1$ is a tautology, the empty set is the only minimal i_γ-eq-conjunction for all $\gamma \in \Pi$: $E^0_\gamma = \{\emptyset\}$. Similarly, $\prec_1$ is a contradictory operator, so at initialization the only i_γ-inf-conjunctions are the inf-conjunctions that are incompatible with I_0, I_1. An implication tree that consists of the root vertex that is the parent of a single loose end vertex encodes these 1-eq-conjunctions and 1-inf-conjunctions, and respects all aforementioned properties. This is depicted in Figure 3b.

Selecting the permutation Proposition 2.12 shows some sufficient conditions for completeness of fixings in $\mathcal{X}_\gamma$ for each $\gamma \in \Pi$, expressed in the state of the algorithm.

Proposition 2.12 (Sufficient conditions for completeness)**.** *Consider $\gamma \in \Pi$, let t be some time index, and $\emptyset \notin D^t_\gamma$. Suppose that, if $\{(i, b)\} \in D^t_\gamma$, then $i \in I^t_{1-b}$. Then, the set of fixings (I^t_0, I^t_1) is complete for $x \succeq \gamma(x)$ if* **(P1.)** *$E^t_\gamma = \emptyset$, or* **(P2.)** *$i^t_\gamma > n$, or* **(P3.)** *all of the following: $\emptyset \notin E^t_\gamma$, and $i^t_\gamma \notin I^t_0$, and $\gamma^{-1}(i^t_\gamma) \notin I^t_1$, and $\gamma(i^t_\gamma) > i^t_\gamma$, and $\gamma^{-1}(i^t_\gamma) > i^t_\gamma$.*

The proof of this proposition, and of the other propositions in this section, are given in Appendix A. These conditions can be used in Line 4 of the algorithm, because the conditions of the second sentence are satisfied at initialization, and at the end of an iteration due to the loop of Line 8. Due to the induction hypothesis, Properties 2.4, 2.5 and 2.6 are satisfied, so the following corollary shows the same conditions in terms of the implication trees.

Corollary 2.13 (Sufficient conditions for completeness). *Consider $\gamma \in \Pi$, and let t be some time index. Suppose that the root of $\mathcal{T}_\gamma^t$ has no necessary fixing vertex as child. The set of fixings (I_0^t, I_1^t) is complete for $\gamma \in \Pi$ if* **(C1.)** *There is no loose end vertex in $\mathcal{T}_\gamma^t$, or* **(C2.)** *$i_\gamma^t > n$, or* **(C3.)** *Every rooted path to a loose end vertex contains a conditional fixing vertex, and $i_\gamma^t \notin I_0^t$, and $\gamma^{-1}(i_\gamma^t) \notin I_1^t$, and $\gamma(i_\gamma^t) > i_\gamma^t$, and $\gamma^{-1}(i_\gamma^t) > i_\gamma^t$.*

Proof. Follows from Proposition 2.12 and the encoding of D_γ^t and E_γ^t by $\mathcal{T}_\gamma^t$. □

Moreover, when these sufficient conditions are used by the algorithm, it is guaranteed that the algorithm terminates. Namely, one of the sufficient conditions is $i_\gamma^t > n$, so one can increase the index of a single permutation at most n times. Also, at most n fixings are possible, so the inner loop can only be called a finite number of times, as well.

Index increasing event At Line 5, the index i_γ of a permutation $\gamma \in \Pi$ is increased by one, along with the timestamp t, denoted here by $i_\gamma^{t+1} = i_\gamma^t + 1$. This does not affect the fixing sets or the conjunction sets of other permutations, so also the encoding implication trees remain the same. In order for a conjunction C to be a i_γ^{t+1}-eq-conjunction, it specifically has to be a i_γ^t-eq-conjunction, and therefore the latter conjunctions are the starting point for determining i_γ^{t+1}-eq-conjunctions. If an inclusionwise minimal i_γ^t-eq-conjunction C is an i_γ^{t+1}-inf-conjunction, then no i_γ^{t+1}-eq-conjunctions are derived from C. Otherwise, this conjunction is extended (if needed) with fixings of i_γ^t to zero or $\gamma^{-1}(i_\gamma^t)$ to one, preventing that $x \succ_{i_\gamma^t+1} \gamma(x)$ for some $x \in F(I_0^t, I_1^t)$ that satisfies the conjunction. Note that, this way, a single conjunction of the previous timestep could induce two other eq-conjunctions.

Proposition 2.14 (Updating eq-conjunctions for index increasing event). *Consider an index increasing event for permutation $\gamma \in \Pi$ at time t. Then, for $\delta \in \Pi \setminus \{\gamma\}$, we have $E_\delta^{t+1} = E_\delta^t$, and $E_\gamma^{t+1} = Y \cup Z$, where*

$$Y = \left\{ C : \begin{array}{l} C \in E_\gamma^t, \textit{and} \\ \textit{for all } x \in \mathcal{X}_\gamma^{(i_\gamma^t)} \cap F(I_0^t, I_1^t) \cap V(C) \textit{ holds } x_{i_\gamma^t} \leq \gamma(x)_{i_\gamma^t}, \textit{ and} \\ \textit{there is } x \in \mathcal{X}_\gamma^{(i_\gamma^t)} \cap F(I_0^t, I_1^t) \cap V(C) \textit{ with } x_{i_\gamma^t} = \gamma(x)_{i_\gamma^t} \end{array} \right\}, \textit{ and}$$

$$Z = \left\{ C \cup S : \begin{array}{l} C \in E_\gamma^t, S \in \{\{(i_\gamma^t, 0)\}, \{(\gamma^{-1}(i_\gamma^t), 1)\}\}, \\ \mathcal{X}_\gamma^{(i_\gamma^{t+1})} \cap F(I_0^{t+1}, I_1^{t+1}) \cap V(C \cup S) \neq \emptyset, \textit{ and} \\ \textit{there is } x \in \mathcal{X}_\gamma^{(i_\gamma^t)} \cap F(I_0^t, I_1^t) \cap V(C) \textit{ with } x_{i_\gamma^t} > \gamma(x)_{i_\gamma^t} \end{array} \right\}.$$

Regarding the i_γ^{t+1}-inf-conjunctions, all i_γ^t-inf-conjunctions are again valid inf-conjunctions. Added to this, new inf-conjunctions have equality up to i_γ^t, and $x_{i_\gamma^t} < \gamma(x)_{i_\gamma^t}$. The latter are constructed from the i_γ^t-eq-conjunctions, which are (if needed) extended with fixings of i_γ^t to zero and $\gamma^{-1}(i_\gamma^t)$ to one to enforce infeasibility at entry i_γ^t. Again, in both cases one is interested in inclusionwise minimal conjunctions, so non-minimal conjunctions are removed.

Proposition 2.15 (Updating inf-conjunctions for index increasing event). *Consider an index increasing event for permutation $\gamma \in \Pi$ at time t. Then, for $\delta \in \Pi \setminus \{\gamma\}$, we have $D_\delta^{t+1} = D_\delta^t$, and $D_\gamma^{t+1} = D_{\gamma,\mathrm{eq}}^{t+1} \cup D_{\gamma,\mathrm{inf}}^{t+1}$ with*

$$D_{\gamma,\mathrm{eq}}^{t+1} = \left\{ C \cup S : \begin{array}{l} C \in E_\gamma^t,\ S \subseteq \{(i_\gamma^t, 0), (\gamma^{-1}(i_\gamma^t), 1)\},\ \gamma(i_\gamma^t) \neq i_\gamma^t, \textit{ and} \\ \textit{either } (i_\gamma^t, 0) \in S \textit{ or } x_{i_\gamma^t} = 0 \textit{ for all } x \in \mathcal{X}_\gamma^{(i_\gamma^t)} \cap F^t \cap V(C), \textit{ and} \\ \textit{either } (\gamma^{-1}(i_\gamma^t), 1) \in S \textit{ or } \gamma(x)_{i_\gamma^t} = 1 \textit{ for all } x \in \mathcal{X}_\gamma^{(i_\gamma^t)} \cap F^t \cap V(C) \end{array} \right\},$$

with $F^t := F(I_0^t, I_1^t)$, and $D_{\gamma,\mathrm{inf}}^{t+1} = \{C \in D_\gamma^t : \textit{for all } C' \in D_{\gamma,\mathrm{eq}}^t \textit{ holds } C \not\supseteq C'\}$.

These propositions allow us to efficiently update the implication trees in case of an index increasing event. The main idea of the implication tree update stems from the observation that all newly introduced conjunctions are based on some i^t_γ-eq-conjunction that gets extended with zero, one or two fixings to find the new conjunctions. These i^t_γ-eq-conjunctions correspond to the loose end vertices in $\mathcal{T}^t_\gamma$, by the induction hypothesis. We construct $\mathcal{T}^{t+1}_\gamma$ by replacing those loose end vertices with a subtree that encodes the new i^{t+1}_γ-inf-conjunctions and i^{t+1}_γ-eq-conjunctions. It is also possible that the tree structure of a subtree containing this loose end vertex changes, to account for the i^t_γ-inf-conjunctions that are setwise dominated by new i^{t+1}_γ-inf-conjunctions. In the special case that i^t_γ is a fixed point of γ, then these updates correspond to $D^{t+1}_\gamma = D^t_\gamma$ and $E^{t+1}_\gamma = E^t_\gamma$. Hence, the implication tree structure remains unchanged in this case. In the following, we therefore assume that i^t_γ is no fixed point of γ.

Consider a loose end vertex $v_\ell \in V^t_{\text{loose},\gamma}$. Its associated minimal i^t_γ-eq-conjunction corresponds to the union of the fixings of the conditional ancestors of v_ℓ (Property 2.4). To ensure that constraint $x \succeq_{i^t_\gamma} \gamma(x)$ is satisfied if these fixings are applied, also all fixings of the necessary fixing vertices on the rooted path to v_ℓ must be applied. That is, because otherwise infeasibility is yielded by the i^t_γ-inf-conjunctions associated with the necessary fixing vertices on the rooted path to v_ℓ (Property 2.5). These fixings are also sufficient to ensure feasibility. Namely, consider a necessary fixing vertex v_n that does not lie on the rooted path to v_ℓ, and let v_c be the first common ancestor of v_n and v_ℓ. Also, let $u_{c,n}$ be the last non-common ancestor of v_c, and $u_{c,\ell}$ the last non-common ancestor of v_ℓ. Then v_c corresponds to $\hat{v}$ in Property 2.8, and without loss of generality $u_{c,n}$ corresponds to $\hat{u}_1$, and $u_{c,\ell}$ to $\hat{u}_2$. Since all fixings on the rooted path to v_ℓ are applied, Property 2.8 yields that $\bar{f}_{\hat{u}_1}$ is applied, since $f_{\hat{u}_1}$ is in the i^t_γ-inf-conjunction associated with v_n. Hence, no infeasibility will be induced by any necessary fixing vertex that does not lie on the rooted path to v_ℓ.

This result allows for phrasing the updates of Proposition 2.14 and 2.15 in terms of the implication tree. Recall that $C_v \coloneqq \{f_u : u \in \operatorname{Anc}(v) \cap (V^t_{\text{necc},\gamma} \cup V^t_{\text{cond},\gamma})\}$ is the conjunction of all fixings found on the path from the root r_γ to v. The previous paragraph yields:

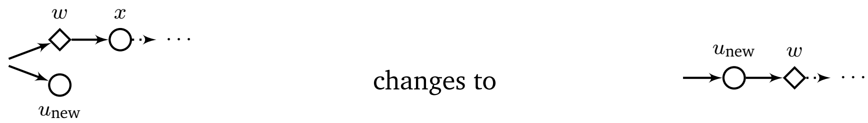
Observation 2.16. *For $v \in V^t_{\text{loose},\gamma}$, with associated i^t_γ-eq-conjunction $C \in E^t_\gamma$, we have $\mathcal{X}^{(i^t_\gamma)}_\gamma \cap F(I^t_0, I^t_1) \cap V(C) = F(I^t_0, I^t_1) \cap V(C_v)$.*

Let $h^t \colon [n] \times V^t_{\text{loose},\gamma} \to \{0, 1, \_\}$ be the value of the fixing of i at time t when respecting the fixings given by I^t_0, I^t_1 and by C_v. If entry i is not fixed, this is denoted by _:

$$h^t(i,v) \coloneqq \begin{cases} 0, & \text{if } i \in I^t_0 \text{ or } (i,0) \in C_v, \\ 1, & \text{if } i \in I^t_1 \text{ or } (i,1) \in C_v, \\ \_, & \text{if } i \notin I^t_0 \cup I^t_1 \text{ and } i \notin \operatorname{Ent}(C_v). \end{cases} \tag{1}$$

Note that is well-defined due to Property 2.9. The implication tree $\mathcal{T}^{t+1}_\gamma$ is found by applying an update to $\mathcal{T}^t_\gamma$ for each loose end vertex $v \in V^t_{\text{loose},\gamma}$, and let $C \in E^t_\gamma$ be the associated i^t_γ-eq-conjunction. By Property 2.6, v is always a leaf of the tree. For compactness, denote $(i,j) \coloneqq (i^t_\gamma, \gamma^{-1}(i^t_\gamma))$, and let $(\alpha,\beta) \coloneqq (h^t(i,v), h^t(j,v))$.

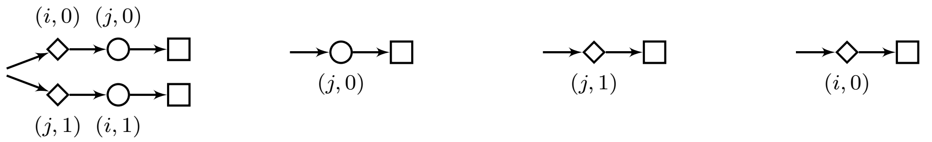
We discuss the updates for different tuples (α,β) in turn. To this end, we need to show Properties 2.4–2.9. We start with Properties 2.6–2.9. On the one hand, if $(\alpha,\beta) = (0,1)$, then Observation 2.16 yields $x_{i^t_\gamma} = 0$ and $\gamma(x)_{i^t_\gamma} = 1$ for all $x \in \mathcal{X}^{(i^t_\gamma)}_\gamma \cap F(I^t_0, I^t_1) \cap V(C)$. The following update is applied. Let u be the first ancestor of v that is a conditional fixing vertex. If u does not exist, then $\emptyset \in D^{t+1}_\gamma$ and we mark the tree as infeasible, so that this situation is handled by Line 7 next. Otherwise, all necessary fixing vertices w in the subtree of u with associated i^t_γ-inf-conjunctions C' have $C' = C \cup \{\bar{f}_w\} \supsetneq C$, so these must be removed. Also loose end vertex v must be removed, since it is no i^{t+1}_γ-eq-conjunction because of (α,β). Hence, replace the subtree rooted at u with a single necessary fixing vertex u_{new} with $f_{u_{\text{new}}} = \bar{f}_u$. A *merging step* is applied if u has a sibling w. That is, the child x of w is removed and w becomes the parent of the children of x, and the parent of w is changed to v. Figure 4 depicts the merging step.



(a) Before merging step

(b) After merging step

Figure 4: Depiction of merging step.



(a) $(\alpha, \beta) = (\_, \_)$ **(b)** $(\alpha, \beta) = (0, \_)$ **(c)** $(\alpha, \beta) = (1, \_)$ **(d)** $(\alpha, \beta) = (\_, 0)$

(e) $(\alpha, \beta) = (0, 0)$ **(f)** $(\alpha, \beta) = (1, 0)$ **(g)** $(\alpha, \beta) = (\_, 1)$ **(h)** $(\alpha, \beta) = (1, 1)$

Figure 5: Possible subtrees to replace a loose end vertex v for (α, β).

These operations remove all vertices from the subtree of u. That is consistent, since u_{new} has C as associated i_γ^{t+1}-inf-conjunction, which dominates the former i_γ^t-inf-conjunctions associated to the necessary fixing vertices in the subtree of u. In the merging step, the removal of x is needed since Property 2.8 yields $f_{u_{\text{new}}} = \bar{f}_u = f_x$, meaning that the inf-conjunction that x encodes is $C \cup \{f_w\} \supsetneq C$. Moreover, no i_γ^t-inf-conjunctions C' with $C' \supsetneq C$ are missed, since by Property 2.8 and 2.9 the necessary fixing vertices associated to C' must be in the rooted path to u_{new}, or in the subtree of w. Last, Properties 2.6, 2.7, 2.8, and 2.9 are maintained, because they are not influenced by a subtree removal or by the merging step.

On the other hand, if $(\alpha, \beta) \neq (0, 1)$, then the loose end vertex v is replaced by a subtree depending on the value of (α, β). The possible new subtrees are listed in Figure 5. These subtrees replace vertex v, so they are connected to its parent.

Observe that loose end vertices are only at the end of the tree, so the update maintains Property 2.6. Due to Property 2.7, any loose end vertex $u \in V_{\text{loose},\gamma}^t$ has $\text{Ent}(C_u) = \text{Ent}(C_v)$, and $h^t(i', u) = \_$ for $i' \in \{i, j\}$ if and only if $i' \notin \text{Ent}(C_v) \cup (I_0^t \cup I_1^t)$. Note that all subtrees in Figure 5 contain exactly these fixing vertices with $h^t(i', v) = \_$ for $i' \in \{i, j\}$, which shows that Property 2.7 is maintained. Property 2.8 holds after the update, since only the tree of Figure 5a that introduces a vertex with outdegree larger than one satisfies this property. Also, Property 2.9 is maintained, since only fixing vertices with fixing (k, b) are introduced with $h^t(k, v) = \_$, this means that $k \notin I_0^t \cup I_1^t$, and k is not an entry of any fixing on C_v.

Concluding, in both cases $(\alpha, \beta) = (0, 1)$ and $(\alpha, \beta) \neq (0, 1)$, Properties 2.6–2.9 are maintained. Last we show that the Properties 2.4 and 2.5 are also maintained, by showing that the tree updates correspond to the elements that C yields in Proposition 2.14 and 2.15. Consider $(\alpha, \beta) = (\_, \_)$. Substituting the expression of Observation 2.16 in Proposition 2.14 yields that the elements E_γ^{t+1} derived from C are $C \cup \{(i, 0)\}$ and $C \cup \{(j, 1)\}$, and these correspond to the conjunctions derived from the new loose end vertices such as in Figure 5a. Likewise, substituting in Proposition 2.15 yields that the only element derived from C in D_γ^{t+1} is $C \cup \{(i, 0), (j, 1)\}$, and this is represented by both necessary fixing vertices in Figure 5a. The new i_γ^{t+1}-inf-conjunctions cannot be setwise contained in an old i_γ^t-conjunction, because no i_γ^t-inf-conjunction containing entries i or j exists. This shows that the update is consistent for $(\alpha, \beta) = (\_, \_)$, and the other cases with $(\alpha, \beta) \neq (0, 1)$ are analogous.

In the same way, for $(\alpha, \beta) = (0, 1)$, the conjunction C does not derive any elements in E_γ^{t+1}. Instead, C is added as element to D_γ^{t+1}. This is consistent with the update, since the update for $(\alpha, \beta) = (0, 1)$ introduces a necessary fixing vertex whose conjunction corresponds to C. Since all conjunctions in D_γ^{t+1} and E_γ^{t+1} are represented by $\mathcal{T}_\gamma^{t+1}$, Lemma 2.10 and 2.11 show that no non-minimal conjunctions are contained, so Property 2.4 and 2.5 are maintained.

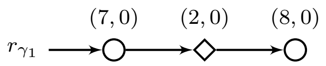


Figure 6: Implication tree of permutation γ_1 in Example 2.2 with $i_{\gamma_1} = 7$ and $x_1 = 1$.

Figure 2 shows the implication tree of Example 2.2 with $i_{\gamma_1} = 6$. We execute an index increasing event. There is a single loose end vertex, and this has $(\alpha, \beta) = (0, 1)$. Thus, if fixing $(7, 1)$ is applied, then an infeasible situation is found. Hence, this conditional fixing vertex turns in a necessary fixing vertex with fixing $(7, 0)$. We also apply a merging step to find the implication tree as in Figure 6.

Selecting a fixing Line 8 searches for a permutation $\gamma \in \Pi$ that has $\{(i, b)\}$ as a i_γ-inf-conjunction, and $i \notin I^t_{1-b}$. This is a minimal inf-conjunction, as otherwise infeasibility was detected at Line 7, so it corresponds, by definition, with $\{(i, b)\} \in D^t_\gamma$ and $i \notin I^t_{1-b}$. Recall that by Property 2.5 such sets correspond to necessary fixing vertices in the implication trees that lie on a path from the root without any conditional fixing vertices on it, and that this correspondence works both ways. Because loose end vertices only occur as leaves (Property 2.6), such paths only contain the root vertex and other necessary fixing vertices. Hence, as a result of the choice of our implication tree data structure, if the root vertex has a necessary fixing vertex as child, then the fixing associated with that fixing vertex has to be applied for completeness. Moreover, if no such vertex exists, also no corresponding conjunction in D^t_γ exists.

Variable fixing event Line 9 applies a fixing $f = (i, b)$. In conjunctions containing this fixing, the fixing can be removed. Likewise, conjunctions containing the converse fixing can impossibly hold, and need to be removed. Last, to ensure inclusionwise minimality, non-minimal elements need to be removed, as well. The following propositions describe the update for the conjunction-sets.

