Symmetric lexicographic subset reverse search for the enumeration of circuits, cocircuits, and triangulations up to symmetry

Preprint 1

Jörg Rambau
University of Bayreuth
Germany

January 19, 2026

Abstract

This paper introduces, analyzes, and applies three variants of the enumeration framework symmetric lexicographic subset reverse search for the enumeration of subsets of a finite set up to symmetry. The framework is implemented in detail for three applications: cocircuits, circuits, and triangulations of point configurations. There are two new methods presented and analyzed to check the lexicographic minimality of a subset in its orbit: the critical-element method and the modified switch-table method. Moreover, new application-dependent methods to reduce the number of necessary enumeration nodes are introduced: rank-pruning for cocircuits and lex-pruning for triangulations. For circuits, a compact data structure, the column representation matrix, is introduced that allows the detection of signed circuits only by admissible column operations. With a C++-implementation of the ideas in the software package `TOPCOM`, in all three applications known benchmarks can be computed faster by a large margin, and new numbers, among others the number of cocircuits of the 9-cube, the number of circuits of the 8-cube, and the number of all triangulations of the product of a 5- and a 3-simplex, can be computed for the first time. For triangulations, a new method to enumerate triangulations with special properties allows to avoid infeasible triangulations directly in the enumeration process, which reduces the effort drastically compared to the enumeration of all triangulations and eliminating the infeasible triangulations in post-processing. For the first time, this method is used to enumerate of all triangulations with prescribed symmetry for instances, where the enumeration of all triangulations would be out of reach today. This is used, e.g., for a computational proof that all triangulations with order-five cyclic symmetry of the product of two 4-simplices are connected to a regular triangulation in the flip-graph. Moreover, for the first time non-subregular triangulations have been found in the 4-dilated 3-simplex and the 3-dilated 4-simplex.

Acknowledgement

The author is very grateful to Michael Joswig and Lars Kastner who provided the exact chirotope of the regular dodecahedron, computed by `polymake`. Chirotope data makes possible the exact computation of all triangulations of a point configurations even if there are no rational coordinates for it. Moreover, the author thanks Lukas Kühne who brought the importance of central and centrally symmetric triangulations of the centered full root polytope to the author's attention. The author also thanks Lisa Lamberti and Komei Fukuda for communicating the (at that time open) question about the number of circuits of the 6-cube, which led to the simultaneous consideration of triangulations and circuits, which was beneficial for both applications.

Contents

List of Tables

List of Algorithms

1 Introduction

This paper systematically studies how to enumerate subsets of a finite set up to symmetry based on a specialization of the reverse-search paradigm to orbits of subsets. In order to demonstrate the benefits of this approach, three applications are presented: for a point or vector configuration, enumerate up to symmetry all its cocircuits, all its circuits, and all its triangulations, respectively. In all three applications the scales of problem instances that can be handled are extended significantly.

Fast enumeration in general is important also in optimization: in combination with a dual-bound scheme, it gives rise to a branch-and-bound optimization algorithm. In dynamic column-generation, the pricing problem is sometimes attacked by enumerative algorithms (see, e.g., [18]). Whenever symmetries are present, an enumeration up to symmetry is the appropriate procedure. This is particularly important for large symmetry groups in branch-and-bound (like the $n!$ symmetries emerging by re-indexing n equivalent decision variables), since the $n!$ branches containing the equivalent optimal solutions can never be pruned.

The reverse-search paradigm is appealing because it generates an enumeration *tree*. An enumeration based on organizing the objects to be enumerated in a tree has two main advantages: first, it can be implemented in a memory efficient way by depth-first-search in the enumeration tree; second, it can be parallelized easily in a lock-free fashion. For example, enumeration based on the reverse-search paradigm can become faster after parallelization than alternative algorithms that are faster without parallelization [3].

A general account for the complexity of a variety of enumeration algorithms can be found in [22], where the objects to be enumerated are considered as the results of certain abstract closure operations. Symmetries are not considered.

The top-level enumeration method in this paper can be seen as a specialization of *reverse search* [2] to subsets up to symmetry. Reverse search has been specialized to the enumeration of orbits in [13] and parallelized in [3]. The specialization of reverse-search to feasible subsets of a finite set is probably folklore, but a specialization exploiting the lexicographic ordering of subsets to enumerate subset *orbits* was first formalized (in a different language) in [24] with applications for very large symmetry groups, where not all group elements can be held in memory. Though examples show the potential of the resulting algorithm, no theoretical analysis of the speed-up compared to naive approaches was provided. In this paper, variants of the same basic idea are designed, analyzed, and exploited in three example applications.