Proposition 2.17 (Updating inf-conjunctions for variable fixing event)**.** *Consider a variable fixing event for fixing $f = (i, b) \in [n] \times \{0, 1\}$ at time t. For every $\gamma \in \Pi$ holds*

$$D^{t+1}_\gamma = \left\{ C \setminus \{f\} : \begin{array}{l} C \in D^t_\gamma, \textit{and} \\ \textit{for all } C' \in D^t_\gamma \textit{ holds } C' \setminus \{f\} \not\subsetneq C \setminus \{f\} \end{array} \right\}. \tag{2}$$

Proposition 2.18 (Updating eq-conjunctions for variable fixing event)**.** *Consider a variable fixing event for fixing $f = (i, b) \in [n] \times \{0, 1\}$ at time t. For every $\gamma \in \Pi$ holds*

$$E^{t+1}_\gamma = \left\{ C \setminus \{f\} : \begin{array}{c} C \in E^t_\gamma, \textit{and} \\ \textit{for all } C' \in E^t_\gamma \textit{ holds } C' \setminus \{f\} \not\subsetneq C \setminus \{f\}, \textit{ and} \\ \textit{for all } C' \in D^t_\gamma \textit{ holds } C' \setminus \{f\} \not\subsetneq C \setminus \{f\} \end{array} \right\}. \tag{3}$$

In terms of the implication trees, for each $\gamma \in \Pi$, the tree $\mathcal{T}^{t+1}_\gamma$ is created from $\mathcal{T}^t_\gamma$ by a sequence of updates. For all fixing vertices $v \in V_{\text{cond},\gamma} \cup V_{\text{necc},\gamma}$ with $\text{Ent}(f_v) = i$, apply one of the following updates: ***(i)*** If $f = f_v$, remove the vertex and connect its parent to its children. If, additionally, v has a sibling, then also remove the subtree rooted at the sibling. ***(ii)*** Or, if $f = \bar{f}_v$ and v is a conditional fixing vertex, remove the vertex and its descendants. ***(iii)*** Otherwise, if $f = \bar{f}_v$ and v is a necessary fixing vertex, then get the first ancestor u of v that is a conditional fixing vertex. If u does not exist, then the tree is marked as infeasible, such that Line 7 returns infeasibility directly after this update. If u does exist, replace u by a necessary fixing vertex with associated fixing $\bar{f}_u$, and remove all its proper descendants. A *merging step* is applied if u has a sibling w. This merging step is identical to the merging step at the index increasing event.

For example, after the index increasing event yielding Figure 6, an index increasing event is called for fixing $(7, 0)$. Entry 7 is thus added to I^t_0, and the tree of the figure is updated such that the necessary fixing node of fixing $(7, 0)$ is removed, and the root is connected to the conditional fixing vertex with fixing $(2, 0)$.

We argue that the tree properties are maintained. The updates remove whole subtrees, remove all fixing vertices with entry i, and apply merging steps. If a leaf vertex u is not removed at a merging step, then the only difference of the set C_u before and after any merging step is the removal of all fixings with entry i. The loose end vertices are either removed, or remain leaves of the tree, so Property 2.6 is maintained. All fixing vertices with entry i are removed, and $I_0^{t+1} \cup I_1^{t+1} = I_0^t \cup I_1^t \cup \{i\}$, so Properties 2.7 and 2.9 are maintained.

To show Property 2.8, note that no tree update can increase the degree of a vertex, so it suffices to check that the property is maintained for each vertex with outdegree two. Let v be a vertex with outdegree larger than one before an update, and let $\hat{u}_1, \hat{u}_2, \hat{w}_1, \hat{w}_2$ be vertices as in Figure 3a, with $\hat{v}$ as v. If $f_{\hat{u}_1} = f$, then either update *(i)* removes the subtree rooted in $\hat{u}_2$, or update *(iii)* is executed for due to $\hat{w}_2$. Which of the two updates applies depends on the order in which the vertices are processed, but the same result is yielded. If $\bar{f}_{\hat{u}_1} = f$, then update *(ii)* removes the subtree rooted in u_1. The case for $f_{\hat{u}_2} = f$ and $\bar{f}_{\hat{u}_2} = f$ is analogous. In these cases, after the update the degree of v is one, so the property holds. If both $f_{\hat{u}_1}$ and $f_{\hat{u}_2}$ are neither f nor $\bar{f}$, then either the substructure (of v and its predecessors at maximal distance 2) of Figure 3a is maintained, or *(iii)* yields that a merging step is applied that changes the degree of v to one.

Last we argue that Properties 2.4 and 2.5 are maintained after applying all updates.

Let $v \in V^t_{\text{cond},\gamma}$ be a conditional fixing vertex to which an update is applied. On the one hand, if $f_v = f$, then all necessary fixing vertices in its subtree induce a conjunction C with $f \in C$. By removing vertex v and reconnecting its parent to its children, these conjunctions are updated to $C \setminus \{f\}$, consistent with Proposition 2.15. The same holds for conjunctions represented by loose end vertices in the subtree. Additionally, if v has a sibling v', then by Property 2.8 vertex v' is a conditional fixing vertex, and the first child u of v is a necessary fixing vertex with $f_u = \bar{f}_{v'}$. Hence, if all conditional fixings up to v are applied, then f_u must be applied as well, so it is not possible to apply the fixing associated with v'. Hence, the subtree rooted in v' can be removed, since it does not contain any minimal conjunction for time $t+1$. On the other hand, if $f = \bar{f}_v$, then all conjunctions C induced by necessary fixing vertices u have $\bar{f}_v \in C$. However, $C' = \{f, \bar{f}\} \in D^t_\gamma$, so $C' \setminus \{f\} \subsetneq C \setminus \{f\}$. Likewise, for loose end vertices in the subtree of v that induce conjunction C has $C' \setminus \{f\} \subseteq C \setminus \{f\}$. In both cases, vertices in this subtree induce no inclusionwise minimal conjunctions, so this subtree can be removed.

Let $v \in V^t_{\text{necc},\gamma}$ be a necessary fixing vertex to which an update is applied, denote its associated conjunction by C. On the one hand, if $f = f_v$, then $\bar{f} \in C$ and update *(i)* is applied. Because fixing f is applied, conjunction C can never be violated any more, so v can be removed. Vertex v cannot have siblings due to Property 2.8, so this is all the update does. On the other hand, if $f = \bar{f}_v$, then $f \in C$ and update *(iii)* is applied. Let u be the first ancestor of v that is either a conditional fixing vertex, or the root vertex. If u is the root vertex, then $\{f\}$ is an inf-conjunction, so applying f yields infeasibility for the permutation. This is correctly marked. Otherwise, if u is a conditional fixing vertex, then any necessary fixing vertex in the subtree of u with associated conjunction C' has $C \setminus \{f\} \subsetneq C' \setminus \{f\} = C'$, and any loose end vertex in the subtree of u with associated conjunction C' has $C \setminus \{f\} \subseteq C' \setminus \{f\} = C'$. Since $C \setminus \{f\}$ is a i^{t+1}_γ-inf-conjunction, all these vertices can be removed. In terms of the tree update, this is phrased as: one may not apply all conditional fixings on the path to u, and f_u itself. Thus, we can replace the subtree of u by a necessary fixing vertex u_{new} with $f_{u_{\text{new}}} = \bar{f}_u$, such that the conjunction associated with this necessary fixing vertex corresponds to $C \setminus \{f\}$. If u has a sibling u', then a merging step is applied that removes the child w' of u'. By Proposition 2.8 w' is a necessary fixing vertex with $f_{w'} = \bar{f}_u = f_{u_{\text{new}}}$ as associated fixing. Since $C \setminus \{f\}$ is the conjunction associated with u_{new}, and because the conjunction associated with w' is inclusionwise larger than in $C \setminus \{f\}$, this vertex w' is correctly removed.

To summarize, all loose end vertices or necessary fixing vertices that induce a conjunction C represent $C \setminus \{f\}$. These are all correct i^{t+1}_γ-inf-conjunctions or i^{t+1}_γ-eq-conjunctions. Only non-minimal conjunctions of its type are removed, and all conjunctions in D^{t+1}_γ and E^{t+1}_γ are generated from conjunctions in D^t_γ and E^t_γ, respectively. Together with Lemma 2.10 and 2.11, no other conjunctions can be represented by $\mathcal{T}^{t+1}_\gamma$, therewith proving Property 2.4 and 2.5.

Running time analysis A timestamp t is used in the algorithm description, but this is only required for the analysis, and not for the implementation. Hence, it suffices to maintain the last state only. The index sets of the fixings (I_0^t, I_1^t) are implemented by two Boolean arrays of size n specifying if an entry i is included in the index set or not. For each permutation $\gamma \in \Pi$, we maintain the tree data structure and the index i_γ.

In the following, we work with the implication tree representation and suppose that the tree updates are applied as described in the previous paragraphs, and that the sufficient conditions of completeness from Corollary 2.13 are used as selection condition.

Consider $\gamma \in \Pi$ and time t, and suppose there exist vertices with outdegree larger than one. Then Property 2.8 yields that any rooted path to a leaf contains at least a conditional fixing vertex. The only way that a new vertex with minimal outdegree two is introduced by the algorithm is during an index increasing event for γ where i_γ^t is no fixed point and $(\alpha, \beta) = (\_, \_)$, which by Property 2.7 yields $i_\gamma^t, \gamma^{-1}(i_\gamma^t) \notin I_0^t \cup I_1^t$ and $i_\gamma^t, \gamma^{-1}(i_\gamma^t) \notin \bigcup_{i' \in [i_\gamma^t - 1]} \{i', \gamma^{-1}(i')\} \setminus (I_0^t \cup I_1^t)$. In particular, this shows that for all $i' < i_\gamma^t$ we have $\gamma^{-1}(i') \neq i_\gamma^t$, i.e. $i' \neq \gamma(i_\gamma^t)$, implying that $\gamma(i_\gamma^t) \geq i_\gamma^t$. Since i_γ^t is no fixed point of γ, $\gamma(i_\gamma^t) > i_\gamma^t$ follows. Also, $\gamma^{-1}(i_\gamma^t) > i_\gamma^t$ follows since i_γ^t is not a fixed point, and $\gamma^{-1}(i_\gamma^t) \neq i'$ for any $i' < i_\gamma^t$. Concluding, all conditions of *(C3)* of Corollary 2.13 hold, meaning that sufficient conditions for completeness of γ are met. Hence, the algorithm will never introduce a second vertex with minimal outdegree two. This means that the implication tree can be encoded by two paths. Moreover, because of Property 2.9 we can encode these two paths as two separate arrays of length n.

As a result, one only needs to store the root vertex, and for each of the two possible branches one vertex per entry for the fixing vertices, and a loose end vertex. To allow for constant-time lookups and insertions of the fixing vertices where the entry has a certain fixing, two arrays of size n are maintained, in which at entry i (necessary or conditional) fixing vertices are stored whose fixings have entry i in either path.

Furthermore, the algorithm maintains $\mathcal{Q}_F$, which consists of a subset of fixings that must be applied as a result of Line 8 of the algorithm. Specifically, this set is empty only if the condition of Line 8 is not met, which means that valid fixings can never be missed. Each time an implication tree for a permutation is updated, it is checked if the root has a necessary fixing vertex as child. If that is the case, then this fixing is added to $\mathcal{Q}_F$, if not already there. This is a constant-time check, since the root has at most two children due to Property 2.8. By implementing $\mathcal{Q}_F$ as a stack and a Boolean direct-lookup array, pushing specific elements to the set, popping the lastly added element from the set, and checking if an element is contained are constant-time operations. The while-loop of Line 8 corresponds to iteratively applying virtual fixings popped from $\mathcal{Q}_F$ until it is empty.

Likewise, the set $\mathcal{Q}_\Pi$ is the set of permutations for which the sufficient conditions of completeness of Corollary 2.13 do not hold. These conditions are checked for a permutation $\gamma \in \Pi$ if the index i_γ increases, or if a fixing is applied. The conditions can be checked in constant time: For *(C1)* there are at most two leaves since there is at most one vertex with outdegree two and loose end vertices are always a leaf of the tree; and for *(C3)* this corresponds with checking if the root vertex only has conditional fixing vertices as child, of which at most two could exist due to Property 2.8.

At initialization, consistent with the initial state of the algorithm, $\mathcal{Q}_F$ is empty and $\mathcal{Q}_\Pi$ consists of all permutations.

Since loose end vertices are leaves, and the tree only has at most a single vertex with outdegree 2 (and the rest having smaller outdegree), implication trees have at most two loose end vertices. At an index increasing event the loose end vertices are replaced by subtrees having at most six vertices. Hence, a constant number of vertices is introduced at an index increasing event. Introduced vertices can be removed at most once, and (at a merging step) changed from a conditional fixing vertex type to a necessary fixing type. Transforming the vertex type back is no option, so this can happen at most once, as well. This means that tree updates at the index increasing or variable fixing events take amortized constant time.

As one of the sufficient conditions of completeness is that $i_\gamma^t > n$, and i_γ^t increases by one for every index increasing event and never decreases, the index increasing events can only be called n times per permutation, so that is at most $n|\Pi|$ times in total. Likewise, at most n different fixings can be applied before infeasibility is found, so at most $n + 1$ variable fixing events can occur at most.

Also, the bottleneck of initialization is the preparation of the constant number of arrays of size n for each permutation. Hence, collecting all observations above, the running time of this algorithm is $\mathcal{O}(n|\Pi|)$. Specifically, with this algorithm specification the bound is tight, as this corresponds to the space requirement at initialization.

3 Propagation of Lexicographic Orders for Cyclic Groups

The propagation algorithm of the previous section ensures that solutions are lexicographically not smaller than their image with respect to the permutations in a set $\Pi \subseteq \mathcal{S}_n$. This is enforced by a propagation loop, which determines valid variable fixings for each individual constraint $x \succeq \gamma(x)$, for each $\gamma \in \Pi$. The derived set of fixings, however, is not necessarily complete, as illustrated next:

Example 3.1. Let $\tilde{\gamma} = (1, 2, 3, 4, 5)$ and $\Pi = \langle\tilde{\gamma}\rangle$. Consider the set of fixings $I_0 = \{2, 5\}$ and $I_1 = \emptyset$. The set of vectors that are feasible for all permutations are $\mathcal{X}_\Pi \cap F(I_0, I_1) = \{(0, 0, 0, 0, 0), (1, 0, 0, 0, 0), (1, 0, 1, 0, 0)\}$, which means that the fourth entry is a zero-fixing in $\mathcal{X}_\Pi \cap F(I_0, I_1)$. However, this entry cannot be detected by the propagation loop, as $(1, 0, 1, 1, 0) \in \mathcal{X}_\delta$ for $\delta \in \{\gamma^1, \gamma^2, \gamma^4\}$, and $(1, 0, 0, 1, 0) \in \mathcal{X}_{\gamma^3}$. This shows that I_0 and I_1 cannot be extended if we consider the permutations of Π independently.

Consequently, there are variable fixings that cannot be detected by treating the elements in a set of permutations Π individually. In particular, this is also not the case if Π forms a cyclic group. The aim of this section is to close this gap for cyclic groups, i.e., devising algorithms finding the complete set of fixings for different types of cyclic groups.

If all entries are fixed initially, a complete propagation algorithm for lexicographic leaders in a group reduces to determine if the single possible vector adhering to the fixings is the lexicographic leader in its Γ-orbit. Babai and Luks [1] show that this is coNP-complete. This means one cannot expect to find a polynomial-time algorithm to find the complete set of fixings of lexicographic leaders in the Γ-orbits, for general symmetry groups $\Gamma \leq \mathcal{S}_n$, unless P $=$ coNP. Luks and Roy [24] also show that, although the problem of testing elements for being lexicographically maximal in their group orbits is coNP-complete, for Abelian groups there always exists an ordering of the group support such that this problem becomes polynomial. Roy [33] gives a generalization of this result for a wider class of groups.

In the following, we provide two propagation algorithms. First, we consider one for groups generated by a single cycle that has a *monotone representation*, or subgroups of such groups. After that, groups are considered that have a generator with a monotone and *ordered* representation. What monotone and ordered representation embraces is made precise below. Although these requirements do not allow to handle all cyclic groups, we will discuss in Section 4 how our algorithms can still be used to handle symmetries in arbitrary binary programs. That is, we describe how the technical assumptions on the groups can be guaranteed for every binary program.

3.1 Cyclic Group Generated by Monotone Cycle or Subgroup Hereof

Let $\Gamma \leq \mathcal{S}_n$ and two disjoint sets $I_0, I_1 \subseteq [n]$ defining initial fixings. To find the complete set of fixings for lexicographic leaders in the Γ-orbit, one needs to determine the maximal sets $J_0, J_1 \subseteq [n]$ such that $\mathcal{X}_\Gamma \cap F(I_0, I_1) = \mathcal{X}_\Gamma \cap F(J_0, J_1)$. The simple Algorithm 2 shows how to find the complete set of fixings for lexicographic leaders in the Γ-orbit. Starting with the initial set of fixings (I_0, I_1), the algorithm extends this set with all necessary fixings for completeness for the constraint $x \succeq \gamma(x)$ for all $\gamma \in \Gamma$, and then checks for each remaining non-fixed entry if any lexicographically maximal vector exists in its Γ-orbit where the non-fixed entry is fixed to zero or to one. To simplify the analysis of Algorithm 2, we maintain a timestamp t, starting at 0, and the set of fixings at time t is denoted by I_0^t and I_1^t.

Algorithm 2: Find the complete set of fixings of $\mathcal{X}_\Gamma \cap F(I_0, I_1)$ with $\Gamma \leq \mathcal{S}_n$

input : group $\Gamma \leq \mathcal{S}_n$, sets $I_0, I_1 \subseteq [n]$
output: the message INFEASIBLE, or FEASIBLE and two subsets of $[n]$
1 **if** $\mathcal{X}_\Gamma \cap F(I_0, I_1) = \emptyset$ **then return** INFEASIBLE;
2 $t \leftarrow 0$; $(I_0^t, I_1^t) \leftarrow$ the complete set of fixings for each individual constraint $x \succeq \gamma(x)$ for all $\gamma \in \Gamma$;
3 **foreach** $i \in [n] \setminus (I_0^0 \cup I_1^0)$ **do**
4   **if** $\mathcal{X}_\Gamma \cap F(I_0^t \cup \{i\}, I_1^t) = \emptyset$ **then** $(I_0^{t+1}, I_1^{t+1}) \leftarrow (I_0^t, I_1^t \cup \{i\})$; $t \leftarrow t + 1$;
5   **if** $\mathcal{X}_\Gamma \cap F(I_0^t, I_1^t \cup \{i\}) = \emptyset$ **then** $(I_0^{t+1}, I_1^{t+1}) \leftarrow (I_0^t \cup \{i\}, I_1^t)$; $t \leftarrow t + 1$;
6 **return** FEASIBLE, (I_0^t, I_1^t);

Fixings valid for each individual constraint $x \succeq \gamma(x)$ for $\gamma \in \Gamma$ are also valid fixings in $\mathcal{X}_\Gamma \cap F(I_0, I_1)$, so the set (I_0^t, I_1^t) after Line 2 is a valid set of fixings for $\mathcal{X}_\Gamma \cap F(I_0, I_1)$. Also, since $\mathcal{X}_\Gamma \cap F(I_0^t, I_1^t) = (\mathcal{X}_\Gamma \cap F(I_0^t \cup \{i\}, I_1^t)) \dot{\cup} (\mathcal{X}_\Gamma \cap F(I_0^t, I_1^t \cup \{i\}))$, the updates of Line 4 and 5 ensure that $\mathcal{X}_\Gamma \cap F(I_0^t, I_1^t) = \mathcal{X}_\Gamma \cap F(I_0, I_1)$ for all $t \geq 0$. Hence, the algorithm never applies incorrect fixings. If no fixing is applied in an iteration for index i, then $\mathcal{X}_\Gamma \cap F(I_0^t \cup \{i\}, I_1^t) \neq \emptyset$ and $\mathcal{X}_\Gamma \cap F(I_0^t, I_1^t \cup \{i\}) \neq \emptyset$ certify the existence of two vectors $x^0, x^1 \in \mathcal{X}_\Gamma \cap F(I_0, I_1)$ with $x_i^0 = 0$ and $x_i^1 = 1$, respectively. The algorithm iterates over all indices $i \in [n]$ with $i \notin I_0^0 \cup I_1^0$, so no valid fixing could be missed. This shows that the algorithm is correct.

In the Lines 1, 4 and 5 it is checked whether a vector exists that is lexicographically maximal in the Γ-orbit, and satisfies certain fixings. Recall the result by Babai and Luks [1], which shows that a restriction of this problem is coNP-complete. Therefore, one cannot expect a polynomial-time realization of this algorithm for general groups, unless P = coNP. In the remainder of this section, we restrict ourselves to groups generated by a *monotone* cycle (or subgroups hereof), for which we show a polynomial-time realization of Algorithm 2. A cycle ζ is called *monotone* if the cycle has exactly one entry i with $\zeta(i) < i$. Let $\Gamma \leq \langle (1, \dots, n) \rangle$. With the following proposition, we argue how the checks of Line 1, 4 and 5 can be applied efficiently.