The first application in this paper is concerned with the enumeration of *signed cocircuits* of a point or vector configuration. Cocircuits in, e.g., hypercubes are in connection to the various ways binary data can be weakly linearly separated and therefore related to topics in data science. In [1] all hyperplanes spanned by vertices of the d-dimensional hypercube $\mathbf{C}_d$ were classified and enumerated up to $d = 8$ (12 days of cpu time on a computer that was fast according to the standards of that time). The authors estimated a cpu time of 35 years for $d = 9$ for their method based on the classification of normal vectors. There seems to be no documented algorithm or code enumerating the hyperplanes in a general point or vector configuration. In this paper, all hyperplanes for, e.g., the 9-cube are enumerated up to symmetry for the first time.

The second application in this paper is concerned with the enumeration of *signed circuits* of a point or vector configuration. This is the dual problem to the first one (in the oriented-matroid sense, see [4]). Circuits are in close connection with minimal infeasible subsystems and sparse kernel elements, which are related to compressed sensing. There is an incremental-polynomial-time algorithm for the enumeration of (unsigned) circuits of a matroid in [17] based on the exchange axiom for circuits, which improves on older algorithms like the one in [23] based on using bases for constructing circuits. However, it is not clear how symmetries could be exploited in this algorithm. Moreover, the nature of the exchange axiom shows that the worst-case time to add one more circuit is at least quadratic in the number of already computed circuits, which seems prohibitive at least for the larger examples in this paper. In this paper, all circuits for, e.g., the 8-cube are enumerated up to symmetry for the first time.

The third application in this paper is concerned with the enumeration of *triangulations* of a point or vector configuration. See [8] for background on triangulations and why their enumeration is interesting. Recently, a parallel enumeration up to symmetry of all *subregular* triangulations was suggested in [14]. A triangulation is *subregular*, if it can be flipped to a regular-triangulation by *upflips*, i.e., flips that lexicographically increase the GKZ-vector. Hence, whether or not a triangulation is subregular usually depends on the order of points in the input. The presented computational results were achieved based on the freely available package `mptopcom`. It enumerates all subregular triangulations up to symmetry. It significantly extends the scales of instances that can be handled compared to the earlier `TOPCOM` [25], which itself is a large step forward from de Loera's pioneering `maple`-code `PUNTOS` from his thesis [6]. The code `mptopcom` has later been specialized for cyclic polytopes to generate some new numbers [15] extending the computational results in [25, 27]. This was possible although for `mptopcom` symmetries must conserve simplex volumes, so that the standard cyclic polytopes do not have valid symmetries in that stronger sense.

All mentioned enumeration algorithms for triangulations are *flip-based*, i.e., they explore the flip graph of triangulations, where two triangulations share an edge if they are connected by a *bistellar flip*. This is a generalization of swapping diagonals in a convex quadrilateral in dimension two (see [8] for a definition in all dimensions). Since Santos's triangulation without flips in [29] it is known that flip-based algorithms may not find all triangulations in general. `TOPCOM` [25] was the first published software to enumerate all triangulations by building them simplex-by-simplex, which will be called an *extension-based* algorithm. However, the instances that could be handled were only toy-size examples. In [13] the extension-based enumeration of triangulations was reduced to the enumeration of maximal cliques in the *proper-intersection graph* of all simplices. However, no computational results were given. And the results in this paper show evidence for the fact that a pure max-clique enumeration cannot be efficient: there are too many maximal cliques that correspond to a *maximal incomplete triangulation*. This is a set of simplices with proper intersections that is not a triangulation and that cannot be extended (for exact definitions see Section 9). The enumeration of all triangulations corresponds to an enumeration of all vertices of the *universal polytope* [7]; since no complete outer description of this polytope is available, the vertex enumeration problem for it is not straight-forward. In this paper, all triangulations of, e.g., the regular dodecahedron or the product of a 5- and a 3-simplex are enumerated up to symmetry for the first time.

Besides the fact that extension-based algorithms reach all triangulations, there is one other motivation for them: If the search shall be restricted to triangulations using only *special simplices* (like unimodular simplices or simplices not containing any points of the configuration other than their vertices), then the flip-based algorithms have to explore the whole flip-graph anyway and filter ex-post by the wanted triangulations (since the subgraph of all wanted triangulations may be disconnected even for easy examples), whereas an extension-based algorithm can exclude the unwanted simplices right from the beginning. Moreover, a new method is presented that can exclude pairs of simplices from consideration that cannot co-exist in a triangulation with *prescribed symmetry*. In this paper, e.g., all central and centrally symmetric triangulations of the full root polytope in ambient 6-space are enumerated for the first time, while the enumeration of all its triangulations seems currently out of reach by far.

The overall contributions of this paper are both incremental and original. The top-level method used in this paper consists of a generic enumeration framework for the enumeration of all orbits of a *downset*, i.e., a set of subsets closed under taking subsets. This framework is called *Symmetric Lexicographic Subset Reverse Search* (`SymLexSubsetRS`) in this paper. Three variants of `SymLexSubsetRS` are studied: enumerate orbits of *maximal* elements *in* a downset, enumerate orbits of *minimal* elements *not in* a downset, and enumerate orbits of *feasible* antichains *in* a downset. The generic algorithm `SymLexSubsetRS` and the variant for feasible subsets in this paper – though developed independently – are essentially identical to the proposed algorithm in [24] for the enumeration of set orbits. The methods in this paper can be seen as variants, refinements, and new specializations of it and its subroutines. Without these incremental and original achievements, the new results would not have been possible. All applications have been implemented in the `TOPCOM` package, which is freely available under the Gnu Public Licence on the author's webpage.

The genuinely original contributions of this paper are twofold: for the general framework, two alternative checks of lexicographic minimality of a subset in its orbit (called the *lex-min check*) are proposed; for the particular applications, new methods are presented for recognizing that a subset cannot be extended to a feasible subset by adding larger elements (called the *lex-ext check*).

Concerning the *lex-min check*, the first new alternative is the *critical-element method*. It is based on new theory presented in Section 3. This alternative is mainly interesting for cases in which the symmetry group is of moderate order, i.e., is given as a list of all permutations in it, and of a degree in the same order of magnitude, i.e., there are at least as many elements as there are permutations. The enumeration of all triangulations usually fits into this scheme.

The second alternative is the *modified switch-table method*. It combines the ideas in [24] with the switch-table method in [14]. This alternative works best for symmetry groups whose order is large compared to the degree. The enumeration of circuits and cocircuits is an appropriate use-case.

In order to assess the efficiency of each method a-priori, a hyper-amortized analysis on uniform inputs is presented. Roughly speaking: for small order and large degree, the critical-element method is faster. For large order and small degree, the modified switch-table method is faster. This analysis sheds light on why, in computational experiments, for triangulations of hypercubes the critical-element method is faster, whereas for (co-)circuits of hypercubes the modified switch-table method wins. This constitutes the first ever theoretical analysis of algorithms based on switch-tables.

Concerning the *lex-ext checks* for the applications, the new rank-based *rank-pruning* accelerates the enumeration of cocircuits up to symmetry. This allowed the first ever enumeration of all hyperplanes in the 9-cube $\mathbf{C}_9$ up to symmetry in less than 14 hours. It is unclear how fast the code from [1] would run on today's computers. However, `TOPCOM`'s enumeration algorithm goes beyond the method in [1] anyway, since it works for general configurations and does not exploit any theory about cubes.

For the enumeration of circuits no effective lex-ext check was found so far. Still, the algorithm could compute some new numbers, among them the numbers of circuits up to symmetry of the hypercubes $\mathbf{C}_6$, $\mathbf{C}_7$, and $\mathbf{C}_8$. Note that in order to enumerate circuits one can also enumerate cocircuits in the *Gale-transform* (see [8] for more background on this). Whether or not this is faster or slower usually depends on the rank and the corank of the configuration. Having specialized algorithms for both means that one can pick the respective faster strategy.

For the enumeration of triangulations up to symmetry, the new lex-ext check *lex-pruning* is the single most important progress. It is based on the property of any triangulation that each interior facet of a simplex is covered by another simplex [8, Cor. 4.1.32]. From this one can derive the rather tight lex-ext check *full-pruning*. A weaker variant is *strong pruning*, which can be implemented more easily. The new lex-ext check lex-pruning heavily exploits the lexicographic ordering of all simplices and their interior facets in all data structures involved. While strong-pruning has to perform many subset operations, lex-pruning only compares two certain integers. Still it is almost as effective as strong-pruning. The numbers of all triangulations of, e.g., the dodecahedron and the product of a 5- and a 3-simplex could be computed this way for the first time in a couple of days.