Proposition 3.2. *Let $I_0, I_1 \subseteq [n]$ be disjoint index sets defining a complete set of fixings for the constraint $x \succeq \gamma(x)$ for each permutation $\gamma \in \Gamma \leq \langle (1, \dots, n) \rangle$. The set $\mathcal{X}_\Gamma \cap F(I_0, I_1)$ is non-empty if and only if the sets $\mathcal{X}_\gamma \cap F(I_0, I_1)$ are non-empty for every permutation $\gamma \in \Gamma$.*

The checks of Line 1, 4 and 5 can be executed using Proposition 3.2, by computing the set of fixings that is complete for each individual constraint $x \succeq \gamma(x)$, for each $\gamma \in \Gamma$. This is possible, for instance, with Algorithm 1 of Section 2 in $\mathcal{O}(n \operatorname{ord}(\Gamma))$ time. As a benefit, the result of that algorithm gives (I_0^0, I_1^0), which satisfies $\mathcal{X}_\Gamma \cap F(I_0, I_1) = \mathcal{X}_\Gamma \cap F(I_0^0, I_1^0)$, as needed in Line 2. There are at most n unfixed entries, so the subroutine that determines the complete set of fixings for each permutation in Γ is called at most $1 + 2n$ times, taking $\mathcal{O}(n \operatorname{ord}(\Gamma))$ time when using Algorithm 1. Hence, the total running time is $\mathcal{O}(n^2 \operatorname{ord}(\Gamma))$. Recall that $\Gamma \leq \langle (1, \dots, n) \rangle$, such that $\operatorname{ord}(\Gamma) \leq n$, so that the running time is polynomially bounded.

Remark 3.3. Fewer calls to the subroutine are required if we can determine a vector $\tilde{x} \in \mathcal{X}_\Gamma \cap F(I_0, I_1)$. If $\tilde{x}_i = 0$, we can avoid checking if $\mathcal{X}_\Gamma \cap F(I_0^t \cup \{i\}, I_1^t)$ is empty or not. The same argument can be used for $\tilde{x}_i = 1$.

Last, for the missing proof of Proposition 3.2, we make repeated use of the following argument:

Remark 3.4. Let $I_0, I_1 \subseteq [n]$ be disjoint index sets defining a complete set of fixings for the constraint $x \succeq \gamma(x)$ for each permutation $\gamma \in \Gamma$. Let $\gamma \in \Gamma$ with $\mathcal{X}_\gamma \cap F(I_0, I_1) \neq \emptyset$, and $k \in [n]$. Suppose that for all $i < k$ we either have $i, \gamma^{-1}(i) \in I_0$ or $i, \gamma^{-1}(i) \in I_1$. This means that for all $x \in F(I_0, I_1)$ and $i < k$ we have $x_i = \gamma(x)_i$, i.e. $x =_k \gamma(x)$. Since the set of fixings is complete with respect to $x \succeq_\gamma (x)$ for γ, and for any $x \in F(I_0, I_1)$ the vectors x and $\gamma(x)$ are identical up to entry k: If $k \in I_0$, then $\gamma^{-1}(k) \in I_0$. Likewise, if $\gamma^{-1}(k) \in I_1$, then $k \in I_1$.

Proof of Proposition 3.2. Let $I_0, I_1 \subseteq [n]$ be disjoint sets that define for each $\gamma \in \Gamma$ a complete set of fixings for $\mathcal{X}_\gamma \cap F(I_0, I_1)$. For brevity, we denote $F := F(I_0, I_1)$. By definition, $\mathcal{X}_\Gamma = \bigcap_{\gamma\in\Gamma} \mathcal{X}_\gamma$. Hence, if there exist $\gamma \in \Gamma$ with $\mathcal{X}_\gamma \cap F = \emptyset$, then $\mathcal{X}_\Gamma \cap F = \emptyset$ as well, proving the first implication. In the remainder of the proof, we show the converse implication. Suppose that $\mathcal{X}_\gamma \cap F$ is non-empty for all permutations $\gamma \in \Gamma$. We now show that there exists a vector $\tilde{x} \in \mathcal{X}_\Gamma \cap F$. First, if $I_0 \cup I_1 = [n]$, there are no unfixed entries. Thus the set F consists of a single element $\tilde{x}$. By assumption, $\mathcal{X}_\gamma \cap F \neq \emptyset$ for every $\gamma \in \Gamma$, so $\tilde{x} \in \mathcal{X}_\gamma$, meaning that $\mathcal{X}_\Gamma \cap F = \bigcap_{\gamma\in\Gamma} \mathcal{X}_\gamma \cap F = \{\tilde{x}\} \neq \emptyset$.

Second, suppose $I_0 \cup I_1 \subsetneq [n]$. Let $\hat{\imath} := \min\{i : i \in [n] \setminus (I_0 \cup I_1)\}$ be the first unfixed entry of the vectors in F, and for any $\gamma \in \Gamma$ let $\check{\imath}_\gamma := \min\{\gamma^{-1}(i) : i \in [n] \setminus (I_0 \cup I_1)\}$ be the first unfixed entry of a vector $\gamma(x)$ for vectors $x \in F$. If the permutation is clear from the context, we drop the subscript of $\check{\imath}_\gamma$. Let $\tilde{x} \in F$ with $\tilde{x}_i = 1$ if $i \in I_1 \cup \{\hat{\imath}\}$, and $\tilde{x}_i = 0$ otherwise. We claim that $\tilde{x} \in \mathcal{X}_\Gamma \cap F$, which completes the proof.

For the sake of contradiction, assume there is a $\gamma \in \Gamma$ with $\tilde{x} \prec \gamma(\tilde{x})$. Then, there is $i \in [n]$ such that $\tilde{x} =_i \gamma(\tilde{x})$ and $\tilde{x}_i = 0 < \gamma(\tilde{x})_i = 1$. Since $\mathcal{X}_\gamma \cap F \neq \emptyset$, we thus have $i \geq \min\{\hat{\imath}, \check{\imath}\}$. That is, $k, \gamma^{-1}(k) \in I_0 \cup I_1$ for all $k < m := \min\{\hat{\imath}, \check{\imath}\}$, and for all $x \in F$ holds that $x =_m \gamma(x)$. Note that therefore $\tilde{x} \succ \gamma(\tilde{x})$ if $\tilde{x}_m = 1$ and $\gamma(\tilde{x})_m = 0$. We will use this observation to find a contradiction in the following.

If $\hat{\imath} \leq \check{\imath}$, then $\gamma^{-1}(\hat{\imath}) \notin I_1$, because otherwise the complete propagation algorithm for $\mathcal{X}_\gamma \cap F$ had fixed $\hat{\imath}$ to 1: see Remark 3.4. Since γ has no fixed points because it is a cyclic shift, $\gamma^{-1}(\hat{\imath}) \neq \hat{\imath}$, so $\gamma(\tilde{x})_{\hat{\imath}} = 0$. This implies $\tilde{x}_{\hat{\imath}} > \gamma(\tilde{x})_{\hat{\imath}}$, contradicting $\tilde{x} \prec \gamma(\tilde{x})$. Hence, $\hat{\imath} > \check{\imath}$, and thus $\check{\imath} \in I_0 \cup I_1$. If $\check{\imath} \in I_0$, then $\gamma^{-1}(\check{\imath})$ is necessarily contained in I_0, as otherwise any $x \in F$ with $\gamma(x)_{\check{\imath}} = 1$ satisfies $x =_{\check{\imath}} \gamma(x)$ and $x_{\check{\imath}} < \gamma(x)_{\check{\imath}}$, which contradicts completeness of the fixings with respect to γ. Thus, in the remainder we have $\hat{\imath} > \check{\imath}$ and $\check{\imath} \in I_1$.

If $\gamma(\tilde{x})_{\check{\imath}} = 0$, then $\tilde{x} \succ \gamma(\tilde{x})$ which contradicts the assumption that $\tilde{x} \prec \gamma(\tilde{x})$. Therefore $\gamma(\tilde{x})_{\check{\imath}} = 1$, which means that $\gamma^{-1}(\check{\imath}) \in I_1 \cup \{\hat{\imath}\}$, and by the definition of $\check{\imath}$ only $\gamma^{-1}(\check{\imath}) = \hat{\imath}$ remains. Let $j := \min\{i \in [n] : \tilde{x}_i \neq \gamma(\tilde{x})_i\}$ be the first entry where the values of $\tilde{x}$ and $\gamma(\tilde{x})$ differ. Then $\tilde{x}_j = 0$ by $\tilde{x} \prec \gamma(\tilde{x})$. It is not possible that $j = \hat{\imath}$, as $\tilde{x}_j = 0 \neq 1 = \tilde{x}_{\hat{\imath}}$. Using the following claim, we show that $j \neq \hat{\imath}$ also yields a contradiction.

Claim. *If $\hat{\imath} > \check{\imath}$, any $x \in \mathcal{X}_\gamma \cap F$ with $x_{\hat{\imath}} = 1$ has $x_i = \tilde{x}_i$ for all $i \leq \min\{n, j + \hat{\imath} - \check{\imath} - 1\}$.*

Proof of claim. Let $x \in \mathcal{X}_\gamma \cap F$ with $x_{\hat{\imath}} = 1$. To prove the claim, we show that $x_i = \tilde{x}_i$ for all $i \leq b$, where $b \leq \min\{n, j+\hat{\imath}-\check{\imath}-1\}$. We proceed by induction on b. Because $x, \tilde{x} \in F$, by $x_{\hat{\imath}} = \tilde{x}_{\hat{\imath}} = 1$ and the definition of $\hat{\imath}$ we have $x_i = \tilde{x}_i$ for all $i \leq \hat{\imath}$, so the statement holds for all $b \leq \hat{\imath}$.

Consider $b \leq \min\{n, j + \hat{\imath} - \check{\imath} - 1\}$ and $b > \hat{\imath}$, and suppose that for all $i \leq b - 1$ we have $x_i = \tilde{x}_i$ **(IH)**. By the previous paragraph, this holds for $b = \hat{\imath} + 1$. We show that $x_b = \tilde{x}_b$, so that the claim follows by induction. If $b \in I_0 \cup I_1$, then $x_b = \tilde{x}_b$ holds trivially because entry b is fixed, so in the remainder we assume $b \notin I_0 \cup I_1$. The following chain of equations holds due to the arguments presented earlier:

$$x_{b-\hat{\imath}+\check{\imath}} \overset{\textbf{(IH)}}{=} \tilde{x}_{b-\hat{\imath}+\check{\imath}} \overset{(b-\hat{\imath}+\check{\imath}<j)}{=} \gamma(\tilde{x})_{b-\hat{\imath}+\check{\imath}} = \tilde{x}_{\gamma^{-1}(b-\hat{\imath}+\check{\imath})} \overset{(*)}{=} \tilde{x}_b. \tag{4}$$

In this, $(*)$ uses that γ is a cyclic shift with $\gamma^{-1}(\check{\imath}) = \hat{\imath}$, such that $\gamma^{-1}(i) = i + \hat{\imath} - \check{\imath}$ for $0 < i \leq n - \hat{\imath} + \check{\imath}$. Observe that $b - \hat{\imath} + \check{\imath} \leq n - \hat{\imath} + \check{\imath}$ due to $b \leq n$, and $b - \hat{\imath} + \check{\imath} > \check{\imath} > 0$ due to $b > \hat{\imath}$.

If $x_{b-\hat{\imath}+\check{\imath}} = 1$, then Equation (4) yields $\tilde{x}_b = 1$. Thus, $b \in I_1 \cup \{\hat{\imath}\}$ by the definition of $\tilde{x}$. As $b > \hat{\imath}$, $b \in I_1$. Consequently, $x_b = \tilde{x}_b = 1$.

Otherwise, if $x_{b-\hat{\imath}+\check{\imath}} = 0$, the induction hypothesis yields $x =_{b-\hat{\imath}+\check{\imath}} \tilde{x}$, as $\hat{\imath} > \check{\imath}$. Also $b < j + \hat{\imath} - \check{\imath}$, so $b - \hat{\imath} + \check{\imath} < j$. Because j is the first entry where $\tilde{x}$ and $\gamma(\tilde{x})$ differ, $\tilde{x} =_{b-\hat{\imath}+\check{\imath}} \gamma(\tilde{x})$. Since $\check{\imath}$ is the first entry where the entries of the vectors after permuting with γ are not fixed, $x, \tilde{x} \in F$ imply $\gamma(x)_i = \gamma(\tilde{x})_i$ for all $i < \check{\imath}$. Last, if $\check{\imath} \leq i < b - \hat{\imath} + \check{\imath} \leq n - \hat{\imath} + \check{\imath}$, the induction hypothesis yields $\gamma(x)_i = x_{i+\hat{\imath}-\check{\imath}} = \tilde{x}_{i+\hat{\imath}-\check{\imath}} = \gamma(\tilde{x})_i$. Combining these results yields $x =_{b-\hat{\imath}+\check{\imath}} \gamma(x)$. Since $x \in \mathcal{X}_\gamma$, this means that $x \succeq \gamma(x)$, and thus from $x =_{b-\hat{\imath}+\check{\imath}} \gamma(x)$ and $x_{b-\hat{\imath}+\check{\imath}} = 0$ follows $\gamma(x)_{b-\hat{\imath}+\check{\imath}} = x_b = 0$. Concluding, Equation (4) yields $0 = x_b = \gamma(x)_{b-\hat{\imath}+\check{\imath}} = x_{b-\hat{\imath}+\check{\imath}} \overset{(4)}{=} \tilde{x}_b$. In each case $x_b = \tilde{x}_b$ is found, so by induction the claim follows. $\diamondsuit$

Recall that $j \neq \hat{\imath} > \check{\imath}$, and that $\gamma^{-1}(\check{\imath}) = \hat{\imath}$. As completeness of the fixing for permutation γ is assumed, and $\hat{\imath} \notin I_0 \cup I_1$, there always exists a vector $x \in \mathcal{X}_\gamma \cap F$ with $x_{\hat{\imath}} = 1$. If the minimum in the claim evaluates to n, then $x = \tilde{x}$, so $\tilde{x} \in \mathcal{X}_\gamma \cap F$, contradicting the assumption that $\tilde{x} \prec \gamma(\tilde{x})$. Otherwise, if the minimum evaluates to $j + \hat{\imath} - \check{\imath} - 1$, then $x =_{j+\hat{\imath}-\check{\imath}} \tilde{x}$, so especially $x_j = \tilde{x}_j = 0$, as a result of $\hat{\imath} > \check{\imath}$. Moreover, because γ is a cyclic shift with $\gamma^{-1}(\check{\imath}) = \hat{\imath}$, $\gamma(x) =_j \gamma(\tilde{x})$. Also, $\gamma(\tilde{x})_j = 1$, so $\gamma^{-1}(j) \in I_1 \cup \{\hat{\imath}\}$. From $\tilde{x}_j = 0 \neq \tilde{x}_{\hat{\imath}} = 1$ follows $\gamma^{-1}(j) \neq \hat{\imath}$, so $\gamma^{-1}(j) \in I_1$. It follows from $\tilde{x} =_j \gamma(\tilde{x})$ and $\hat{\imath} > \check{\imath}$ that $x =_j \tilde{x} =_j \gamma(\tilde{x}) =_j \gamma(x)$, $x_j = 0$ and $\gamma(x)_j = 1$, such that $x \prec \gamma(x)$. This contradicts $x \in \mathcal{X}_\gamma$. So every case with $\tilde{x} \prec \gamma(\tilde{x})$ yields a contradiction, proving that $\tilde{x} \in \mathcal{X}_\gamma$. $\square$

3.2 Cyclic Group Generated by Ordered and Monotone Subcycles

In this section, a complete propagation algorithm for lexicographic leaders of the Γ-orbit is presented, where Γ is generated by a composition $\tilde{\gamma} = \zeta_1 \circ \cdots \circ \zeta_m$ of *monotone* and *ordered* subcycles, possibly of different lengths. That is, for $i \in [m]$, each subcycle ζ_i is monotone, so there is only a single entry k with $\zeta_i(k) < k$. Also, the subcycles are *ordered*: for each $i, j \in [m]$ with $i < j$, all entries in the support of ζ_i are smaller than the entries in the support of ζ_j. If $\tilde{\gamma}$ has no fixed points, one can write $\tilde{\gamma} = (1, \dots, z_1)(z_1 + 1, \dots, z_2) \cdots (z_{m-1} + 1, \dots, z_m)$ for some $1 < z_1 < z_2 < \cdots < z_m = n$. Since this case allows to deal with subcycles of different length, this case can handle more types of cyclic groups than the variant of the previous section. To derive an efficient algorithm for such groups, we need a stronger version of Proposition 3.2, which we discuss next. Afterwards, we proceed with the discussion of the algorithm.

Proposition 3.5. *Let $I_0, I_1 \subsetneq [n]$ with $I_0 \cup I_1 \subsetneq [n]$ be disjoint index sets defining a complete set of fixings for every permutation in group $\Gamma \leq \langle(1, \dots, n)\rangle$. If $\mathcal{X}_\Gamma \cap F(I_0, I_1)$ is non-empty, then there exists $x \in F(I_0, I_1)$ such that $x \succ \gamma(x)$ for all $\gamma \in \Gamma \setminus \{\mathrm{id}\}$.*

Proof. Let $\hat{\imath} \coloneqq \min\{i \in [n] : i \notin I_0 \cup I_1\}$ be the smallest unfixed entry. We provide two constructions of a vector $\tilde{x} \in \mathcal{X}_\Gamma \cap F(I_0, I_1)$ with $\tilde{x} \succ \gamma(\tilde{x})$: one for $\hat{\imath} \leq \frac{1}{2}n$, and one for $\hat{\imath} > \frac{1}{2}n$.

Claim 3.6. *If $\hat{\imath} > \frac{1}{2}n$, then $\tilde{x} \in F(I_0, I_1)$ with $\tilde{x}_i = 1$ if $i \in I_1$ and $\tilde{x}_i = 0$ if $i \notin I_1$ satisfies $\tilde{x} \succ \gamma(\tilde{x})$ for all $\gamma \in \Gamma \setminus \{\mathrm{id}\}$.*

Proof. We prove the claim by contradiction in two parts. First, suppose $\tilde{x} = \gamma(\tilde{x})$ for some $\gamma \in \Gamma \setminus \{\mathrm{id}\}$. Then, $\tilde{x} = \delta(\tilde{x})$ for all $\delta \in \langle\gamma\rangle$, and since $\gamma \neq \mathrm{id}$ is a cyclic shift, there exists $\delta \in \langle\gamma\rangle$ with $\delta(\hat{\imath}) \leq \frac{1}{2}n < \hat{\imath}$. For this reason, we may assume w.l.o.g. that $\gamma(\hat{\imath}) \leq \frac{1}{2}n$. From this and $\tilde{x}_{\gamma(\hat{\imath})} = \gamma(\tilde{x})_{\gamma(\hat{\imath})} = \tilde{x}_{\hat{\imath}} = 0$, we conclude $\gamma(\hat{\imath}) \in I_0$. As permutation γ is a cyclic shift, $\gamma^{-1}(i) = i + \hat{\imath} - \gamma(\hat{\imath}) \pmod n$, wherein the modular residual classes identify with $\{1, \dots, n\}$. From $1 \leq \gamma(\hat{\imath}) \leq \frac{1}{2}n < \hat{\imath} \leq n$, we have that $1 \leq \gamma^{-1}(i) < \hat{\imath}$ for all $1 \leq i < \gamma(\hat{\imath})$, showing that $i, \gamma^{-1}(i) \in I_0 \cup I_1$ for all $i < \gamma(\hat{\imath})$. By the assertion $\tilde{x} = \gamma(\tilde{x})$, all vectors $x \in F(I_0, I_1)$ thus have $x =_{\gamma(\hat{\imath})} \gamma(x)$, and $x_{\gamma(\hat{\imath})} = 0$ by $\gamma(\hat{\imath}) \in I_0$. But $\hat{\imath} \notin I_0 \cup I_1$, and any $x \in F(I_0, I_1)$ with $x \succeq \gamma(x)$ must thus have $0 = x_{\gamma(\hat{\imath})} \geq \gamma(x)_{\gamma(\hat{\imath})} = x_{\hat{\imath}}$, such that $x_{\hat{\imath}} = 0$, which violates completeness for permutation γ. Hence $\tilde{x} \neq \gamma(\tilde{x})$.

Second, suppose $\tilde{x} \prec \gamma(\tilde{x})$ for some $\gamma \in \Gamma \setminus \{\mathrm{id}\}$. Let $j \coloneqq \min\{i \in [n] : \tilde{x}_i \neq \gamma(\tilde{x})_i\}$ and let $\check{\imath} \coloneqq \min\{i \in [n] : \gamma^{-1}(i) \notin I_0 \cup I_1\}$ be the first non-fixed entry in a vector permuted by γ. We derive some properties of such vectors $x \in F(I_0, I_1)$. Every vector $x \in F(I_0, I_1)$ satisfies $x_i = \tilde{x}_i$ for $i < \hat{\imath}$ and $\gamma(x)_i = \gamma(\tilde{x})_i$ for $i < \check{\imath}$. Moreover, $0 = \tilde{x}_j < \gamma(\tilde{x})_j = 1$, showing $j \notin I_1$ and $\gamma^{-1}(j) \in I_1$. As $\mathcal{X}_\Gamma \cap F(I_0, I_1)$ is non-empty and $\tilde{x} \prec \gamma(\tilde{x})$, $j \geq \min\{\hat{\imath}, \check{\imath}\}$. Thus, $x =_{\min\{\hat{\imath}, \check{\imath}\}} \gamma(x)$ for all $x \in F(I_0, I_1)$. We distinguish three cases.

On the one hand, suppose $\check{\imath} < \min\{\hat{\imath}, j\}$. Then there is $x \in F(I_0, I_1)$ with $\gamma(x)_{\check{\imath}} = 1$ and $x \succeq \gamma(x)$. But $\check{\imath} < \hat{\imath}$ implies $x_{\check{\imath}} = \tilde{x}_{\check{\imath}} = \gamma(\tilde{x})_{\check{\imath}} = 0$ and thus $\check{\imath} \in I_0$. So all vectors $x \in F(I_0, I_1)$ have $x =_{\check{\imath}} \gamma(x)$, and $x_{\check{\imath}} = 0$, but if $\gamma(x)_{\check{\imath}} = 1$, then $x \prec \gamma(x)$. To ensure $x \succeq \gamma(x)$, entry $\gamma^{-1}(\check{\imath})$ must be fixed to zero as well, which is in contradiction with completeness of the fixings and $\gamma^{-1}(\check{\imath}) \notin I_0 \cup I_1$. Hence $\check{\imath} \geq \min\{\hat{\imath}, j\}$, and we conclude from $j \geq \min\{\hat{\imath}, \check{\imath}\}$ that $j \geq \hat{\imath}$ and $\check{\imath} \geq \hat{\imath}$.

On the other hand, if $j = \hat{\imath}$, then $\gamma(\tilde{x})_{\hat{\imath}} = 1$, i.e., $\gamma^{-1}(\hat{\imath}) \in I_1$, Then, completeness yields $\hat{\imath} \in I_0$, which is a contradiction. Consequently, $j > \hat{\imath}$. Since $\hat{\imath} > \frac{1}{2}n$, there is exactly one sequence of $\hat{\imath} - 1$ consecutive numbers modulo n in $I_0 \cup I_1$ followed by a non-contained number. This sequence is $1, \ldots, \hat{\imath}$. In the vector permuted by γ, this sequence is mapped to $1 + (\check{\imath} - \hat{\imath}), \ldots, \hat{\imath} + (\check{\imath} - \hat{\imath})$, because γ is a cyclic shift, $\check{\imath}$ is the first non-fixed entry in the vector permuted by γ, and $\check{\imath} \geq \hat{\imath} > \frac{1}{2}n$. Hence, $\gamma^{-1}(\check{\imath}) = \hat{\imath}$, and thus $\gamma^{-1}(i) = i - \check{\imath} + \hat{\imath} \pmod{n}$. Then $\check{\imath} > \hat{\imath}$ follows from $\gamma \neq \mathrm{id}$, so $\check{\imath} > \hat{\imath}$ and $j > \hat{\imath}$.

Because $j > \hat{\imath}$ and $\check{\imath} > \hat{\imath}$, any vector $x \in F(I_0, I_1)$ has $x =_{\hat{\imath}} \gamma(x)$. From this we conclude $\gamma^{-1}(\hat{\imath}) \notin I_1$, because otherwise, completeness of I_1 implied $\hat{\imath} \in I_1$ to ensure $x \succeq_{\hat{\imath}+1} \gamma(x)$, contradicting the definition of $\hat{\imath}$. Thus, $\gamma^{-1}(\hat{\imath}) \in I_0$ because $\check{\imath} > \hat{\imath}$. However, this leads to a contradiction with completeness of permutation $\gamma^{-1} \in \Gamma$, as we show in the remainder of the proof.

The previously derived explicit formula for $\gamma^{-1}(i)$ implies $\gamma(i) = i + \check{\imath} - \hat{\imath} \pmod{n}$. As $\frac{1}{2}n < \hat{\imath} < \check{\imath} < n$, the first unfixed entry in the vector permuted by γ^{-1} is $\gamma^{-1}(\hat{\imath}) = 2\hat{\imath} - \check{\imath}$. Applying permutation γ^{-1} to any vector, permutes the entries $1 + (\check{\imath} - \hat{\imath}), \ldots, \hat{\imath} + (\check{\imath} - \hat{\imath})$ to $1, \ldots, \hat{\imath}$. Because for any vector $x \in F(I_0, I_1)$ we have $x =_{\hat{\imath}} \gamma(x)$, we can apply γ^{-1} on either side to find $\gamma^{-1}(x) =_{\gamma^{-1}(\hat{\imath})} \gamma^{-1}(\gamma(x))$, so $x =_{\gamma^{-1}(\hat{\imath})} \gamma^{-1}(x)$. By the previous paragraph, $\gamma^{-1}(\hat{\imath}) \in I_0$, so any $x \in F(I_0, I_1)$ has $x =_{\gamma^{-1}(\hat{\imath})} \gamma^{-1}(x)$ and $x_{\gamma^{-1}(\hat{\imath})} = 0$. As the set of fixings is complete for $x \succeq \gamma^{-1}(x)$, all $x \in F(I_0, I_1)$ have $\gamma^{-1}(x)_{\gamma^{-1}(\hat{\imath})} = x_{\hat{\imath}} = 0$, so $\hat{\imath} \in I_0$. This contradicts that $\hat{\imath} \notin I_0 \cup I_1$.

A contradiction is found for $\tilde{x} \preceq \gamma(\tilde{x})$ for all $\gamma \in \Gamma \setminus \{\mathrm{id}\}$, so $\tilde{x} \succ \gamma(\tilde{x})$. $\diamondsuit$

Claim 3.7. *If $\hat{\imath} \leq \frac{1}{2}n$, then $\tilde{x} \in F(I_0, I_1)$ with $\tilde{x}_i = 1$ if $i \in I_1 \cup \{\hat{\imath}\}$ and $\tilde{x}_i = 0$ if $i \notin I_1 \cup \{\hat{\imath}\}$ satisfies $\tilde{x} \succ \gamma(\tilde{x})$ for all $\gamma \in \Gamma \setminus \{\mathrm{id}\}$.*

Proof. The proof of Proposition 3.2 shows that $\tilde{x} \succeq \gamma(\tilde{x})$ for all $\gamma \in \Gamma$. To show that $\tilde{x} \succ \gamma(\tilde{x})$ if $\hat{\imath} < \frac{1}{2}n$, for the sake of contradiction suppose that $\tilde{x} = \gamma(\tilde{x})$ for some $\gamma \in \Gamma \setminus \{\mathrm{id}\}$. Then especially $\tilde{x} = \delta(\tilde{x})$ for all $\delta \in \langle\gamma\rangle$. Since $\tilde{x}_{\hat{\imath}} = 1$, for all $\delta \in \langle\gamma\rangle$ we have $\delta(\tilde{x})_{\hat{\imath}} = 1$, such that $\mathrm{orbit}(\gamma, \hat{\imath}) \subseteq I_1 \cup \{\hat{\imath}\}$.

Let $\delta \in \langle\gamma\rangle \setminus \{\mathrm{id}\}$ and $\check{\imath}_\delta := \min\{i \in [n] : \delta^{-1}(i) \notin I_0 \cup I_1\}$. If $\hat{\imath} \leq \check{\imath}_\delta$, then $x =_{\hat{\imath}} \delta(x)$ for all $x \in F(I_0, I_1)$. Moreover, $\delta(x)_{\hat{\imath}} = 1$ follows from $\mathrm{orbit}(\gamma, \hat{\imath}) \subseteq I_1 \cup \{\hat{\imath}\}$ and that $\delta \neq \mathrm{id}$ is cyclic, such that $\delta^{-1}(\hat{\imath}) \neq \hat{\imath}$. But then completeness of the fixings for permutation δ implies that $\hat{\imath} \in I_1$, which violates the definition of $\hat{\imath}$. Hence, $\hat{\imath} > \check{\imath}_\delta$ for all $\delta \in \langle\gamma\rangle \setminus \{\mathrm{id}\}$.

As $\check{\imath}_\delta < \hat{\imath}$, we have $\check{\imath}_\delta \in I_0 \cup I_1$, and $x =_{\check{\imath}_\delta} \delta(x)$ for all $x \in F(I_0, I_1)$. If $\check{\imath}_\delta \in I_0$, then completeness of the fixings for permutation δ yields $\delta^{-1}(\check{\imath}_\delta) \in I_0$, violating the definition of $\check{\imath}_\delta$. Thus, $\check{\imath}_\delta \in I_1$, and as such from $1 = \tilde{x}_{\check{\imath}_\delta} = \delta(\tilde{x})_{\check{\imath}_\delta}$ follows $\delta^{-1}(\check{\imath}_\delta) \in I_1 \cup \{\hat{\imath}\}$. Again, by the definition of $\check{\imath}_\delta$, $\delta^{-1}(\check{\imath}_\delta) \notin I_0 \cup I_1$, so $\delta^{-1}(\check{\imath}_\delta) = \hat{\imath}$. Hence, $\check{\imath}_\delta = \delta(\hat{\imath})$. As $\gamma \neq \mathrm{id}$ is a cyclic shift and $\hat{\imath} \leq \frac{1}{2}n$, there exists a $\delta \in \langle\gamma\rangle \setminus \{\mathrm{id}\}$ with $\check{\imath}_\delta = \delta(\hat{\imath}) > \frac{1}{2}n \geq \hat{\imath}$. This violates that $\hat{\imath} > \check{\imath}_\delta$ for all $\delta \in \langle\gamma\rangle \setminus \{\mathrm{id}\}$, so a contradiction follows. $\diamondsuit$ $\square$

In the remainder of this section, let $\Gamma \leq \langle\tilde{\gamma}\rangle$, where $\tilde{\gamma}$ is a composition of m disjoint, ordered and monotone cycles $\zeta_1, \ldots, \zeta_m$, with $N_i := \mathrm{supp}(\zeta_i) := \{k \in [n] : \zeta_i(k) \neq k\}$ for all $i \in [m]$, and with $\mathrm{supp}(\tilde{\gamma}) = [n] = N_1 \dot\cup \cdots \dot\cup N_m$.

Algorithm 3 shows a complete propagation algorithm for the lexicographic leaders of the Γ-orbit with a given set of initial fixings. It describes function `propagate`(Γ, I_0, I_1, k, computeFixings). When choosing $k = 1$, this determines whether $\mathcal{X}_\Gamma \cap F(I_0, I_1)$ is empty (i.e., INFEASIBLE) or not (i.e., FEASIBLE). If the Boolean parameter computeFixings is TRUE, then it additionally computes the complete set of fixings of $\mathcal{X}_\Gamma \cap F(I_0, I_1)$.

When running the algorithm with computeFixings set to FALSE, the worst-case running time is determined by determining the complete set of fixings of $\mathcal{X}_\gamma \cap F(I_0^t, I_1^t)$ for each $\gamma \in \hat{\Delta}_c := \mathrm{restr}(\Delta_c, N_c) \leq \langle\zeta_c\rangle$. As ζ_c is a monotone cycle, Algorithm 1 of Section 2 can be used. Hence, the running time of this variant of the algorithm is $f(\Gamma) := \mathcal{O}\left(\sum_{c \in [m]} |N_c| \,\mathrm{ord}(\hat{\Delta}_c)\right) \subseteq \mathcal{O}\left(n \max\{|N_c| : c \in [m]\}\right) \subseteq \mathcal{O}(n^2)$, as $\mathrm{ord}(\hat{\Delta}_c) \leq \mathrm{ord}(\langle\zeta_c\rangle) = |N_c|$. In the case where the complete set of fixings is to be determined, one additionally peeks for feasibility by calling the function recursively, with computeFixings set to FALSE. That happens at most as many times as the number of unfixed entries

Algorithm 3: `propagate`(Γ, I_0, I_1, k, computeFixings)

input : group $\Gamma \leq \langle\tilde{\gamma}\rangle$ for monotone and ordered $\tilde{\gamma} = \zeta_1 \circ \cdots \circ \zeta_m \in \mathcal{S}_n$, sets $I_0, I_1 \subseteq [n]$, $k \in [m]$, Boolean variable "computeFixings"

output: the message INFEASIBLE or FEASIBLE, and, if "computeFixings" is true, also two subsets of $[n]$

1 $\Delta_k \leftarrow \Gamma$, $(J_0, J_1) \leftarrow (I_0, I_1)$;
2 **for** $c \leftarrow k, \dots, m$ **do**
3 $\quad$ $(J_0, J_1) \leftarrow$ complete set of fixings of $\mathcal{X}_\gamma \cap F(J_0, J_1)$ for every $\gamma \in \mathrm{restr}(\Delta_c, N_c)$;
4 $\quad$ **if** $\mathcal{X}_{\mathrm{restr}(\Delta_c, N_c)} \cap F(J_0 \cap N_c, J_1 \cap N_c) = \emptyset$ **then**
5 $\quad\quad$ **return** INFEASIBLE;
6 $\quad$ **if** *computeFixings* **then**
7 $\quad\quad$ $(J_0', J_1') \leftarrow (J_0, J_1)$;
8 $\quad\quad$ **for** $i \in N_c \setminus (J_0' \cup J_1')$ **do**
9 $\quad\quad\quad$ **if** *`propagate`(Δ_c, $J_0 \cup \{i\}$, J_1, c, false) is* INFEASIBLE **then**
10 $\quad\quad\quad\quad$ $(J_0, J_1) \leftarrow (J_0, J_1 \cup \{i\})$;
11 $\quad\quad\quad$ **else if** *`propagate`(Δ_c, J_0, $J_1 \cup \{i\}$, c, false) is* INFEASIBLE **then**
12 $\quad\quad\quad\quad$ $(J_0, J_1) \leftarrow (J_0 \cup \{i\}, J_1)$;
13 $\quad$ **if** $N_c \setminus (J_0 \cup J_1) = \emptyset$ **then**
14 $\quad\quad$ $\Delta_{c+1} \leftarrow \mathrm{stab}(J_0 \cap N_c, \Delta_c) \cap \mathrm{stab}(J_1 \cap N_c, \Delta_c)$;
15 $\quad$ **else**
16 $\quad\quad$ $\Delta_{c+1} \leftarrow \mathrm{STAB}(N_c, \Delta_c)$;
17 **return** FEASIBLE, and (J_0, J_1) if computeFixings is true;

in the cycle, that is $|N_c|$ times for all $c \in [m]$. The total running time is therefore $\mathcal{O}\left(\sum_{c\in[m]}\left(|N_c|\operatorname{ord}(\hat{\Delta}_c) + |N_c| f(\Gamma)\right)\right) \subseteq \mathcal{O}\left(n^2 \max\{|N_c| : c \in [m]\}\right) \subseteq \mathcal{O}(n^3)$.

In this remainder of this section, we prove that Algorithm 3 correctly detects infeasibility, or finds the complete set of fixings.

Proposition 3.8. *Let $\tilde{\gamma} = \zeta_1 \circ \cdots \circ \zeta_m \in \mathcal{S}_n$ be a monotone and ordered permutation, let $\Gamma \leq \langle\tilde{\gamma}\rangle$, and let $I_0, I_1 \subseteq [n]$. Then, `propagate`(Γ, I_0, I_1, 1, true) returns* FEASIBLE *if and only if $\mathcal{X}_\Gamma \cap F(I_0, I_1) \neq \emptyset$. Moreover, the returned sets J_0 and J_1 define a complete set of fixings.*

Proof. We claim that, at the beginning of iteration c of the for-loop in Line 2, (J_0, J_1) defines a complete set of fixings on the first $c-1$ cycles of $\tilde{\gamma}$. Moreover, we claim that Δ_c is the intersection of all setwise stabilizers $\mathrm{stab}(J_0 \cap N_k, \Gamma) \cap \mathrm{stab}(J_1 \cap N_k, \Gamma)$ for cycles $k \in [c-1]$ with $N_k \subseteq J_0 \cup J_1$ and the pointwise stabilizers of all remaining cycles $k \in [c-1]$. The claim holds true for $c = 1$, so assume we have shown the claim for the first $c \in [m]$ iterations. We show that at the end of iteration c, (J_0, J_1) is complete on the first c cycles of $\tilde{\gamma}$ and Δ_{c+1} has the described properties. The assertion follows then by induction.

First, we show that we cannot derive further variable fixings from any $\gamma \in \Gamma \setminus \Delta_c$. For $x \in \mathbb{R}^n$, let x^c be the restriction of x onto N_c. Let $\gamma \in \Gamma \setminus \Delta_c$ and let $N_{k'}$, $k' \in [c-1]$, be the first cycle violating a stabilizer condition of Δ_c. On the one hand, if $N_{k'} \subseteq J_0 \cup J_1$, then all variables in $N_{k'}$ are fixed. Since all the previous cycles are stabilized, this means that, for any $x \in \mathcal{X}_\Gamma \cap F(J_0, J_1)$, x and $\gamma(x)$ coincide on the first $k'-1$ cycles and $x^{k'} \neq \gamma(x^{k'})$. As $x \succeq \gamma(x)$, the latter implies $x^{k'} \succ \gamma(x^{k'})$ and thus $x \succ \gamma(x)$. Hence, no variable fixings on N_c can be derived. On the other hand, if $N_{k'} \setminus (J_0 \cup J_1) \neq \emptyset$, Proposition 3.5 implies that there is $x \in \mathcal{X}_{\mathrm{restr}(\Delta_{k'}, N_{k'})} \cap F(J_0 \cap N_{k'}, J_1 \cap N_{k'})$ such that $x^k \succ \gamma(x^k)$ for any $\gamma \in \Delta_{k'}$ not stabilizing cycle $N_{k'}$ completely. Consequently, we again cannot deduce variable fixings on cycle N_k from such γ, and it is sufficient to check only the permutations in Δ_c to derive further variable fixings.

Second, we show that the algorithm finds all variable fixings on cycle N_c. By the above discussion, Δ_c stabilizes every $x \in \mathcal{X}_\Gamma \cap F(J_0, J_1)$ on the first $c-1$ cycles. Hence, the algorithm correctly terminates in Line 5 if $\mathcal{X}_{\Delta_c} \cap F(I_0^c, I_1^c) = \emptyset$ as any $x \in F(J_0, J_1)$ then satisfies $x^c \prec \gamma(x^c)$ for some $\gamma \in \Delta_c$. Next, the algorithm uses `propagate` to check, for each $i \in N_c \setminus (I_0^c \cup I_1^c)$ whether $\mathcal{X}_{\Delta_c} \cap F(J_0 \cup \{i\}, J_1)$ is empty. In this case, J_1 can be extended by i. The same reasoning is used to find variables that can be fixed to 0. Since this step is carried out for all not yet fixed variables on cycle N_k, this step finds all variable fixings on N_c provided that `propagate` is able to correctly determine whether $\mathcal{X}_\Delta \cap F(J_0, J_1)$ is empty or not. Then, after computing all fixings on N_c, the algorithm checks whether all entries on N_c are already fixed and updates the stabilizer Δ_{c+1} according to the rules described in the first paragraph.

To conclude the proof, we need to show that `propagate` indeed is able to determine whether $\mathcal{X}_{\Delta_c} \cap F(J_0, J_1)$ is empty or not. So, assume Δ_c stabilizes the first $c-1$ cycles as described above and that we call `propagate`(Δ, J_0, J_1, k, false). Then, the algorithm performs the following steps: it checks whether the If-statement in Line 5 shows infeasibility for cycle c, updates the stabilizer according to fixings on cycle N_c, and proceeds in the same way on cycles $c+1, \dots, m$. As explained above, the algorithm correctly terminates if it reaches Line 5 in one of these iterations, because Δ_c stabilizes the previous cycles depending on whether all of their entries are fixed or not. If the algorithm does not reach Line 5, we thus need to show that $\mathcal{X}_{\Delta_c} \cap F(J_0, J_1) \neq \emptyset$.

To this end, we construct $x \in \mathcal{X}_\Gamma \cap F(J_0, J_1)$ via subvectors $x^c \in \{0,1\}^{N_c}$ specifying each cycle $c \in [m]$. If $N_c \subseteq J_0 \cup J_1$, all entries on N_c are fixed and we define x^c accordingly; otherwise, let $x^c \in F(J_0 \cap N_c, J_1 \cap N_c)$ be a vector that satisfies $x^c \succ \gamma(x^c)$ for all nontrivial $\gamma \in \mathrm{restr}(\Delta_c, N_c)$, which exists by Proposition 3.5. We claim that $x \succeq \gamma(x)$ for all $\gamma \in \Gamma$. If the claim was wrong, there would exist $c \in [m]$ and $\gamma \in \Gamma$ such that x and $\gamma(x)$ coincide on the first $c-1$ cycles and $x^c \prec \gamma(x^c)$. Since γ stabilizes the first $c-1$ cycles, $\gamma \in \Delta_c$. Consequently, $N_c \subseteq J_0 \cup J_1$, because otherwise $x^c \succ \gamma(x^c)$ for all $\gamma \in \Delta_c$ that do not stabilize cycle c pointwise (i.e., $\mathrm{restr}(\gamma, N_c) \neq \mathrm{id}$), by construction. That is the case for γ that we consider, as otherwise $x^c \prec \gamma(x^c)$ is violated. This, however, means that $\mathcal{X}_{\mathrm{restr}(\Delta_c, N_c)} \cap F(J_0, J_1) = \emptyset$ and Algorithm 3 steps into Line 5. A contradiction to our assumption. □

4 Computational Results

This section's aim is to investigate the practical performance of the algorithms derived in the preceding sections. We are particularly interested in the following questions:

Q1 Given a cyclic group Γ, by how much does the running time of Algorithm 1 improve on the time for propagating the constraints $x \succeq \delta(x)$, $\delta \in \Gamma$, individually?

Q2 Algorithm 3 can be called with the boolean parameter "computeFixings". If it is TRUE, all possible fixings can be found for ordered and monotone permutations, but this requires additional computational effort. Is this additional running time compensated by strong fixings that can be detected?

Note that these questions are only *meaningful* for cyclic groups of order greater than 2 as otherwise there is no difference between Algorithm 1 and individual propagation.