For the enumeration of triangulations up to symmetry with a prescribed automorphism group, the new method based on *group-feasible simplices with group-proper intersection* for the first time allows to restrict the search space for triangulations directly to the symmetric triangulations, thereby avoiding the need to implicitly enumerate all triangulations during the process. This contribution opens the door for the application of a variant of the *Kramer-Mesner method* [19] from design theory (restrict the search to symmetric designs) to otherwise intractable search problems in the set of all triangulations. This way, all central and centrally symmetric triangulations of the full root polytope in ambient 6-space have been found in about an hour.

Symmetric lexicographic subset reverse search can be used to count, enumerate, or to list objects up to symmetry (details below). The proposed specializations to the three applications are in most cases provably not output-polynomial. This is shown by concrete pathological examples. Still, for the computational examples in these three applications no implementations have been published so far that are nearly as fast as the implementations in TOPCOM (see [25] for the foundations of the version prior to this work) of the methods in this paper.

The paper is organized as follows. Section 2 reviews some important notions and algorithms. Moreover, the notational conventions are fixed and a short general problem statement is given. In Section 3 the theoretical considerations for the new lex-min checks are proven. Section 4 is devoted to an algorithmic analysis of the variants of the top-level algorithm and the lex-min checks valid for arbitrary applications. The discussion of three special applications starts in Section 5 with some common preliminaries on point and vector configurations. Section 6 describes the computational environment used for the numerical experiments. Then, Sections 7 through 9 present the new results that are relevant for each application individually. Finally, Section 10 contains a summary and some conclusions.

2 Preliminaries

In this section, some basic notions and notation are introduced. Moreover, some known algorithms are formulated in terms of the *reverse-search* framework in order to make it easier to compare the known with the novel.

Let $[n]$ denote the set of integers $\{1,2,\dots,n\}$ with the convention $[0]=\emptyset$. For $k\in\mathbb{Z}$ the set of all k-element subsets of $[n]$ is written as $\binom{[n]}{k}$. The power set of $[n]$ is denoted by $2^{[n]}$.

The elements of $2^{[n]}$ and, thus, also of $\binom{[n]}{k}$ are totally ordered by the *subset-lexicographic order* given by $\emptyset <_{\text{lex}} S$ for all $S \neq \emptyset$ and $S <_{\text{lex}} R$ if and only if either $\min S < \min R$ or $\min S = \min R$ and $S \setminus \{\min S\} <_{\text{lex}} R \setminus \{\min R\}$.

For subsets S and S' of $[n]$ the *lex-inclusion partial order* is defined by $S \subseteq_{\text{lex}} S'$ if $S \subseteq S'$ and $s < s'$ for all $s \in S$ and $s' \in S' \setminus S$. In that case S is a *lex-subset* of S' or S' *lex-contains* S. Note that the Hasse-diagram of this partial order is a tree rooted at the empty set.

For an undirected graph $G=(V,E)$ and a node $v \in V$, the *neighborhood* $N(v)$ *of* v *in* G is the set of nodes connected to v by an edge in E. Its cardinality $|N(v)|$ is the *degree* $d(v)$ of $v \in V$, and the maximum of all degrees over all nodes is denoted by $d^{\max}(G)$ (or $d^{\max}$ if G is clear from the context).

The symmetric group on n elements is considered as the set of bijections from $[n]$ to itself.

It is denoted by $\mathfrak{S}_n$, and subgroups of it are usually denoted by $\mathfrak{G}$. For a subgroup $\mathfrak{G}$ of $\mathfrak{S}_n$ and a finite set Ω, a map $\phi : \mathfrak{G} \times \Omega \to \Omega$ is a *(left) group action* of $\mathfrak{G}$ on Ω if $\phi(\pi \cdot \sigma, \omega) = \phi\big(\pi, \phi(\sigma, \omega)\big)$ for all $\pi, \sigma \in \mathfrak{G}$ and all $\omega \in \Omega$. Most of the times, certain clearly induced group actions ϕ are denoted by $\pi(\omega) := \phi(\pi, \omega)$ for $\pi \in \mathfrak{G}$ and $\omega \in \Omega$, leading to $(\pi \cdot \sigma)(\omega) = \pi\big(\sigma(\omega)\big)$. Given such a group action, the *stabilizer subgroup* of $\omega \in \Omega$ in $\mathfrak{G}$ is $\mathfrak{G}_\omega := \{\pi \in \mathfrak{G} : \pi(\omega) = \omega\}$. For $\Omega' \subseteq \Omega$, the *point-wise stabilizer of* Ω' is $\mathfrak{G}_{\Omega'} := \{\pi \in \mathfrak{G} : \pi(\omega) = \omega \text{ for all } \omega \in \Omega'\}$. This must be distinguished from the *set-wise stabilizer of* Ω', denoted by $\mathfrak{G}_{\{\Omega'\}}$, which is defined as $\mathfrak{G}_{\{\Omega'\}} := \{\pi \in \mathfrak{G} : \pi(\omega) \in \Omega' \text{ for all } \omega \in \Omega'\}$. The $\mathfrak{G}$*-orbit* of ω is $\mathfrak{G}(\omega) := \{\pi(\omega) : \pi \in \mathfrak{G}\}$. The set of all $\mathfrak{G}$-orbits is $S/\mathfrak{G} := \{\mathfrak{G}(\omega) : \omega \in \Omega\}$.

For a permutation $\pi \in \mathfrak{S}_n$ and a graph G with node set $V = [n]$ and edge set E let $\pi\big(\{v, w\}\big) = \big\{\pi(v), \pi(w)\big\}$ denote the induced action of π on edges, for all edges $\{v, w\} \in E$. The definition $\pi(E) := \{\pi(e) : e \in E\}$ induces a new graph $\pi(G) = \big(\pi([n]) = [n], \pi(E)\big)$ on the same node set. This defines an action of $\mathfrak{S}_n$ on the set of all graphs on the node set $[n]$. The automorphism group $\mathrm{Aut}(G)$ of a graph $G = ([n], E)$ is the set of all $\pi \in \mathfrak{S}_n$ with $\pi(G) = G$. The elements of $\mathrm{Aut}(G)$ are the *symmetries of* G.

Similarly, for $\pi \in \mathfrak{S}_n$ the induced action of π on any k-element subset $S = \{s_1, \dots, s_k\} \in \binom{[n]}{k}$ is denoted by $\pi(S) = \big\{\pi(s_1), \dots, \pi(s_k)\big\} \in \binom{[n]}{k}$. For a subset $\mathscr{S} = \{S_1, \dots, S_m\}$ of $2^{[n]}$ the induced action of π on $\mathscr{S}$ is denoted by $\pi(\mathscr{S}) = \big\{\pi(S_1), \dots, \pi(S_m)\big\}$. The automorphism group $\mathrm{Aut}(\mathscr{S})$ is the set of all $\pi \in \mathfrak{S}_n$ with $\pi(\mathscr{S}) = \mathscr{S}$.

The analogous concept can be used for sets of subsets of $2^{[n]}$. For example, in one application there is the induced action of a permutation $\pi \in \mathfrak{S}_n$ on a triangulation $\mathscr{T}$ of a point configuration with n labeled points given by the label sets of its maximal simplices. That action is denoted by $\pi(\mathscr{T})$ as well.

Here and in the following, the size of the actual output, generated in the function **output** in all the algorithms below, is significant for the assessment of whether or not an algorithm can be called output-polynomial. If the output is just printing a single symbol for each found object, then the output is equivalent to a unary encoding of the number of found objects, and an output-polynomial algorithm can take time polynomial in the input size and the number of found objects. This problem setup is called the *enumeration problem* in this paper; if the output is skipped altogether, then the remaining output is only the number of objects to be counted, and an output-polynomial algorithm must be polynomial in the input size and the logarithm of the number of found objects. This problem setup is called the *counting problem* in this paper. Finally, if for each enumerated object the object has to be listed, then the ouput is a list of encodings of all enumerated objects. This problem setup is called the *listing problem* in this paper. If the output size of a single object is exponential in the input size, then an algorithm that is polynomial in the input and output sizes for the listing problem can be exponential in the input and output sizes for the enumeration problem. For example, the output of a single triangulation can be exponential in the input size and even in the number of triangulations (see Section 9.2 Theorem 13 (iii)). A similar distinction will be necessary concerning the encoding of the input. If the symmetries are given as an explicit, complete set of permutations in tupel notation, then a polynomial-time algorithm can take time polynomial in the order and the degree of the symmetry group. If the symmetries are given in terms of generators in cycle-notation, then a polynomial-time algorithm needs to be polynomial in the actual input-size of the generators, which can be substantially more restrictive.