To answer Q1 and Q2, we have implemented our methods in the mixed-integer programming framework SCIP [3]. SCIP already provides methods to compute symmetries of a mixed-integer program (MIP), and it can enforce $x \succeq \gamma(x)$ for a symmetry γ of a MIP via propagation and separation methods for so-called symresack and orbisack constraints, see [10]. If the symmetry group of a MIP is a product group and one of its factors defines an action associated with a certain symmetric group, SCIP applies orbitopal fixing, see Kaibel and Pfetsch [13] as well as Bendotti et al. [2], to handle the action of the entire factor. We have extended this code by a plugin that implements the enforcement of $x \succeq \delta(x)$ for all $\delta \in \langle\gamma\rangle$. This plugin uses Algorithm 3 if the representation of γ is monotone and ordered. Otherwise we use Algorithm 1, where Π consists of all non-identity group elements of $\langle\gamma\rangle$.

Computational Setup All experiments use a developer's version of `SCIP` 8.0.0.1 (git hash 430ca7b) as mixed-integer programming framework and `SoPlex` 5.0.1.3 as LP solver. `SCIP` detects symmetries of a MIP by building an auxiliary graph [34, 28, 32], and computing its automorphism group Γ using `bliss` 0.73 [12]. This way, we find the permutations $\gamma_1, \dots, \gamma_k$ that generate Γ. `SCIP` checks whether Γ is a product group $\Gamma_1 \otimes \dots \otimes \Gamma_\ell$, since the symmetries of each factor Γ_i can be handled independently, c.f. [10]. Based on the generators, `SCIP` heuristically decides whether a factor Γ_i can be completely handled by orbitopal fixing, c.f. [10]. Since this check evaluates positively only if no generator of Γ_i has a cycle of length at least 3, we also make use of orbitopal fixing in our experiments since no permutation that is *meaningful* for Q1 and Q2 is already handled by orbitopal fixing.

The generators of the remaining factors are handled using different strategies that will allow us to answer Q1 and Q2. More precisely, we compare the following settings for the generators γ_i that are not handled via orbitopal fixing:

nosym No symmetry handling is applied, in particular also no orbitopal fixing;

gen We only propagate generator constraints $x \succeq \gamma_i(x)$;

group We propagate the constraints $x \succeq \delta(x)$ for all $\delta \in \langle \gamma_i \rangle \setminus \{\text{id}\}$ individually;

nopeek We propagate the constraints $x \succeq \delta(x)$ for all $\delta \in \langle \gamma_i \rangle \setminus \{\text{id}\}$ using Algorithm 3 if γ_i is monotone and ordered with parameter *completeFixings* set to FALSE, and otherwise using Algorithm 1;

peek Same as `nopeek`, but with *completeFixings* set to TRUE. The constraints $x \succeq \delta(x)$ are handled for all $i \in [k]$ and $\delta \in \langle \gamma_i \rangle \setminus \{\text{id}\}$ by Algorithm 3 with *completeFixings* set to TRUE if γ_i is monotone and ordered (guaranteeing completeness). Otherwise, Algorithm 1 is used and we check whether further variable fixings can be found using Observation 2.1. This check is carried out for all unfixed variables whose fixing values were looked up in the previous calls to Algorithm 1.

Only in the `peek` setting with an ordered and monotone generator, our methods are guaranteed to find the complete set of fixings for the group generated by this generator. Since none of the generators $\gamma_1, \dots, \gamma_k$ is guaranteed to be in monotone and ordered representation, we also implemented a heuristic that relabels the variable array to guarantee that at least one generator per factor is monotone and ordered. It iterates over the generators, sorted descending with respect to the largest subcycle size and then descending on the group order. If the variable indices of the generator are not yet relabeled, then it relabels these such that the generator becomes monotone and ordered. For this, we provide three options: ***(i)*** sorting the subcycles in decreasing length (`max`), ***(ii)*** sorting the subcycles in increasing length (`min`), and ***(iii)*** sorting the subcycles based on the variable index that is minimal in the original variable ordering (`respect`). For example, consider three generators $\gamma_1 = (1, 8, 7, 3)$, $\gamma_2 = (3, 4, 5, 8)$, and $\gamma_3 = (2, 5, 6, 9, 4)$. The heuristic relabels γ_1, does not relabel γ_2 since it also contains variable index 3, and relabels γ_3. Using hats to distinguish the relabeled space, the heuristic yields $[1, 2, 3, 4, 5, 6, 7, 8, 9] \mapsto [\hat{1}, \hat{5}, \hat{4}, \hat{9}, \hat{6}, \hat{7}, \hat{3}, \hat{2}, \hat{8}]$, such that $\gamma_1 = (\hat{1}, \hat{2}, \hat{3}, \hat{4})$, $\gamma_2 = (\hat{4}, \hat{9}, \hat{6}, \hat{2})$, and $\gamma_3 = (\hat{5}, \hat{6}, \hat{7}, \hat{8}, \hat{9})$. Note that γ_1 and γ_3 are monotone (and ordered) in this relabeled space. In particular, with this heuristic we make sure that at least the first processed generator has a monotone and ordered representation.

Note that a cyclic group $\langle \gamma \rangle$ acting on n elements might have order $\Omega(2^n)$. To prevent memory overflow in Algorithm 1 and in the individual propagation (`group`), we added a safeguard that might only propagate a restricted set of permutations in these algorithms. Let s be the cardinality of the support of the generator γ and $\Pi := \langle \gamma \rangle \setminus \{\text{id}\}$. If $|\Pi| > 10^4$ or $s|\Pi| > 5 \cdot 10^6$, we only enforce the symmetry constraints for $\Pi' := \{\gamma^i : i \in [k]\}$, where $k \in \mathbb{Z}_+$ is maximal such that $|\Pi'| \leq 10^4$ and $s|\Pi'| \leq 5 \cdot 10^6$.

In the following subsections, we report on our experiments for two different classes of problems, including benchmarking instances. To reduce the impact of performance variability, all instances have been run with 5 different random seeds. The running time t_i per instance i is reported in shifted geometric mean $\prod_{i=1}^{n}(t_i + s)^{\frac{1}{n}} - s$ with a shift of $s = 10\,\text{s}$, as well as the number of instances solved (S) within the time limit of 2 h per instance. If the time limit is hit, then we report a running time of 2 h for that

Table 1: 3-edge coloring on flower snark graphs J_n for $n \in \{3, \ldots, 43\}$.

	nosym		gen		group		nopeek		peek	
relabeling	time(s)	S	time(s)	S	time(s)	S	time(s)	S	time(s)	S
original	730.78	54	172.35	88	187.56	87	169.79	93	153.23	97
max	–		407.02	65	312.93	70	278.00	77	270.19	78
min	–		97.99	102	131.58	95	127.48	92	119.67	95
respect	–		184.44	88	173.41	86	174.32	88	178.46	87
aggregated	730.78	54	189.90	343	191.75	338	180.33	350	172.84	357

run. In the article's main part, we present aggregated results, showing the shifted geometric means of all instances with the given settings, as well as the number of instances solved. Detailed tables can be found in Appendix B. These tables contain information per instance, as well as the total time spent on the instances, and the total time and percentage of time spent on symmetry handling. All computations were run on a Linux cluster with Intel Xeon E5 3.5 GHz quad core processors and 32 GB memory. The code was executed using a single thread.

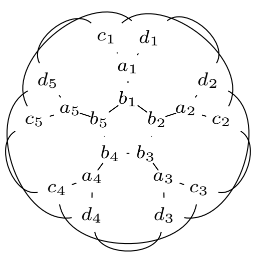
Edge-coloring flower snarks Flower snark graphs [5, 11], described in Figure 7, are a family of undirected graphs with chromatic index 4, i.e., there is no coloring of the edges with three colors such that incident edges are colored differently. Deciding whether an edge coloring with 3 colors exists for an undirected graph $\mathcal{G} = (V, E)$ is equivalent to decide whether

$$S_{\mathcal{G}} = \left\{ x \in \{0,1\}^{E \times [3]} : \begin{array}{l} x_{e,k} + x_{e',k} \leq 1 \text{ for all } k \in [3] \text{ and } e, e' \in E \text{ with } |e \cap e'| = 1, \\ \sum_{k \in [3]} x_{e,k} = 1 \text{ for all } e \in E. \end{array} \right\}$$

is empty, which can be done by binary programming techniques. Margot [27] studied this binary program for the flower snarks J_{13}, J_{15} and J_{21}, and we also use these snarks in our experiments as they admit cyclic symmetries. The flower snarks are defined for odd $n \geq 3$ and have an automorphism group of order $4n$. The group is generated by a cycle of order $2n$ and a reflection. Symmetries in the problem are therefore given by these graph automorphisms, and by interchanging the edge colors. These symmetries are automatically identified by SCIP.

Our testset consists of all these instances with all odd parameters $n \in \{3, \ldots, 49\}$. In Table 1 we present the aggregated results for these instances, and we refer to Section B for the results per instance. Comparing the shifted geometric means of our proposed methods (`nopeek` and `peek`) with naively propagating individual constraints $x \succeq \delta(x)$ (`group`), without relabeling, our proposed methods gain 9.5 % and 18.3 %, respectively. In particular, in both the `peek` and `nopeek` variant at least one run of all instances solved the problem, while `group` could not solve a single instance for $n \geq 45$. These results are even more pronounced if we consider the subset of instances that could not be solved without symmetry handling within the time limit, which is for parameter $n \geq 27$. These results are shown in Table 2: we gain 30.8 % and 16.9 % when comparing `nopeek` and `peek` with in the original labeling, respectively.

Since the instances only have one generator with order larger than two, our algorithm guarantees completeness with respect to this cyclic subgroup in the automatically



Construction of flower snark graph Graph J_n is defined for odd $n \geq 3$ and has $4n$ vertices labeled by a_i, b_i, c_i, d_i for $i \in [n]$. For $i \in [n]$, connect a_i to b_i, c_i, d_i, make cycle $(b_1, b_2, \ldots, b_n)$, and for $i < n$ connect c_i to d_{i+1}, and d_i to c_{i+1}. Last, connect c_n to c_1 and d_n to d_1. The automorphism group generators are $(a_1, \ldots, a_n)(b_1, \ldots, b_n)(c_1, \ldots, c_n, d_1, \ldots, d_n)$, and $(c_1, d_1) \cdot \prod_{i=2}^{\frac{n+1}{2}} (a_i, a_{n+2-i})(b_i, b_{n+2-i})(c_i, c_{n+2-i})(d_i, d_{n+2-i})$.

Figure 7: Flower snark graph J_5 and construction of J_n.

Table 2: 3-edge coloring on flower snark graphs J_n for $n \in \{27, \ldots, 43\}$.

	nosym		gen		group		nopeek		peek	
relabeling	time(s)	S	time(s)	S	time(s)	S	time(s)	S	time(s)	S
original	7200.00	0	1266.33	33	1526.99	32	1269.40	38	1056.42	42
max	–		4752.76	10	3501.50	15	2884.53	22	2734.31	23
min	–		531.09	47	912.18	40	889.73	37	797.14	40
respect	–		1480.93	33	1444.56	31	1452.88	33	1509.23	32
aggregated	7200.00	0	1478.12	123	1630.32	118	1475.85	130	1366.37	137

Table 3: MIPLIB 2010 and MIPLIB 2017 benchmark instances with group generators larger than two.

	nosym		gen		group		nopeek		peek	
relabeling	time(s)	S	time(s)	S	time(s)	S	time(s)	S	time(s)	S
original	1853.78	42	491.69	50	407.76	50	449.34	48	506.23	48
max	–		1061.29	47	812.31	48	771.56	48	796.69	49
min	–		901.65	48	612.10	50	837.48	47	657.29	49
respect	–		970.19	47	743.01	49	639.20	50	723.93	49
aggregated	1853.78	42	822.47	192	623.37	197	656.66	193	662.01	195

relabeled `peek`-variants. As the tables show, relabeling has a big impact on the running times and on the number of instances that we can solve. We observe a significant improvement when comparing the `nopeek` and `peek` variants with `group` in the `max`-relabeling and `min`-relabeling. In the `respect`-relabeling a worse result is found. Surprisingly, the weaker symmetry handling method `gen` is on average the fastest in the `min`-relabeling.

The `peek`-variant requires significant additional computational time. This is also reflected in the total time spent on handling symmetries, which is 16.7 %, 18.0 %, 6.9 % and 5.9 % for the original, max, min and respect relabelings, respectively. As a comparison, this is at most 3.7 % for the `nopeek`-variant. However, this investment pays off in terms of the average time to solve the instances, since the `peek`-variant solves the instances on average faster than the `nopeek`-variant, when comparing the aggregated results. We hypothesize that the additional fixings found by peeking outweighs the additional time spent on handling symmetries, and that this effect will be even more pronounced for more difficult instances with a larger cyclic symmetry subgroup order. This is consistent with the numbers in the tables of the supplements: For instances with a higher value of the parameter n, the running times of `nopeek` and `peek` relative to `nosym`, `gen` and `symregroup` decrease. For the flower snark instances, we thus can answer Q1 and Q2 affirmatively as Algorithm 3 and (in the not relabeled case) Algorithm 1 clearly outperform the `group` setting.

MIPLIB testset MIPLIB 2010 [17] and MIPLIB 2017 [8] are data bases of benchmark MIP instances consisting of 1426 instances. We extracted all instances for which SCIP could identify at least one generator that defines a cyclic group of order at least 3, and that SCIP can presolve within three hours. This results in 38 instances relevant for our experiments. We restricted this test set further by removing all instances that cannot be solved by any of our settings within three hours or that can be solved within presolving. This results in 10 instances, which we use for the experiments below. Note that this is a very small testset and that the characteristics between the instances can be very different. For this reason, we carefully analyze aggregated results and refer to the appendix for detailed per-instance results when necessary.

The results of the performance tests on the 10 instances are shown in Table 3. It is clear that the results are very sensitive to the chosen relabeling variant. On average, the instances could be solved significantly faster when the variable ordering of the original problem is used, rather than any relabeled variant. We hypothesize that this result can be attributed to internal behavior of SCIP, where operations (such as branching) on variables with a low internal index are preferred. Moreover, in contrast to the flower snark instances, many benchmark instances find multiple generators of order larger than two, the order of these generators may be low, and we may not always be able to relabel all these group generators such that they are all monotone and

Table 4: Relevant MIPLIB 2010 and MIPLIB 2017 benchmark instances, without automatic relabeling.

	nosym		gen		group		nopeek		peek	
instance	time(s)	S	time(s)	S	time(s)	S	time(s)	S	time(s)	S
cod105	7200.04	0	65.94	5	61.84	5	59.14	5	57.45	5
cov1075	4761.35	5	117.17	5	47.48	5	32.71	5	33.15	5
fastxgemm-n2r6s0t2	1628.36	5	286.01	5	209.40	5	141.51	5	145.82	5
fastxgemm-n2r7s4t1	6394.82	2	971.15	5	812.15	5	768.81	5	776.56	5
neos-1324574	6251.90	5	2398.62	5	2080.58	5	2064.22	5	2060.96	5
neos-3004026-krka	129.58	5	262.07	5	144.61	5	685.34	4	1286.86	4
neos-953928	2332.51	5	1471.52	5	1166.06	5	975.68	5	975.07	5
neos-960392	850.24	5	1954.62	5	2278.05	5	2451.05	4	2454.52	4
supportcase29	282.05	5	196.09	5	253.30	5	388.96	5	662.54	5
wachplan	2713.70	5	906.09	5	939.21	5	849.65	5	849.25	5
All instances combined	1853.78	42	491.69	50	407.76	50	449.34	48	506.23	48
Total time	45:46:14		13:24:05		12:12:07		15:57:33		18:43:22	
Symmetry time	0:00:00		0:03:44		0:03:47		0:06:56		1:56:05	
Percentage time	0.0%		0.5%		0.5%		0.7%		10.3%	

Table 5: Relevant MIPLIB 2010 and MIPLIB 2017 benchmark instances, without *neos-3004026-krka*, *neos-920392*, and *supportcase29*.

	nosym		gen		group		nopeek		peek	
relabeling	time(s)	S	time(s)	S	time(s)	S	time(s)	S	time(s)	S
original	3917.80	27	501.57	35	393.39	35	337.55	35	338.57	35
max	–		1193.75	33	668.57	35	694.06	35	698.20	35
min	–		1062.99	33	719.87	35	710.16	35	706.24	35
respect	–		1226.05	33	761.70	35	678.57	35	715.09	35
aggregated	3917.80	27	940.64	134	616.62	140	580.20	140	588.38	140

ordered. Together, for such diverse instances, this may explain why relabeling has a negative impact on the running times. This means that even in the heuristically relabeled and peek-cases, since after relabeling often not all generators are monotone and ordered, we often need to fallback to the simpler variant (Algorithm 1), which does not guarantee completeness for the generating subgroup and comes at a higher asymptotic computational cost. This might explain why our more sophisticated algorithms perform on average by roughly 5 % and 10 % worse than the group setting. Due to the small size of the test set, however, one needs to be careful interpreting the aggregated numbers as they can be highly biased because of outliers.

Looking into the results on a per-instance basis confirms this conjecture. Table 4 provides these results without relabeling. We can observe two classes of instances. The instances for which no symmetry handling (nosym) or only handling generators (gen) perform best—*neos-3004026-krka*, *neos-960392*, and *supportcase29*—and the remaining instances which benefit from more aggressive symmetry handling. On the three mentioned instances, our methods are significantly slower than group. However, since (almost) no symmetry handling performs best for these instances, we conjecture that symmetry handling might sometimes not be a favorable strategy. For example, neos-3004026-krka and neos-960392 have already an optimal dual bound after processing the root node, i.e., once a matching primal solution is found, SCIP immediately terminates. Based on the results it thus seems that the symmetry-based variable fixings hinder SCIP's heuristics to find a good solution. In this case, of course, less symmetry handling is beneficial.

On the remaining seven instances that benefit from symmetry handling, however, our methods nopeek and peek clearly dominate the group setting. As Table 5 shows, we can improve the running time of group by 5.9 % and 4.6 %, respectively, if considering the mean running for all relabeling strategies. Without relabeling, the performance improves even by 14.2 % and 13.9 %, respectively. Thus, these numbers also indicate a positive answer to Question Q1: If symmetry handling is important to solve an instance efficiently, then exploiting the structure of cyclic groups via Algorithms 1 and 3 outperforms the approach that neglects the group structure. For Question Q2, the answer is less clear, because the performance of all symmetry handling methods is highly sensitive to changing the variable labeling.

Conclusion The algorithms developed in this article allow to efficiently handle symmetries of cyclic groups. In particular, if the generator of the cyclic group is ordered and monotone, Algorithm 3 is guaranteed to find all symmetry based variable fixings that can be derived from a set of given fixings. As illustrated by our numerical experiments for the flower snark instances, this algorithm clearly dominates the propagation of individual constraints (`group`) and the weaker (but faster) variant of Algorithm 3 that does not come with a guarantee of completeness. On general benchmark instances, the situation is less clear. On the one hand, it seems that some instances do not benefit from symmetry handling, e.g., because of interferences with heuristics. On the other hand, while flower snark instances only had one cyclic generator, the benchmark instances might admit several cyclic generators that cannot be made ordered and monotone simultaneously. Here, the best results for instances that benefit from symmetry handling could be achieved without relabeling. In this case, the completeness of Algorithm 3 is not guaranteed, but the weaker `nopeek` version still dominates the individual treatment of permutations. We thus conclude that the dedicated methods that exploit the structure of cyclic groups developed in this article clearly help to improve the performance of symmetry handling in `SCIP`.

Nevertheless, the numerical results also show interesting possibilities for future research. On the one hand, it seems that it is important to identify which instances benefit from (aggressive) symmetry handling. On the other hand, since completeness of Algorithm 3 is only guaranteed for monotone and ordered permutations γ, it would be helpful to derive methods that achieve completeness even if one of the assumptions on γ are dropped. This reduces the impact of the labeling, which has be shown to highly influence the performance of `SCIP`. Finally, if no alternative algorithms can be developed, also more sophisticated methods for finding a good relabeling could be beneficial. This, however, is out of scope of this article.

Acknowledgments We are thankful to Rudi Pendavingh, Marc E. Pfetsch, and Frits Spieksma for carefully reading a preliminary version of the article and their suggestions to improve the presentation.

References

[1] László Babai and Eugene M. Luks. Canonical labeling of graphs. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 171–183, 1983.

[2] Pascale Bendotti, Pierre Fouilhoux, and Cécile Rottner. Orbitopal fixing for the full (sub-)orbitope and application to the unit commitment problem. *Mathematical Programming*, 186:337–372, 2021.

[3] Ksenia Bestuzheva, Mathieu Besançon, Wei-Kun Chen, Antonia Chmiela, Tim Donkiewicz, Jasper van Doornmalen, Leon Eifler, Oliver Gaul, Gerald Gamrath, Ambros Gleixner, Leona Gottwald, Christoph Graczyk, Katrin Halbig, Alexander Hoen, Christopher Hojny, Rolf van der Hulst, Thorsten Koch, Marco Lübbecke, Stephen J. Maher, Frederic Matter, Erik Mühmer, Benjamin Müller, Marc E. Pfetsch, Daniel Rehfeldt, Steffan Schlein, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Boro Sofranac, Mark Turner, Stefan Vigerske, Fabian Wegscheider, Philipp Wellner, Dieter Weninger, and Jakob Witzig. The SCIP Optimization Suite 8.0. Technical report, Optimization Online, December 2021.

[4] James Crawford, Matthew Ginsberg, Eugene Luks, and Amitabha Roy. Symmetry-breaking predicates for search problems. In *Proceedings of the 5-th Internationa Conference on Knowledge Representation and Reasoning, (KR '96)*, pages 148–159. Morgan Kaufmann, 1996.