In the remainder of this section, the general-purpose algorithm *Symmetric Lexicographic Subset Reverse Search* (`SymLexSubsetRS`) is developed in the language of the reverse-search paradigm from [2]. Starting at the basic Reverse Search, the extensions to orbits and the specializations to subsets are introduced one-by-one. The algorithms in this section are not new, but a summary of all variants in a unified notational environment is helpful to understand the extensions.

As a simplification, a recursive form is used for presentation, which usually increases the memory consumption of the algorithms. However, the more memory-efficient original Reverse-Search framework (with non-recursive backtracking by pivoting) can be applied to all presented algorithms. In this paper, recursive implementations have been used throughout, since they were faster in the presented applications.

First, the original Reverse Search is formulated [2]. Reverse Search (`RS`) is an algorithm to enumerate the nodes of a graph $G = (V, E)$ with known maximal degree $d^{\max}$. The graph is implicitly given by an adjacent-nodes function $\mathrm{Adj} : V \times [d^{\max}] \to V \cup \{\mathsf{NULL}\}$ that returns for each node $v \in V$ and each integer $i \in [d^{\max}]$ the ith neighbor of v if $i \le d(v)$ and NULL if $d(v) < i \le d^{\max}$. The enumeration is structured by a non-degenerate cost function $\phi : V \to \mathbb{R}$ and a pivot-function $p : V \to N(V) \cup \{\mathsf{NULL}\}$, which is a rule to specify a ϕ-improving neighbor in G in case there is one. A recursive representation of Reverse Search in pseudo-code is shown in Algorithm 1.

The run-time complexity of `RS` is $O(d^{\max}\tau(\mathrm{Adj})|V| + \tau(p)|E|)$, where $\tau(f)$ denotes the maximal time to compute a function value of a function f. Since $2|E| \le d^{\max}|V|$, this run-time complexity is in $O(d^{\max}(\tau(\mathrm{Adj}) + \tau(p))|V|)$. This is particularly interesting when $\tau(\mathrm{Adj})$ and $\tau(p)$ do not depend on $|V|$. In that case, `RS` is linear in the output size for the enumeration problem [2].

If one has a group $\mathfrak{G}$ acting on G one is usually only interested in $|V|$ up to symmetry. In other words, the number of $\mathfrak{G}$-orbits $|V/\mathfrak{G}|$ of V shall be determined. How the reverse search principle can be adapted to this setting, was first presented in [13]. The idea is to extend the adjacent-nodes function Adj to $\mathfrak{G}$-orbits in the obvious way and to modify the pivot-function to consist of two steps: In the first step, the ϕ-minimal element, the *canonical element*, in the current orbit is taken; in the second step, the ϕ-reducing pivot-function on G is followed. Thus, another orbit is reached in case the minimal element of the current orbit was no global minimum already. The downside is that all canonical elements in the neighborhood of a node must be stored intermediately to avoid duplicate counting of orbits. This undermines the memory efficiency of the reverse search principle. Algorithm 2 shows a detailed pseudo-code representation.

A specialized form of Reverse Search arises when a set of “feasible” subsets of a finite set shall be enumerated by adding elements, checking feasibility one-by-one, and backtracking. This automatically will touch many subsets of feasible sets. Thus, one can restrict to the enumerations of downsets, i.e., sets of subsets, where each subset of a feasible set is feasible itself. Algorithm 3 shows the specialization of Reverse Search, which is a folklore

```
Algorithm: RS(Adj, $\phi$, $p$, $v$)
Input: a connected graph $G=(V,E)$ with maximal degree $d^{\max}$, implicitly given by an
    adjacent-nodes function $\mathrm{Adj}: V \times [d^{\max}] \to V \cup \{\mathsf{NULL}\}$, a cost function $\phi: V \to \mathbb{R}$
    with $\phi(v) \neq \phi(w)$ for all $v \neq w$ in $V$ and $\phi(v^*) = \min_{v\in V} \phi(v)$, a pivot function
    $p: V \to N(V) \cup \{\mathsf{NULL}\}$ with $\phi\big(p(v)\big) < \phi(v)$ for all $v \in V \setminus \{v^*\}$ and $p(v^*) = \mathsf{NULL}$, a
    seed node $v$ in $V$
Output: the number of nodes in $V_{\geq v} := \{w \in V : \phi(w) \geq \phi(v)\}$
/* build a depth-first-search tree with root node $v$:                        */
output $v$ and $c \leftarrow 1$ ;                                          /* count $v$ */
for $j = 1, \dots, d^{\max}$ do                          /* iterate over neighbors of $v$ */
    $w \leftarrow \mathrm{Adj}(v, j)$ ;                                   /* get $j$th neighbor */
    if $w = \mathsf{NULL}$ then                            /* if past the last neighbor */
        break ;                                                  /* exit the loop */
    if $v = p(w)$ then                                         /* if $w$ pivots to $v$ */
        $c \leftarrow c +$ RS(Adj, $\phi$, $p$, $w$) ;                          /* recurse */
return c;
```