[5] Stanley Fiorini and Robin J. Wilson. *Edge-colourings of graphs*. Number 16 in Research Notes in Mathematics. Pitman Publishing Limited, 1977.

[6] Matteo Fischetti and Leo Liberti. Orbital shrinking. In A. Ridha Mahjoub, Vangelis Markakis, Ioannis Milis, and Vangelis Th. Paschos, editors, *Combinatorial Optimization*, volume 7422 of *Lecture Notes in Computer Science*, pages 48–58. Springer Berlin Heidelberg, 2012.

[7] Eric J. Friedman. Fundamental domains for integer programs with symmetries. In Andreas Dress, Yinfeng Xu, and Binhai Zhu, editors, *Combinatorial Optimization and Applications*, volume 4616 of *Lecture Notes in Computer Science*, pages 146–153. Springer Berlin Heidelberg, 2007.

[8] Ambros Gleixner, Gregor Hendel, Gerald Gamrath, Tobias Achterberg, Michael Bastubbe, Timo Berthold, Philipp M. Christophel, Kati Jarck, Thorsten Koch, Jeff Linderoth, Marco Lübbecke, Hans D. Mittelmann, Derya Ozyurt, Ted K. Ralphs, Domenico Salvagnin, and Yuji Shinano. MIPLIB 2017: Data-Driven Compilation of the 6th Mixed-Integer Programming Library. *Mathematical Programming Computation*, 13:443–490, 2021.

[9] Christopher Hojny. Packing, partitioning, and covering symresacks. *Discrete Applied Mathematics*, 283:689–717, 2020.

[10] Christopher Hojny and Marc E. Pfetsch. Polytopes associated with symmetry handling. *Mathematical Programming*, 175:197–240, 2019.

[11] Rufus Isaacs. Infinite families of nontrivial trivalent graphs which are not tait colorable. *The American Mathematical Monthly*, 82(3):221–239, 1975.

[12] Tommi Junttila and Petteri Kaski. bliss: A tool for computing automorphism groups and canonical labelings of graphs. URL: http://www.tcs.hut.fi/Software/bliss/.

[13] Volker Kaibel, Matthias Peinhardt, and Marc E. Pfetsch. Orbitopal fixing. *Discrete Optimization*, 8(4):595–610, 2011.

[14] Volker Kaibel and Marc E. Pfetsch. Packing and partitioning orbitopes. *Mathematical Programming*, 114(1):1–36, 2008.

[15] George Katsirelos, Nina Narodytska, and Toby Walsh. Breaking generator symmetry. *CoRR*, abs/0909.5099, 2009.

[16] George Katsirelos, Nina Narodytska, and Toby Walsh. On the complexity and completeness of static constraints for breaking row and column symmetry. In David Cohen, editor, *Principles and Practice of Constraint Programming – CP 2010*, pages 305–320. Springer Berlin Heidelberg, 2010.

[17] Thorsten Koch, Tobias Achterberg, Erling Andersen, Oliver Bastert, Timo Berthold, Robert E. Bixby, Emilie Danna, Gerald Gamrath, Ambros M. Gleixner, Stefan Heinz, Andrea Lodi, Hans Mittelmann, Ted Ralphs, Domenico Salvagnin, Daniel E. Steffy, and Kati Wolter. MIPLIB 2010. *Mathematical Programming Computation*, 3(2):103–163, 2011.

[18] Ailsa H. Land and Alison G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960.

[19] Leo Liberti. Automatic generation of symmetry-breaking constraints. In *Combinatorial optimization and applications*, volume 5165 of *Lecture Notes in Computer Science*, pages 328–338. Springer, Berlin, 2008.

[20] Leo Liberti. Reformulations in mathematical programming: automatic symmetry detection and exploitation. *Mathematical Programming*, 131(1-2):273–304, 2012.

[21] Leo Liberti. Symmetry in mathematical programming. In Jon Lee and Sven Leyffer, editors, *Mixed Integer Nonlinear Programming*, volume 154 of *IMA Series*, pages 236–286. Springer, New York, 2012.

[22] Leo Liberti and James Ostrowski. Stabilizer-based symmetry breaking constraints for mathematical programs. *Journal of Global Optimization*, 60:183–194, 2014.

[23] Andreas Loos. *Describing Orbitopes by Linear Inequalities and Projection Based Tools*. PhD thesis, Otto-von-Guericke-Universität Magdeburg, 2010.

[24] Eugene Luks and Amitabha Roy. The complexity of symmetry-breaking formulas. *Annals of Mathematics and Artificial Intelligence*, 41:19–45, 2004.

[25] François Margot. Pruning by isomorphism in branch-and-cut. *Mathematical Programming*, 94(1):71–90, 2002.

[26] François Margot. Exploiting orbits in symmetric ILP. *Mathematical Programming*, 98(1–3):3–21, 2003.

[27] François Margot. Symmetric ilp: Coloring and small integers. *Discrete Optimization*, 4(1):40–62, 2007.

[28] François Margot. Symmetry in integer linear programming. *50 Years of Integer Programming 1958-2008*, pages 647–686, 2010.

[29] Nina Narodytska and Toby Walsh. Breaking symmetry with different orderings. In Christian Schulte, editor, *Principles and Practice of Constraint Programming*, volume 8124 of *Lecture Notes in Computer Science*, pages 545–561. Springer Berlin Heidelberg, 2013.

[30] James Ostrowski, Miguel F. Anjos, and Anthony Vannelli. Modified orbital branching for structured symmetry with an application to unit commitment. *Mathematical Programming*, 150(1):99–129, 2015.

[31] James Ostrowski, Jeff Linderoth, Fabrizio Rossi, and Stefano Smriglio. Orbital branching. *Mathematical Programming*, 126(1):147–178, 2011.

[32] Marc E. Pfetsch and Thomas Rehn. A computational comparison of symmetry handling methods for mixed integer programs. *Mathematical Programming Computation*, 11(1):37–93, 2019.

[33] Amitabha Roy. Symmetry-breaking formulas for groups with bounded orbit projections. In *The Seventh International Workshop on Symmetry and Constraint Satisfaction Problems (SymCon'07)*, page 72, 2007.

[34] Domenico Salvagnin. A dominance procedure for integer programming. *Master's thesis, University of Padova, Padova, Italy*, 2005.

[35] Domenico Salvagnin. Symmetry breaking inequalities from the Schreier-Sims table. In Willem-Jan van Hoeve, editor, *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 521–529, Cham, 2018. Springer International Publishing.

A Proofs of Section 2

In Section 2, we presented various propositions, without giving their proofs. In this appendix, the missing proofs are provided.

Proposition 2.12 (Sufficient conditions for completeness)**.** *Consider $\gamma \in \Pi$, let t be some time index, and $\emptyset \notin D^t_\gamma$. Suppose that, if $\{(i,b)\} \in D^t_\gamma$, then $i \in I^t_{1-b}$. Then, the set of fixings (I^t_0, I^t_1) is complete for $x \succeq \gamma(x)$ if* **(P1.)** *$E^t_\gamma = \emptyset$, or* **(P2.)** *$i^t_\gamma > n$, or* **(P3.)** *all of the following: $\emptyset \notin E^t_\gamma$, and $i^t_\gamma \notin I^t_0$, and $\gamma^{-1}(i^t_\gamma) \notin I^t_1$, and $\gamma(i^t_\gamma) > i^t_\gamma$, and $\gamma^{-1}(i^t_\gamma) > i^t_\gamma$.*

Proof. Let $\gamma \in \Pi$, t be some time index, $\emptyset \notin D^t_\gamma$, and that $\{(i,b)\} \in D^t_\gamma$ implies $i \in I^t_{1-b}$. For brevity, we denote $F^t := F(I^t_0, I^t_1)$, and $\mathcal{X}^t_\gamma := \mathcal{X}^{(i^t_\gamma)}_\gamma$. We prove correctness for each of *(P1)*, *(P2)* and *(P3)* by contradiction, by assuming that the set of fixings (I^t_0, I^t_1) is not complete for the constraint $x \succeq \gamma(x)$ with $\gamma \in \Pi$. We denote a missing valid fixing by $f = (i,b)$, and show that a contradiction follows. This means that $\mathcal{X}_\gamma \cap F^t \cap V(\bar{f}) = \emptyset$. Also, $i \notin I^t_0 \cup I^t_1$ implies $\{\bar{f}\} \notin D^t_\gamma$, which means $\mathcal{X}^t_\gamma \cap F^t \cap V(\bar{f}) \neq \emptyset$.

First suppose *(P1)*, that $E^t_\gamma = \emptyset$. Then no vector $x \in F^t$ exists with $x =_{i^t_\gamma} \gamma(x)$. In this case, $x \prec \gamma(x)$ for all $x \in F^t$ with $x_i = 1-b$, since $\mathcal{X}_\gamma \cap F^t \cap V(\bar{f}) = \emptyset$. Hence, $x \prec_{i^t_\gamma} \gamma(x)$. A contradiction follows, as this observation and $\emptyset \notin D^t_\gamma$ shows that $\{\bar{f}\}$ is a minimal i^t_γ-inf-conjunction, contradicting $\{\bar{f}\} \notin D^t_\gamma$.

Second suppose *(P2)*. If $i^t_\gamma > n$, then $\mathcal{X}_\gamma \cap F^t = \mathcal{X}^t_\gamma \cap F^t$ violates $\mathcal{X}_\gamma \cap F^t \cap V(\bar{f}) = \emptyset \neq \mathcal{X}^t_\gamma \cap F^t \cap V(\bar{f})$.

Last suppose *(P3)*. For any vector $x \in \{0,1\}^n$, with $x_{i^t_\gamma} = 1$ and $\gamma(x)_{i^t_\gamma} = x_{\gamma^{-1}(i^t_\gamma)} = 0$, we have that $x \succeq_{i^t_\gamma} \gamma(x)$ implies $x \succeq \gamma(x)$, so $\mathcal{X}_\gamma \supseteq \mathcal{X}^t_\gamma \cap V(\{(i^t_\gamma, 1), (\gamma^{-1}(i^t_\gamma), 0)\})$. Hence, $\mathcal{X}_\gamma \cap F^t \cap V(\bar{f}) = \emptyset$ yields $\mathcal{X}^t_\gamma \cap F^t \cap V(\{\bar{f}, (i^t_\gamma, 1), (\gamma^{-1}(i^t_\gamma), 0)\}) = \emptyset$, meaning that $C = \{\bar{f}, (i^t_\gamma, 1), (\gamma^{-1}(i^t_\gamma), 0)\}$ is an i^t_γ-inf-conjunction. As $\gamma(i^t_\gamma) > i^t_\gamma$, entry i^t_γ does not occur in the first $i^t_\gamma - 1$ entries of a vector x and $\gamma(x)$, so $(i^t_\gamma, 1)$ cannot be element of a minimal i^t_γ-inf-conjunction. Likewise, because $\gamma^{-1}(i^t_\gamma) > i^t_\gamma$ and $\gamma(\gamma^{-1}(i^t_\gamma)) = i^t_\gamma$, entry $\gamma^{-1}(i^t_\gamma)$ does also not occur in the first $i^t_\gamma - 1$ entries of any vector x and $\gamma(x)$, so $(\gamma^{-1}(i^t_\gamma), 0)$ cannot be element of any minimal i^t_γ-inf-conjunction, too. By removing these, we have that $\{\bar{f}\}$ is an i^t_γ-inf-conjunction, contradicting $\mathcal{X}^t_\gamma \cap F^t \cap V(\bar{f}) \neq \emptyset$. □

Proposition 2.14 (Updating eq-conjunctions for index increasing event)**.** *Consider an index increasing event for permutation $\gamma \in \Pi$ at time t. Then, for $\delta \in \Pi \setminus \{\gamma\}$, we have $E^{t+1}_\delta = E^t_\delta$, and $E^{t+1}_\gamma = Y \cup Z$, where*

$$Y = \left\{ C : \begin{array}{l} C \in E^t_\gamma, \textit{and} \\ \textit{for all } x \in \mathcal{X}^{(i^t_\gamma)}_\gamma \cap F(I^t_0, I^t_1) \cap V(C) \textit{ holds } x_{i^t_\gamma} \leq \gamma(x)_{i^t_\gamma}, \textit{ and} \\ \textit{there is } x \in \mathcal{X}^{(i^t_\gamma)}_\gamma \cap F(I^t_0, I^t_1) \cap V(C) \textit{ with } x_{i^t_\gamma} = \gamma(x)_{i^t_\gamma} \end{array} \right\}, \textit{ and}$$

$$Z = \left\{ C \cup S : \begin{array}{l} C \in E^t_\gamma, S \in \{\{(i^t_\gamma, 0)\}, \{(\gamma^{-1}(i^t_\gamma), 1)\}\}, \\ \mathcal{X}^{(i^{t+1}_\gamma)}_\gamma \cap F(I^{t+1}_0, I^{t+1}_1) \cap V(C \cup S) \neq \emptyset, \textit{ and} \\ \textit{there is } x \in \mathcal{X}^{(i^t_\gamma)}_\gamma \cap F(I^t_0, I^t_1) \cap V(C) \textit{ with } x_{i^t_\gamma} > \gamma(x)_{i^t_\gamma} \end{array} \right\}.$$

Proof. In this proof, for brevity we denote $F^t \coloneqq F(I^t_0, I^t_1)$, and $\mathcal{X}^t_\gamma \coloneqq \mathcal{X}^{(i^t_\gamma)}_\gamma$. In particular, because no fixing is applied at time t, we have $F^{t+1} = F^t$.

An index increasing event for permutation $\gamma \in \Pi$ does not impact any permutation but γ, so the update is correct for all $\delta \in \Pi \setminus \{\gamma\}$. For γ, we prove correctness by showing $E^{t+1}_\gamma \subseteq Y \cup Z$ and $E^{t+1}_\gamma \supseteq Y \cup Z$, respectively. Consistent with the index increasing event, we have $i^{t+1}_\gamma = i^t_\gamma + 1$, $I^{t+1}_0 = I^t_0$, and $I^{t+1}_1 = I^t_1$.

To prove the first inclusion, let $\bar{C} \in E^{t+1}_\gamma$ be a minimal i^{t+1}_γ-eq-conjunction. Then $\bar{C}$ is especially an i^t_γ-eq-conjunction (but not necessarily minimal).

On the one hand, suppose that $\bar{C}$ is a *minimal* i^t_γ-eq-conjunction, that is $\bar{C} \in E^t_\gamma$. We show that $\bar{C} \in Y$. Since $\bar{C}$ is an i^{t+1}_γ-eq-conjunction, all $x \in \hat{F}(C) \coloneqq F(I^t_0, I^t_1) \cap V(\bar{C})$ satisfy $x \preceq_{i^{t+1}_\gamma} \gamma(x)$ and there is $\tilde{x} \in \hat{F}(C)$ with $\tilde{x} =_{i^{t+1}_\gamma} \gamma(\tilde{x})$. As all $x \in \mathcal{X}^t_\gamma \cap F^t \cap V(\bar{C})$ have $x =_{i^t_\gamma} \gamma(x)$, all conditions of the set Y for $\bar{C} \in E^t_\gamma$ are satisfied, so $\bar{C} \in Y$.

On the other hand, suppose that $\bar{C}$ is no minimal i^t_γ-eq-conjunction. We show that $\bar{C} \in Z$. There is $C \in E^t_\gamma$ with $C \subsetneq \bar{C}$, such that all $x \in \mathcal{X}^t_\gamma \cap F^t \cap V(C)$ have $x =_{i^t_\gamma} \gamma(x)$. Because $\bar{C} \in E^{t+1}_\gamma$ and $C \subsetneq \bar{C}$, there exists $x \in \mathcal{X}^{t+1}_\gamma \cap F^{t+1} \cap V(C) \subseteq \mathcal{X}^t_\gamma \cap F^t \cap V(C)$ with $x \succ_{i^{t+1}_\gamma} \gamma(x)$, meaning that this vector x satisfies $x_{i^t_\gamma} > \gamma(x)_{i^t_\gamma}$. Hence, $C \in E^t_\gamma$ satisfies the last condition of set Z. We show the remaining conditions with the following claim:

Claim A.1. *Let $\bar{C} \in E^{t+1}_\gamma$, and $C \in E^t_\gamma$ with $C \subsetneq \bar{C}$. Suppose there exists a vector $x \in \mathcal{X}^t_\gamma \cap F^t \cap V(C)$ with $x_{i^t_\gamma} > \gamma(x)_{i^t_\gamma}$.* **(i)** *If $x_{i^t_\gamma} = \gamma(x)^t_{i_\gamma} = 0$ for all $x \in \mathcal{X}^{t+1}_\gamma \cap F^{t+1} \cap V(\bar{C})$, then $(i^t_\gamma, 0) \in \bar{C}$; and* **(ii)** *if $x_{i^t_\gamma} = \gamma(x)^t_{i_\gamma} = 1$ for all $x \in \mathcal{X}^{t+1}_\gamma \cap F^{t+1} \cap V(\bar{C})$, then $(\gamma^{-1}(i^t_\gamma), 1) \in \bar{C}$.*

Proof of claim. The proof for statements *(i.)* and *(ii.)* are analogous, so we only prove it for *(i.)*. Suppose that the precondition of *(i.)* holds, and let $\hat{x} \in \mathcal{X}^{t+1}_\gamma \cap F^{t+1} \cap V(\bar{C})$, such that $\hat{x} =_{i^t_\gamma + 1} \gamma(\hat{x})$ and $\hat{x}_{i^t_\gamma} = 0$. In particular, $(i^t_\gamma, 1) \notin \bar{C}$. For sake of contradiction, suppose that $(i^t_\gamma, 0) \notin \bar{C}$. Then $(i^t_\gamma, 0) \notin C$, as well. We find a contradiction for the three cases $i^t_\gamma = \gamma(i^t_\gamma)$, $i^t_\gamma < \gamma(i^t_\gamma)$ and $i^t_\gamma > \gamma(i^t_\gamma)$, together implying *(i.)*. Since there exists a vector $x \in \mathcal{X}^t_\gamma \cap F^t \cap V(C)$ with $x_{i^t_\gamma} > \gamma(x)_{i^t_\gamma}$, i^t_γ is no fixed point of γ, so $i^t_\gamma \neq \gamma(i^t_\gamma)$. If $i^t_\gamma < \gamma(i^t_\gamma)$, let $z \in F^t \cap V(\bar{C})$ with $z_i = \hat{x}_i$ for $i \neq i^t_\gamma$ and $z_{i^t_\gamma} = 1$. Then $z \succ_{i^{t+1}_\gamma} \hat{x} =_{i^{t+1}_\gamma} \gamma(\hat{x}) =_{i^{t+1}_\gamma} \gamma(z)$, where the last equality is a consequence of that the vector $\hat{x}$

and z only differ in entry i^t_γ, which is entry $\gamma(i^t_\gamma) > i^t_\gamma$ of the permuted vectors $\gamma(\hat{x})$ and $\gamma(z)$. This is not possible, as $\bar{C}$ is a i^{t+1}_γ-eq-conjunction. Last, if $i^t_\gamma > \gamma(i^t_\gamma)$, let $x \in \mathcal{X}^t_\gamma \cap F^t \cap V(C)$ with $x_{i^t_\gamma} = 1$, and let $z \in \mathcal{X}^t_\gamma \cap F^t \cap V(C)$ with $z_i = x_i$ for all $i \neq i^t_\gamma$, and $z_{i^t_\gamma} = 0$. Then $x =_{\gamma(i^t_\gamma)} z$ and $\gamma(x) =_{\gamma(i^t_\gamma)} \gamma(z)$, and $x_{\gamma(i^t_\gamma)} = \gamma(x)_{\gamma(i^t_\gamma)} = x_{i^t_\gamma} = 1$, while $1 = x_{\gamma(i^t_\gamma)} = z_{\gamma(i^t_\gamma)}$ and $\gamma(z_{\gamma(i^t_\gamma)}) = z_{i^t_\gamma} = 0$. Hence, $z \succ_{\gamma(i^t_\gamma)+1} \gamma(z)$, which contradicts that C is a valid i^t_γ-eq-conjunction. A contradiction follows in all cases, so $(i^t_\gamma, 0) \in \bar{C}$. ♢

By the claim, we thus have that $(i^t_\gamma, 0) \in \bar{C}$ or $(\gamma^{-1}(i^t_\gamma), 1) \in \bar{C}$. The two cases are analogous, so suppose without loss of generality that $(i^t_\gamma, 0) \in \bar{C}$. As C is an i^t_γ-eq-conjunction, $\hat{C} := C \cup \{(i^t_\gamma, 0)\} \subseteq \bar{C}$ is also an i^t_γ-eq-conjunction. In particular, it is also a i^{t+1}_γ-eq-conjunction, since all $x \in F(I^{t+1}_0, I^{t+1}_1) \cap V(\hat{C})$ have $x \preceq_{i^t_\gamma} \gamma(x)$ and $0 = x_{i^t_\gamma} \leq \gamma(x)_{i^t_\gamma}$, meaning that $x \preceq_{i^{t+1}_\gamma} \gamma(x)$. As $\hat{C} \subseteq \bar{C}$, there must also be one vector $x \in F(I^{t+1}_0, I^{t+1}_1) \cap V(\hat{C})$ with $x =_{i^{t+1}_\gamma} \gamma(x)$, that is $\mathcal{X}^{t+1}_\gamma \cap F^{t+1} \cap V(\hat{C}) \neq \emptyset$. Since $\bar{C} \in E^{t+1}_\gamma$ and $\hat{C} \subseteq \bar{C}$ is an i^{t+1}_γ-eq-conjunction, it must hold that $\bar{C} = \hat{C} = C \cup S$, where $S = \{(i^t_\gamma, 0)\}$. All remaining conditions hold, so $\bar{C} \in Z$. Together with the analogous case, this proves that $E^{t+1}_\gamma \subseteq Y \cup Z$. In the remainder we prove the remaining inclusion.