Algorithm 1: The generic reverse search algorithm (cf. [2])

observation.[1] A notable further specialization can be seen in Algorithm 4: by using the natural order for extending subsets by a new element one can guarantee that the subsets are found in lexicographic order.

If downsets shall be enumerated up to symmetry, the lexicographic order on subsets simplifies affairs substantially. If the canonical representative is defined to be the lex-min element in each orbit, then there is no need for computing and comparing canonical representatives for all found elements anymore. The only thing needed is to check whether or not the found element can be lex-decreased *at all* by the action of an element in $\mathfrak{G}$. Since Lemma 2 guarantees that a canonical subset can never lex-contain a non-canonical subset, the enumeration can simply ignore any non-canonical element and backtrack. Therefore, in the following, a subset S is called *canonical* if it is lex-min in its orbit. The resulting algorithm (Algorithm 5) is essentially identical to the algorithm proposed in [24].

With this, the processing of a node in the enumeration of subsets up to symmetry is reduced to checking whether a subset is lex-min in its orbit and to checking, whether a subset is feasible. Thus, the collection of canonicals in the neighborhood of a subset need not be stored.

In [24], a *generator-based recursive method* is proposed to check whether a subset is lex-min in its orbit. The applications in that paper indicate that the methods aim at groups of very large order. For groups of medium-sized order (like in the millions) the, also generator-based, *switch-table method* (originally for vectors, which is more general) is presented in [14].

[1]Strictly speaking, for the algorithms in this paper it is enough for $\mathscr{D}$ to be a *left-downset*, i.e., closed under taking lex-subsets. General downsets are taken in order to keep the problem statements independent of the algorithms and additional structures used for their solutions.

```
Algorithm: SymRS(Adj, $\phi$, $p$, $\mathfrak{G}$, $\check{v}$)
Input: a connected graph $G=(V,E)$ with maximal degree $d^{\max}$, implicitly given by an
       adjacent-nodes function $\mathrm{Adj} : V \times [d^{\max}] \to V \cup \{\mathsf{NULL}\}$, a cost function $\phi : V \to \mathbb{R}$
       with $\phi(v) \neq \phi(w)$ for all $v \neq w$ in $V$ and $\phi(v^*) = \min_{v\in V} \phi(v)$ , a pivot-function
       $p : V \to N(V) \cup \{\mathsf{NULL}\}$ with $\phi\big(p(v)\big) < \phi(v)$ for all $v \in V \setminus \{v^*\}$ and $p(v^*) = \mathsf{NULL}$, a
       subgroup $\mathfrak{G}$ of $\mathrm{Aut}(G)$ and a canonical-representative function Canonical($v$) =: $\check{v}$
       with $\phi(\check{v}) < \phi(w)$ for all $w \in \mathfrak{G}(v)$, a seed node $\check{v}$ in $V$ that is the canonical
       representative of $\mathfrak{G}(\check{v})$
Output: the number of canonical $\check{w}$ with $\phi(\check{w}) \geq \phi(\check{v})$
/* build a depth-first-search tree with root node $\check{v}$:                     */
output $\check{v}$ and $c \leftarrow 1$ ;                                        /* count $\check{v}$ */
$\check{W} \leftarrow \{\check{v}\}$ ;                          /* collects already processed canonicals */
for $j = 1, \ldots, d^{\max}$ do                                   /* iterate over neighbors of $\check{v}$ */
    $w \leftarrow \mathrm{Adj}(\check{v}, j)$ ;                                        /* get $j$th neighbor */
    if $w = \mathsf{NULL}$ then                                      /* if past the last neighbor */
        break ;                                                            /* exit the loop */
    $\check{w} \leftarrow$ Canonical($w$) ;                          /* compute canonical representative */
    if $\check{w} \notin \check{W}$ then                                                /* if $\check{w}$ is new */
        $u \leftarrow p(\check{w})$ ;                                                  /* compute pivot */
        $\check{u} \leftarrow$ Canonical($u$) ;                      /* compute canonical representative */
        if $\check{u} = \check{v}$ then                                   /* if $\mathfrak{G}(w)$ pivots to $\mathfrak{G}(\check{v})$ */
            $\check{W} \leftarrow \check{W} \cup \{\check{w}\}$ ;             /* add $\check{w}$ to set of processed canonicals */
            $c \leftarrow c +$ RS(Adj, $\phi$, $p$, $\mathfrak{G}$, $\check{w}$) ;                                /* recurse */
return c;
```