Let $\bar{C} \in Y$. Then $\bar{C} \in E^t_\gamma$, so it is a valid i^t_γ-eq-conjunction. Added to this, by the second condition, all $x \in \mathcal{X}^t_\gamma \cap F^t \cap V(\bar{C})$ have $x \preceq_{i^{t+1}_\gamma} \gamma(x)$, and by the third condition, there is an $x \in \mathcal{X}^t_\gamma \cap F^t \cap V(\bar{C})$ with $x =_{i^{t+1}_\gamma} \gamma(x)$. Thus, $\bar{C}$ is an (i^{t+1}_γ)-eq-conjunction, and it remains to show minimality. This, however, follows immediately, because for any $C' \subsetneq \bar{C}$ there exists $x \in \mathcal{X}^t_\gamma \cap F^t \cap V(C')$ with $x \succ_{i^t_\gamma} \gamma(x)$, so in particular $x \succ_{i^{t+1}_\gamma} \gamma(x)$. Hence, $\bar{C} \in E^{t+1}_\gamma$, showing that $Y \subseteq E^{t+1}_\gamma$.

Let $\bar{C} \in Z$, with representation $C \cup S$ for $C \in E^t_\gamma$, and $S = \{(i^t_\gamma, 0)\}$ or $S = \{(\gamma^{-1}(i^t_\gamma), 1)\}$. Denote $S = \{(j, b)\}$. Since there is $x \in \mathcal{X}^t_\gamma \cap F^t \cap V(C)$ with $x_{i^t_\gamma} > \gamma(x)_{i^t_\gamma}$, C is no i^{t+1}_γ-eq-conjunction. By additionally applying (j, b), one enforces $x_{i^t_\gamma} \leq \gamma(x)_{i^t_\gamma}$. Since C is an i^t_γ-eq-conjunction, all $x \in \mathcal{X}^t_\gamma \cap F^t \cap V(C \cup S)$ have $x =_{i^t_\gamma} \gamma(x)$ and $x_{i^t_\gamma} \leq \gamma(x)_{i^t_\gamma}$, so $x \preceq_{i^{t+1}_\gamma} \gamma(x)$. If $\mathcal{X}^t_\gamma \cap F^t \cap V(C \cup S) \neq \emptyset$, then there exists a vector $x \in \mathcal{X}^t_\gamma \cap F^t \cap V(C \cup S)$ with $x \succeq_{i^{t+1}_\gamma} \gamma(x)$ and $x \preceq_{i^{t+1}_\gamma} \gamma(x)$, so $x =_{i^{t+1}_\gamma} \gamma(x)$. This shows that $C \cup S$ is an i^{t+1}_γ-eq-conjunction. This is a minimal conjunction, by the following two arguments: First, by the last condition there is a vector $x \in V(C)$ with $x \succ_{i^{t+1}_\gamma} \gamma(x)$, so the fixing from S cannot be removed. Second, C is a minimal i^t_γ-eq-conjunction, so for any $C' \subsetneq C$ there is a vector $x' \in \mathcal{X}^t_\gamma \cap F^t \cap V(C')$ with $x' \succ_{i^t_\gamma} \gamma(x')$. Let $x \in \{0,1\}^n$ with $x_i = x'_i$ for $i \neq j$ and $x_j = b$. By the definition of Z there is a $y \in \mathcal{X}^t_\gamma \cap F^t \cap V(C)$ with $y_{i^t_\gamma} > \gamma(y)_{i^t_\gamma}$, implying $(j, 1-b) \notin C'$, so that $x \in F(I^t_0, I^t_1) \cap V(C' \cup S)$. Since the only difference between x and x' is entry j, a difference in the first $i^t_\gamma - 1$ entries between x and x' is only possible in the case $(j, b) = (\gamma^{-1}(i^t_\gamma), 1)$ and $j < i^t_\gamma$, which makes $x_j \geq x'_j$. Case $(j, b) = (i^t_\gamma, 0)$ does not affect the first $i^t_\gamma - 1$ entries of x and x'. Hence, $x \succeq_{i^t_\gamma} x'$. Analogously, $\gamma(x) \preceq_{i^t_\gamma} \gamma(x')$. Concluding, we have $x \succeq_{i^t_\gamma} x' \succeq_{i^t_\gamma} \gamma(x') \succeq_{i^t_\gamma} \gamma(x)$, showing the existence of $x \in \mathcal{X}^{t+1}_\gamma \cap F^{t+1} \cap V(C' \cup S)$ with $x \succ_{i^t_\gamma} \gamma(x)$. Hence, no fixing from $\bar{C} = C \cup S$ can be removed from the set to find a smaller i^{t+1}_γ-eq-conjunction, so $\bar{C} = C \cup S \in E^{t+1}_\gamma$, showing that $Z \subseteq E^{t+1}_\gamma$. Concluding with the previous paragraphs $E^{t+1}_\gamma = Y \cup Z$ follows. □

Proposition 2.15 (Updating inf-conjunctions for index increasing event). *Consider an index increasing event for permutation $\gamma \in \Pi$ at time t. Then, for $\delta \in \Pi \setminus \{\gamma\}$, we have $D^{t+1}_\delta = D^t_\delta$, and $D^{t+1}_\gamma = D^{t+1}_{\gamma,\text{eq}} \cup D^{t+1}_{\gamma,\text{inf}}$ with*

$$D^{t+1}_{\gamma,\text{eq}} = \left\{ C \cup S : \begin{array}{c} C \in E^t_\gamma,\ S \subseteq \{(i^t_\gamma, 0), (\gamma^{-1}(i^t_\gamma), 1)\},\ \gamma(i^t_\gamma) \neq i^t_\gamma, \textit{ and} \\ \textit{either } (i^t_\gamma, 0) \in S \textit{ or } x_{i^t_\gamma} = 0 \textit{ for all } x \in \mathcal{X}^{(i^t_\gamma)}_\gamma \cap F^t \cap V(C), \textit{ and} \\ \textit{either } (\gamma^{-1}(i^t_\gamma), 1) \in S \textit{ or } \gamma(x)_{i^t_\gamma} = 1 \textit{ for all } x \in \mathcal{X}^{(i^t_\gamma)}_\gamma \cap F^t \cap V(C) \end{array} \right\},$$

with $F^t := F(I^t_0, I^t_1)$, and $D^{t+1}_{\gamma,\text{inf}} = \{C \in D^t_\gamma : \text{for all } C' \in D^t_{\gamma,\text{eq}} \text{ holds } C \not\supseteq C'\}$.

Proof. Again, for brevity we denote $\mathcal{X}_\gamma^t := \mathcal{X}_\gamma^{(i_\gamma^t)}$ and $F^t := F(I_0^t, I_1^t)$. As the event at time t is an index increasing event, in particular $F^{t+1} = F^t$.

An index increasing event for permutation $\gamma \in \Pi$ does not impact any permutation but γ, so the update is correct for all $\delta \in \Pi \setminus \{\gamma\}$. For γ, we prove correctness in four parts: First, ***(i)*** all $\bar{C} \in D_\gamma^{t+1}$ that are no i_γ^t-inf-conjunction have $\bar{C} \in D_{\gamma,\text{eq}}^{t+1}$, then ***(ii)*** $D_{\gamma,\text{eq}}^{t+1} \subseteq D_\gamma^{t+1}$, after which we show that ***(iii)*** all $\bar{C} \in D_\gamma^{t+1}$ that are i_γ^t-inf-conjunction have $\bar{C} \in D_{\gamma,\text{inf}}^{t+1}$, and last ***(iv)*** $D_{\gamma,\text{inf}}^{t+1} \subseteq D_\gamma^{t+1}$. Part *(i.)* and *(iii.)* show $D_\gamma^{t+1} \subseteq D_{\gamma,\text{eq}}^{t+1} \cup D_{\gamma,\text{inf}}^{t+1}$, and part *(ii.)* and *(iv.)* show the converse inclusion, therewith proving the proposition. Consistent with the index increasing event, we have $i_\gamma^{t+1} = i_\gamma^t + 1$, $I_0^{t+1} = I_0^t$, and $I_1^{t+1} = I_1^t$.

To show *(i.)*, let $\bar{C} \in D_\gamma^{t+1}$ be a minimal i_γ^{t+1}-inf-conjunction, and suppose that $\bar{C}$ is no i_γ^t-inf-conjunction. Let $x \in \mathcal{X}_\gamma^t \cap F^t \cap V(\bar{C})$. Then $x \prec_{i_\gamma^{t+1}} \gamma(x)$ as $\bar{C}$ is an i_γ^{t+1}-inf-conjunction, and $x \succeq_{i_\gamma^t} \gamma(x)$ as $\bar{C}$ is no i_γ^t-inf-conjunction. Together, $x =_{i_\gamma^t} \gamma(x)$ must hold, and $x_{i_\gamma^t} < \gamma(x)_{i_\gamma^t}$. Hence, $\bar{C}$ is an i_γ^t-eq-conjunction.

If $\bar{C}$ is a minimal i_γ^t-eq-conjunction, i.e., $\bar{C} \in E_\gamma^t$, then for any $x \in \mathcal{X}_\gamma^t \cap F^t \cap V(\bar{C})$ holds $0 = x_{i_\gamma^t} < \gamma(x)_{i_\gamma^t} = 1$. Thus, the conditions of $D_{\gamma,\text{eq}}^{t+1}$ hold for $C = \bar{C} \in E_\gamma^t$ and $S = \emptyset$ (as $x_{i_\gamma^t} = 0$, $\gamma(x)_{i_\gamma^t} = 1$ for all $x \in \mathcal{X}_\gamma^t \cap F^t \cap V(\bar{C})$), that is $\bar{C} = C \cup S \in D_{\gamma,\text{eq}}^{t+1}$.

Otherwise, if $\bar{C}$ is a non-minimal i_γ^t-eq-conjunction, let $C \subsetneq \bar{C}$ with $C \in E_\gamma^t$. By minimality of $\bar{C} \in D_\gamma^{t+1}$, there exists an $x \in \mathcal{X}_\gamma^{t+1} \cap F^{t+1} \cap V(C)$ with $x \succeq_{i_\gamma^{t+1}} \gamma(x)$. Since $C \in E_\gamma^t$ also $x =_{i_\gamma^t} \gamma(x)$. Thus, $x_{i_\gamma^t} \geq \gamma(x)_{i_\gamma^t}$.

Suppose that $(i_\gamma^t, 0) \in \bar{C} \setminus C$. Since $\bar{C} \in D_\gamma^{t+1}$, all $x \in \mathcal{X}_\gamma^{t+1} \cap F^{t+1} \cap V(\bar{C} \setminus \{(i_\gamma^t, 0)\})$ have $x_{i_\gamma^t} = 1$, and by minimality also such a vector exists. By $C \subseteq \bar{C} \setminus \{(i_\gamma^t, 0)\} \subsetneq \bar{C}$ and $\mathcal{X}_\gamma^{t+1} \cap F^{t+1} \subseteq \mathcal{X}_\gamma^t \cap F^t$, we have $x \in \mathcal{X}_\gamma^{t+1} \cap F^{t+1} \cap V(\bar{C} \setminus \{(i_\gamma^t, 0)\}) \subseteq \mathcal{X}_\gamma^t \cap F^t \cap V(C)$. Hence, $(i_\gamma^t, 0) \in \bar{C} \setminus C$ implies that not all $x \in \mathcal{X}_\gamma^t \cap F^t \cap V(C)$ have $x_{i_\gamma^t} = 0$. Conversely, suppose that all $x \in \mathcal{X}_\gamma^t \cap F^t \cap V(C)$ have $x_{i_\gamma^t} = 0$. If $(i_\gamma, 0) \in \bar{C} \setminus C$, then by $\bar{C} \in D_\gamma^{t+1}$ there is an $x \in \mathcal{X}_\gamma^{t+1} \cap F^{t+1} \cap V(\bar{C} \setminus \{(i_\gamma, 0)\})$ with $x_{i_\gamma^t} = 1$, which contradicts $x \in \mathcal{X}_\gamma^{t+1} \cap F^{t+1} \cap V(\bar{C} \setminus \{(i_\gamma, 0)\}) \subseteq \mathcal{X}_\gamma^t \cap F^t \cap V(C)$. Hence, together with the previous part, we find that either $(i_\gamma^t, 0) \in \bar{C} \setminus C$ or that all $x \in \mathcal{X}_\gamma^t \cap F^t \cap V(C)$ have $x_{i_\gamma^t} = 0$. Analogously one can show that either $(\gamma^{-1}(i_\gamma^t), 1) \in \bar{C} \setminus C$, or that $\gamma(x)_{i_\gamma^t} = 1$ for all $x \in \mathcal{X}_\gamma^t \cap F^t \cap V(C)$.

Consistent with the definition of $D_{\gamma,\text{eq}}^{t+1}$, let $S \subseteq \{(i_\gamma^t, 0), (\gamma^{-1}(i_\gamma^t), 1)\}$ with $(i_\gamma^t, 0) \in S$ if there is an $x \in \mathcal{X}_\gamma^t \cap F^t \cap V(C)$ with $x_{i_\gamma^t} = 1$, and $(\gamma^{-1}(i_\gamma^t), 1) \in S$ if there is an $x \in \mathcal{X}_\gamma^t \cap F^t \cap V(C)$ with $\gamma(x)_{i_\gamma^t} = 0$. Then $C \cup S$ is an i_γ^{t+1}-inf-conjunction, because C is an i_γ^t-eq-conjunction, and the fixings from S additionally ensure that $x_{i_\gamma^t} < \gamma(x)_{i_\gamma^t}$ for all $x \in \mathcal{X}_\gamma^t \cap F^t \cap V(C \cup S)$. As a result of the previous paragraph, we thus have that $f_0 := (i_\gamma^t, 0) \in S$ if and only if $f_0 \in \bar{C} \setminus C$, and analogously that $f_1 := (\gamma^{-1}(i_\gamma^t), 1) \in S$ if and only if $f_1 \in \bar{C} \setminus C$. Hence, $S \subseteq \bar{C} \setminus C$, showing that $C \cup S \subseteq \bar{C}$. Since $\bar{C}$ is a minimal i_γ^{t+1}-inf-conjunction and $C \cup S$ is a i_γ^{t+1}-inf-conjunction it must thus hold that $\bar{C} = C \cup S$. This shows that $\bar{C} \in D_{\gamma,\text{eq}}^{t+1}$.

To prove *(ii.)*, let $\bar{C} \in D_{\gamma,\text{eq}}^{t+1}$ be represented by $C \cup S$ for $C \in E_\gamma^t$, and $S \subseteq \{(i_\gamma^t, 0), (\gamma^{-1}(i_\gamma^t), 1)\}$, where S is consistent with the definition. Since $C \in E_\gamma^t$, all $x \in \mathcal{X}_\gamma^t \cap F^t \cap V(C \cup S)$ have $x =_{i_\gamma^t} \gamma(x)$, and by the definition of S, also $x_{i_\gamma^t} < \gamma(x)_{i_\gamma^t}$. Hence, all $x \in \mathcal{X}_\gamma^t \cap F^t \cap V(C \cup S)$ have $x \prec_{i_\gamma^{t+1}} \gamma(x)$, such that $\mathcal{X}_\gamma^{t+1} \cap F^{t+1} \cap V(C \cup S) = \emptyset$, meaning that $\bar{C} = C \cup S$ is an i_γ^{t+1}-inf-conjunction. It is also minimal, since removing any element from $\bar{C}$ yields a non-empty set $\mathcal{X}_\gamma^{t+1} \cap F^{t+1} \cap V(\bar{C})$, as we show next. Note that, by the definition of S, that C and S are non-overlapping. Consider $f \in C$. Since $C \in E_\gamma^t$, there is a vector $x \in \mathcal{X}_\gamma^t \cap F^t \cap V(C \setminus \{f\})$ with $x \succ_{i_\gamma^t} \gamma(x)$. If $x \notin V(S)$, one can construct $y \in \mathcal{X}_\gamma^t \cap F^t \cap V((C \cup S) \setminus \{f\})$ by copying x and applying the fixings of S. The application of fixing $(i_\gamma^t, 0)$ does not affect the first $i_\gamma^t - 1$ entries of x, and can only make vector $\gamma(x)$ partially lexicographically smaller up to i_γ^t, as it changes a 1-entry to a 0-entry. Likewise, the application of $(\gamma^{-1}(i_\gamma^t), 0)$ does not affect the first $i_\gamma^t - 1$ entries of $\gamma(x)$ and can only make vector x partially lexicographically larger up to i_γ^t. Hence, $y \succeq_{i_\gamma^t} x \succ_{i_\gamma^t} \gamma(x) \succeq_{i_\gamma^t} \gamma(y)$, showing that $y \succ_{i_\gamma^t} \gamma(y)$, so in particular $y \succ_{i_\gamma^{t+1}} \gamma(y)$, meaning that $y \in \mathcal{X}_\gamma^{t+1} \cap F^{t+1} \cap V((C \cup S) \setminus \{f\})$. Conversely, consider $f \in S$, and by analogy between the two possibilities, suppose without loss of generality $f = (i_\gamma^t, 0) \in S$. By the definition of S, there exists $x \in \mathcal{X}_\gamma^t \cap F^t \cap V(C)$ with $x =_{i_\gamma^t} \gamma(x)$ and $x_{i_\gamma^t} = 1$, that is $x \succeq_{i_\gamma^{t+1}} \gamma(x)$. Also here, if $x \notin V(S \setminus \{f\})$, one can apply the fixing from $S \setminus \{f\}$ and find $y \in \mathcal{X}_\gamma^{t+1} \cap F^{t+1} \cap V((C \cup S) \setminus \{f\})$ with $y \succeq_{i_\gamma^{t+1}} \gamma(y)$. Hence, no i_γ^{t+1}-inf-conjunction could be found by dropping any fixing from $\bar{C} = C \cup S$, showing that $\bar{C} \in D_\gamma^{t+1}$, and thus $D_{\gamma,\text{eq}}^{t+1} \subseteq D_\gamma^{t+1}$.

Next, to show *(iii.)*, let $\bar{C} \in D_\gamma^{t+1}$ be a minimal i_γ^{t+1}-inf-conjunction that is also a i_γ^t-inf-conjunction. Since $\bar{C}$ is a minimal i_γ^{t+1}-inf-conjunction, for all $C \subsetneq \bar{C}$ there exists $x \in F(I_0^t, I_1^t) \cap V(C)$ with $x \succeq_{i_\gamma^{t+1}} \gamma(x)$, so especially with $x \succeq_{i_\gamma^t} \gamma(x)$. This shows that $\bar{C}$ is also a minimal i_γ^t-inf-conjunction, meaning that $\bar{C} \in D_\gamma^t$. Since by *(ii.)* we have $D_{\gamma,\text{eq}}^{t+1} \subseteq D_\gamma^{t+1}$, a contradiction with minimality of $\bar{C}$ at time $t+1$ would follow if there was a $C \in D_{\gamma,\text{eq}}^{t+1}$ with $C \subsetneq \bar{C}$. Hence, $\bar{C} \in D_{\gamma,\text{inf}}^{t+1}$. The combination of part *(i.)* and *(iii.)* shows $D_\gamma^{t+1} \subseteq D_{\gamma,\text{eq}}^{t+1} \cup D_{\gamma,\text{inf}}^{t+1}$.

Last, to show part *(iv.)*, let $\bar{C} \in D_{\gamma,\text{inf}}^{t+1}$. Then $\bar{C} \in D_\gamma^t$ so it is a minimal i_γ^t-inf-conjunction. In particular it is a i_γ^{t+1}-inf-conjunction. Suppose that it is no minimal i_γ^{t+1}-inf-conjunction. Then there is a $C \in D_\gamma^{t+1}$ with $C \subsetneq \bar{C}$. By the previous parts, $C \in D_{\gamma,\text{eq}}^{t+1} \cup D_{\gamma,\text{inf}}^{t+1}$. If $C \in D_{\gamma,\text{inf}}^{t+1}$, then $C \in D_\gamma^t$, and that contradicts minimality of $\bar{C} \in D_\gamma^t$. If $C \in D_{\gamma,\text{eq}}^{t+1}$, then this contradicts the definition of $\bar{C} \in D_{\gamma,\text{inf}}^{t+1}$. Since either case yields a contradiction, $\bar{C}$ must be a minimal i_γ^{t+1}-inf-conjunction, meaning that $\bar{C} \in D_\gamma^{t+1}$, proving $D_{\gamma,\text{inf}}^{t+1} \subseteq D_\gamma^{t+1}$. □

Proposition 2.17 (Updating inf-conjunctions for variable fixing event). *Consider a variable fixing event for fixing $f = (i, b) \in [n] \times \{0, 1\}$ at time t. For every $\gamma \in \Pi$ holds*

$$D_\gamma^{t+1} = \left\{ C \setminus \{f\} : \begin{array}{c} C \in D_\gamma^t, \textit{and} \\ \textit{for all } C' \in D_\gamma^t \textit{ holds } C' \setminus \{f\} \not\subsetneq C \setminus \{f\} \end{array} \right\}. \tag{2}$$

Proof. Denote the right-hand side of Equation (2) by H. We prove the correctness of Equation (2) by showing $D_\gamma^{t+1} \subseteq H$ and $D_\gamma^{t+1} \supseteq H$, respectively.

Let $\bar{C} \in D_\gamma^{t+1}$. As fixing f is applied and $\bar{C}$ is minimal, $f \notin \bar{C}$. On the one hand, suppose that $\bar{C}$ is an i_γ^t-inf-conjunction at time t. It is necessarily minimal, as otherwise there exists a $C \in D_\gamma^t$ with $C \subsetneq \bar{C}$. C is also an i_γ^t-inf-conjunction at time $t+1$, contradicting minimality of $\bar{C}$. Hence, $\bar{C} \in D_\gamma^t$. Suppose that $C' \in D_\gamma^t$ with $C' \setminus \{f\} \subsetneq \bar{C} \setminus \{f\} = \bar{C}$. Then $C' \setminus \{f\}$ is an i_γ^t-inf-conjunction at time $t+1$, and inclusionwise smaller than $\bar{C}$, contradicting $\bar{C} \in D_\gamma^{t+1}$. Consequently, $\bar{C} \setminus \{f\} = \bar{C} \in H$.

On the other hand, suppose that $\bar{C}$ is no i_γ^t-inf-conjunction at time t. As the only difference from time t to $t+1$ is the application of fixing f, $C \coloneqq \bar{C} \cup \{f\}$ is an i_γ^t-inf-conjunction at time t. This is also minimal, since $\bar{C}$ is no i_γ^t-inf-conjunction at time t, meaning that f cannot be removed from C, and $\bar{C}$ is a minimal i_γ^t-inf-conjunction at time $t+1$, meaning that no element from $\bar{C}$ can be removed from C, as well. Hence, $C \in D_\gamma^t$. Now suppose that $C' \in D_\gamma^t$ with $C' \setminus \{f\} \subsetneq C \setminus \{f\} = \bar{C}$. Then $C' \setminus \{f\}$ is an i_γ^t-inf-conjunction at time $t+1$, which violates the minimality of $\bar{C} \in D_\gamma^{t+1}$. Concluding, $C \setminus \{f\} = \bar{C} \in H$. Together this proves that $D_\gamma^{t+1} \subseteq H$.

Finally, to show the reverse inclusion, let $\bar{C} \in H$ represented by $C \setminus \{f\}$ for some $C \in D_\gamma^t$. Since C is an i_γ^t-inf-conjunction at time t, so is $\bar{C}$ one at time $t+1$. Hence, if $\bar{C} \notin D_\gamma^{t+1}$, there exists a $\bar{C}' \in D_\gamma^{t+1}$ with $\bar{C}' \subsetneq \bar{C}$. By the first part of the proof, there exists $C' \in D_\gamma^t$ with $\bar{C}' = C' \setminus \{f\}$, contradicting minimality of C. Consequently, $\bar{C} \in D_\gamma^{t+1}$ showing $D_\gamma^{t+1} \supseteq H$. □

Proposition 2.18 (Updating eq-conjunctions for variable fixing event). *Consider a variable fixing event for fixing $f = (i, b) \in [n] \times \{0, 1\}$ at time t. For every $\gamma \in \Pi$ holds*

$$E_\gamma^{t+1} = \left\{ C \setminus \{f\} : \begin{array}{c} C \in E_\gamma^t, \textit{and} \\ \textit{for all } C' \in E_\gamma^t \textit{ holds } C' \setminus \{f\} \not\subsetneq C \setminus \{f\}, \textit{ and} \\ \textit{for all } C' \in D_\gamma^t \textit{ holds } C' \setminus \{f\} \not\subseteq C \setminus \{f\} \end{array} \right\}. \tag{3}$$

Proof. Denote the right-hand side of Equation (3) by H. We prove the correctness of Equation (3) by showing $E_\gamma^{t+1} \subseteq H$ and $E_\gamma^{t+1} \supseteq H$, respectively.

First, let $\bar{C} \in E_\gamma^{t+1}$. Suppose we have already established the existence of $C \in E_\gamma^t$ with $\bar{C} = C \setminus \{f\}$. We show that in this case the conditions of the right-hand side of Equation (3) hold. If there is a $C' \in E_\gamma^t$ with $C' \setminus \{f\} \subsetneq C \setminus \{f\} = \bar{C}$, then $C' \setminus \{f\}$ is also an i_γ^t-eq-conjunction at time $t+1$, which is not possible as $\bar{C}$ is inclusionwise minimal. Likewise, if there is $C' \in D_\gamma^t$ with $C' \setminus \{f\} \subseteq C \setminus \{f\} = \bar{C}$, then $\bar{C}$ is both an i_γ^t-inf-conjunction at time $t+1$ and an i_γ^t-eq-conjunction at time $t+1$, which is not possible.

We turn the missing proof of existence of $C \in E_\gamma^t$. On the one hand, suppose that $\bar{C}$ is an i_γ^t-eq-conjunction at time t. Analogous to the proof of Proposition 2.17, we conclude that $\bar{C}$ is necessarily a minimal i_γ^t-eq-conjunction, so $\bar{C} \in E_\gamma^t$. On the other hand, suppose that $\bar{C}$ is no i_γ^t-eq-conjunction at time t. Then $C \coloneqq \bar{C} \cup \{f\}$ is an i_γ^t-eq-conjunction at time t. This is minimal, as removing f from C yields $\bar{C}$, which is no i_γ^t-eq-conjunction at time t, and if a valid i_γ^t-eq-conjunction at time t is found when removing any element from $\bar{C}$, then this would violate minimality of $\bar{C}$ at time $t+1$. Hence, $C \in E_\gamma^t$. So in both cases there is a $C \in E_\gamma^t$ with $\bar{C} = C \setminus \{f\}$. Consequently $E_\gamma^{t+1} \subseteq H$.

Second and last, to show the reverse inclusion, let $\bar{C} \in H$ represented by $C \setminus \{f\}$ for some $C \in E_\gamma^t$. As C is an i_γ^t-eq-conjunction at time t, also $\bar{C}$ is an i_γ^t-eq-conjunction at time $t+1$. Suppose that $\bar{C} \notin E_\gamma^{t+1}$, meaning that minimality does not hold. Then there is $\bar{C}' \in E_\gamma^{t+1}$ with $\bar{C}' \subsetneq \bar{C}$. By $E_\gamma^{t+1} \subseteq H$, we have that $\bar{C}' = C' \setminus \{f\}$ for some $C' \in E_\gamma^t$. But then $C' \setminus \{f\} = \bar{C}' \subsetneq \bar{C} = C \setminus \{f\}$ contradicts $\bar{C} \in H$. Hence, $\bar{C} \in E_\gamma^{t+1}$, and thus $E_\gamma^{t+1} \supseteq H$. $\square$

B Supplements

This appendix provides the per-instance results of the experiments described in Section 4.

Table 6: 3-edge coloring on flower snark graphs J_n, original relabeling.

	nosym		gen		group		nopeek		peek	
instance	time(s)	S	time(s)	S	time(s)	S	time(s)	S	time(s)	S
3	0.01	5	0.00	5	0.00	5	0.00	5	0.00	5
5	0.02	5	0.01	5	0.01	5	0.01	5	0.01	5
7	0.37	5	0.03	5	0.03	5	0.03	5	0.03	5
9	0.65	5	0.20	5	0.19	5	0.17	5	0.17	5
11	2.68	5	0.58	5	0.53	5	0.55	5	0.55	5
13	16.51	5	1.39	5	1.21	5	1.32	5	1.34	5
15	48.17	5	5.29	5	5.11	5	4.68	5	4.86	5
17	95.36	5	15.98	5	13.15	5	12.08	5	12.32	5
19	198.58	5	30.62	5	20.92	5	28.42	5	29.56	5
21	339.21	5	51.57	5	41.70	5	40.66	5	40.84	5
23	3141.73	4	52.46	5	59.23	5	58.71	5	57.97	5
25	7200.00	0	71.86	5	73.22	5	86.04	5	86.84	5
27	7200.00	0	81.67	5	110.99	5	132.82	5	134.37	5
29	7200.00	0	290.86	5	317.28	5	294.04	5	229.28	5
31	7200.00	0	297.63	5	348.92	5	423.39	5	425.73	5
33	7200.00	0	1659.06	2	708.61	4	324.87	5	349.46	5
35	7200.00	0	1177.03	4	3689.75	2	1834.47	3	1356.33	3
37	7200.00	0	1588.07	3	3603.63	2	3701.40	2	2523.97	2
39	7200.00	0	2122.42	2	3375.47	1	3797.39	1	3277.88	2
41	7200.00	0	3798.78	1	2679.75	2	5067.89	1	1923.04	3
43	7200.00	0	3553.34	1	6878.06	1	5734.13	2	6885.16	1
45	7200.00	0	7200.00	0	7200.00	0	4892.76	1	2481.88	3
47	7200.00	0	7200.00	0	7200.00	0	2322.81	2	2507.41	2
49	7200.00	0	7200.00	0	7200.00	0	4666.12	1	4616.06	1
All instances combined	730.78	54	172.35	88	187.56	87	169.79	93	153.23	97
Total time	136:26:10		69:19:48		74:10:26		63:51:23		54:15:29	
Symmetry time	0:00:00		1:37:28		2:18:15		2:20:04		9:03:07	
Percentage time	0.0%		2.3%		3.1%		3.7%		16.7%	

Table 7: 3-edge coloring on flower snark graphs J_n, max-relabeling.

	nosym		gen		group		nopeek		peek	
instance	time(s)	S	time(s)	S	time(s)	S	time(s)	S	time(s)	S
3	–		0.00	5	0.00	5	0.00	5	0.00	5
5	–		0.02	5	0.01	5	0.01	5	0.01	5
7	–		0.05	5	0.04	5	0.04	5	0.04	5
9	–		0.44	5	0.34	5	0.35	5	0.34	5
11	–		0.80	5	0.80	5	0.62	5	0.58	5
13	–		2.52	5	1.58	5	1.29	5	1.35	5
15	–		8.88	5	4.48	5	4.92	5	4.22	5
17	–		28.51	5	12.36	5	11.17	5	10.82	5
19	–		60.61	5	31.00	5	35.13	5	30.12	5
21	–		95.28	5	55.67	5	52.78	5	52.47	5
23	–		143.54	5	108.25	5	84.91	5	108.19	5
25	–		215.28	5	122.76	5	170.44	5	140.46	5
27	–		1511.94	3	340.70	5	211.61	5	290.88	5
29	–		4975.37	2	1180.29	3	529.31	5	481.28	5
31	–		7200.00	0	6579.03	1	1922.91	4	1381.55	3
33	–		7200.00	0	7200.00	0	3435.54	2	4207.49	2
35	–		7200.00	0	4618.61	1	6872.39	1	4162.52	2
37	–		7200.00	0	7200.00	0	7200.00	0	6535.44	1
[We omitted the results for $n \in \{39, \ldots, 49\}$ from this table, since no setting could solve the instance within the time limit.]										
All instances combined	–		407.02	65	312.93	70	278.00	77	270.19	78
Total time	–		113:32:08		102:47:17		94:51:24		90:47:44	
Symmetry time	–		2:41:02		3:31:28		3:02:04		16:23:01	
Percentage time	–		2.4%		3.4%		3.2%		18.0%	

Table 8: 3-edge coloring on flower snark graphs J_n, `min`-relabeling.

	`nosym`		`gen`		`group`		`nopeek`		`peek`	
instance	time(s)	S	time(s)	S	time(s)	S	time(s)	S	time(s)	S
3	–		0.00	5	0.00	5	0.00	5	0.00	5
5	–		0.01	5	0.01	5	0.02	5	0.02	5
7	–		0.03	5	0.03	5	0.03	5	0.03	5
9	–		0.05	5	0.05	5	0.07	5	0.07	5
11	–		0.69	5	0.55	5	0.56	5	0.55	5
13	–		1.22	5	1.38	5	1.14	5	1.14	5
15	–		2.52	5	1.96	5	1.59	5	1.61	5
17	–		7.77	5	5.17	5	3.99	5	3.38	5
19	–		19.32	5	13.54	5	14.61	5	14.21	5
21	–		28.24	5	31.52	5	28.03	5	28.31	5
23	–		51.41	5	46.26	5	35.49	5	38.36	5
25	–		73.62	5	62.88	5	61.79	5	65.63	5
27	–		73.12	5	83.37	5	85.86	5	88.03	5
29	–		99.74	5	95.35	5	112.66	5	117.75	5
31	–		167.98	5	218.68	5	145.33	5	125.67	5
33	–		179.86	5	250.65	5	264.39	5	192.32	5
35	–		280.75	5	441.17	5	241.95	5	399.40	5
37	–		233.34	5	1124.95	3	1685.18	2	863.11	4
39	–		1333.45	3	4479.43	1	3955.85	1	5817.81	1
41	–		1278.07	3	4517.44	1	7200.00	0	2239.94	2
43	–		3865.27	1	4249.17	2	3396.23	1	3527.04	1
45	–		1519.78	3	7200.00	0	3385.45	2	4479.20	1
47	–		4357.15	1	3742.17	2	7200.00	0	4414.05	1
49	–		4161.40	1	6796.76	1	6900.11	1	7200.00	0
All instances combined	–		97.99	102	131.58	95	127.48	92	119.67	95
Total time	–		39:18:45		58:01:08		60:22:17		55:16:02	
Symmetry time	–		1:07:30		2:06:51		2:02:49		3:48:44	
Percentage time	–		2.9%		3.6%		3.4%		6.9%	

Table 9: 3-edge coloring on flower snark graphs J_n, `respect`-relabeling.

	`nosym`		`gen`		`group`		`nopeek`		`peek`	
instance	time(s)	S	time(s)	S	time(s)	S	time(s)	S	time(s)	S
3	–		0.00	5	0.00	5	0.00	5	0.00	5
5	–		0.01	5	0.01	5	0.01	5	0.02	5
7	–		0.03	5	0.03	5	0.03	5	0.03	5
9	–		0.21	5	0.23	5	0.21	5	0.21	5
11	–		0.59	5	0.50	5	0.51	5	0.50	5
13	–		1.06	5	1.39	5	1.00	5	0.99	5
15	–		3.93	5	2.56	5	3.17	5	3.26	5
17	–		9.01	5	7.51	5	8.36	5	7.75	5
19	–		23.34	5	15.71	5	16.11	5	16.60	5
21	–		48.78	5	30.36	5	30.54	5	33.10	5
23	–		66.14	5	47.37	5	45.44	5	44.85	5
25	–		85.33	5	66.92	5	73.00	5	74.54	5
27	–		94.18	5	99.08	5	92.91	5	112.44	5
29	–		150.11	5	167.78	5	157.78	5	232.94	5
31	–		315.77	5	208.05	5	399.83	5	226.83	5
33	–		1130.03	4	882.55	5	599.91	5	662.99	5
35	–		1401.11	5	1723.64	3	3068.62	3	1429.65	4
37	–		7200.00	0	6042.10	1	1375.05	3	4953.76	2
39	–		5072.36	1	3873.88	1	5507.11	1	4105.52	1
41	–		5875.18	1	4071.27	1	7200.00	0	7200.00	0
43	–		4127.42	1	7200.00	0	7200.00	0	7200.00	0
45	–		7200.00	0	7200.00	0	7200.00	0	7200.00	0
47	–		4642.87	1	7200.00	0	6160.08	1	7200.00	0
49	–		7200.00	0	7200.00	0	7200.00	0	7200.00	0
All instances combined	–		184.44	88	173.41	86	174.32	88	178.46	87
Total time	–		71:53:32		72:39:08		70:44:12		72:09:40	
Symmetry time	–		1:53:37		2:27:03		2:24:02		4:13:49	
Percentage time	–		2.6%		3.4%		3.4%		5.9%	

Table 10: Relevant MIPLIB 2010 and MIPLIB 2017 benchmark instances, original relabeling.

	nosym		gen		group		nopeek		peek	
instance	time(s)	S	time(s)	S	time(s)	S	time(s)	S	time(s)	S
cod105	7200.04	0	65.94	5	61.84	5	59.14	5	57.45	5
cov1075	4761.35	5	117.17	5	47.48	5	32.71	5	33.15	5
fastxgemm-n2r6s0t2	1628.36	5	286.01	5	209.40	5	141.51	5	145.82	5
fastxgemm-n2r7s4t1	6394.82	2	971.15	5	812.15	5	768.81	5	776.56	5
neos-1324574	6251.90	5	2398.62	5	2080.58	5	2064.22	5	2060.96	5
neos-3004026-krka	129.58	5	262.07	5	144.61	5	685.34	4	1286.86	4
neos-953928	2332.51	5	1471.52	5	1166.06	5	975.68	5	975.07	5
neos-960392	850.24	5	1954.62	5	2278.05	5	2451.05	4	2454.52	4
supportcase29	282.05	5	196.09	5	253.30	5	388.96	5	662.54	5
wachplan	2713.70	5	906.09	5	939.21	5	849.65	5	849.25	5
All instances combined	1853.78	42	491.69	50	407.76	50	449.34	48	506.23	48
Total time	45:46:14		13:24:05		12:12:07		15:57:33		18:43:22	
Symmetry time	0:00:00		0:03:44		0:03:47		0:06:56		1:56:05	
Percentage time	0.0%		0.5%		0.5%		0.7%		10.3%	

Table 11: Relevant MIPLIB 2010 and MIPLIB 2017 benchmark instances, max-relabeling.

	nosym		gen		group		nopeek		peek	
instance	time(s)	S	time(s)	S	time(s)	S	time(s)	S	time(s)	S
cod105	–		6635.01	3	1450.64	5	1563.47	5	1629.04	5
cov1075	–		883.55	5	156.74	5	157.80	5	194.83	5
fastxgemm-n2r6s0t2	–		312.75	5	229.14	5	145.76	5	151.16	5
fastxgemm-n2r7s4t1	–		975.69	5	813.14	5	777.50	5	787.83	5
neos-1324574	–		1847.38	5	1473.25	5	1201.85	5	1288.09	5
neos-3004026-krka	–		485.01	5	746.75	4	519.21	4	417.15	4
neos-953928	–		518.51	5	426.36	5	1219.34	5	927.31	5
neos-960392	–		1855.01	4	1475.19	5	1740.15	5	1799.96	5
supportcase29	–		578.87	5	1888.74	4	1060.72	4	1679.39	5
wachplan	–		1965.03	5	2125.04	5	1767.51	5	1691.19	5
All instances combined	–		1061.29	47	812.31	48	771.56	48	796.69	49
Total time	–		24:29:35		19:32:41		19:18:10		17:59:25	
Symmetry time	–		0:04:15		0:08:27		0:11:46		1:31:29	
Percentage time	–		0.3%		0.7%		1.0%		8.5%	

Table 12: Relevant MIPLIB 2010 and MIPLIB 2017 benchmark instances, min-relabeling.

	nosym		default		symre		nopeek		peek	
instance	time(s)	S	time(s)	S	time(s)	S	time(s)	S	time(s)	S
cod105	–		6724.34	3	1450.69	5	1541.47	5	1629.39	5
cov1075	–		419.35	5	144.54	5	200.75	5	195.73	5
fastxgemm-n2r6s0t2	–		324.24	5	199.63	5	150.58	5	168.10	5
fastxgemm-n2r7s4t1	–		969.75	5	812.81	5	779.84	5	789.20	5
neos-1324574	–		1812.54	5	1962.48	5	2351.54	5	1980.54	5
neos-3004026-krka	–		237.08	5	131.02	5	1366.37	3	511.05	4
neos-953928	–		457.24	5	828.82	5	534.57	5	535.76	5
neos-960392	–		2166.11	5	1900.46	5	1799.27	5	1825.84	5
supportcase29	–		440.40	5	282.07	5	753.98	4	179.23	5
wachplan	–		2020.76	5	1702.72	5	1873.75	5	1844.22	5
All instances combined	–		901.65	48	612.10	50	837.48	47	657.29	49
Total time	–		23:29:14		14:28:11		20:57:14		16:24:51	
Symmetry time	–		0:05:54		0:04:36		0:08:17		0:22:54	
Percentage time	–		0.4%		0.5%		0.7%		2.3%	

Table 13: Relevant MIPLIB 2010 and MIPLIB 2017 benchmark instances, respect-relabeling.

	nosym		gen		group		nopeek		peek	
instance	time(s)	S	time(s)	S	time(s)	S	time(s)	S	time(s)	S
cod105	–		6729.60	3	1449.94	5	1561.65	5	1629.89	5
cov1075	–		847.73	5	170.26	5	158.31	5	213.03	5
fastxgemm-n2r6s0t2	–		263.54	5	180.64	5	139.46	5	147.08	5
fastxgemm-n2r7s4t1	–		970.69	5	815.20	5	777.01	5	788.21	5
neos-1324574	–		2688.34	5	2216.11	5	1836.15	5	1759.82	5
neos-3004026-krka	–		374.60	4	578.92	4	189.27	5	418.20	4
neos-953928	–		1013.14	5	1710.39	5	1351.12	5	1346.81	5
neos-960392	–		1717.12	5	2152.39	5	2101.22	5	2096.46	5
supportcase29	–		269.59	5	272.41	5	420.73	5	467.09	5
wachplan	–		1019.60	5	1017.95	5	928.59	5	947.01	5
All instances combined	–		970.19	47	743.01	49	639.20	50	723.93	49
Total time	–		25:20:14		17:57:44		14:57:39		16:48:27	
Symmetry time	–		0:04:26		0:19:34		0:05:03		0:55:03	
Percentage time	–		0.3%		1.8%		0.6%		5.5%	