Algorithm 2: The application of reverse-search to the graph of all orbits in $V/\mathfrak{G}$

```
Algorithm: SubsetRS($n$, $\mathscr{D}$, $S$)
Input: $n \in \mathbb{N}$, a downset $\mathscr{D}$ of $2^{[n]}$, a seed set $S \in \mathscr{D}$
Output: the number of elements $S'$ in $\mathscr{D}$ with $S \subseteq_{\mathrm{lex}} S'$
/* build a depth-first-search tree with root node $S$:                           */
output $S$ and $c \leftarrow 1$ ;                                              /* count $S$ */
for $j \in [n] \setminus S$ with $j > \max(S)$ do                /* for extensions with new max */
    $S' \leftarrow S \cup \{j\}$ ;                                             /* build new set */
    if IsInDownset($S'$, $\mathscr{D}$) then                       /* if new set belongs to $\mathscr{D}$ */
        $c \leftarrow c +$ LexSubsetRS($n$, $\mathscr{D}$, $S'$) ;                               /* recurse */
return c;
```

Algorithm 3: The standard application of reverse-search to the Hasse-diagram of a downset $\mathscr{D}$ of subsets of an n-element set that exploits the straight-forward set-valued inverse of the pivot function

```
Algorithm: LexSubsetRS($n, \mathscr{D}, S$)
Input: $n \in \mathbb{N}$, a downset $\mathscr{D}$ of $2^{[n]}$, a seed set $S \in \mathscr{D}$
Output: the number of elements $S'$ in $\mathscr{D}$ with $S \subseteq_{\text{lex}} S'$
/* build a depth-first-search tree with root node $S$:                    */
output $S$ and $c \leftarrow 1$ ;                                       /* count $S$ */
for $j = \max(S)+1, \ldots, n$ do              /* ordered traversal of new elements */
    $S' \leftarrow S \cup \{j\}$ ;                                      /* build new set */
    if IsInDownset($S', \mathscr{D}$) then                /* if new set belongs to $\mathscr{D}$ */
        $c \leftarrow c +$ LexSubsetRS($n, \mathscr{D}, S'$) ;                  /* recurse */
return c;
```

Algorithm 4: A specialized implementation of SubsetRS($n, \mathscr{D}, S$) traversing the pivot-inverse according to the lexicographic order of subsets of $[n]$

```
Algorithm: SymLexSubsetRS($n, \mathscr{D}, \mathfrak{G}, S$)
Input: $n \in \mathbb{N}$, a downset $\mathscr{D}$ of $2^{[n]}$, a subgroup $\mathfrak{G}$ of the automorphism group of $\mathscr{D}$, a seed set
       $S \in \mathscr{D}$ with $S = \text{lex-min}\,\mathfrak{G}(S)$
Output: the number of canonical $S'$ in $\mathscr{D}$ with $S \subseteq_{\text{lex}} S'$
/* build a depth-first-search tree with root node $S$:                    */
output $S$ and $c \leftarrow 1$ ;                                       /* count $S$ */
for $i = \max(S)+1, \ldots, n$ do      /* ordered traversal of new maximal elements */
    $S' \leftarrow S \cup \{i\}$ ;                      /* add a new element on the right */
    answer $\leftarrow$ IsLexMin($S', \mathfrak{G}$) ;                        /* lex-min check */
    if answer = FALSE then         /* if new set is not lex-min in its orbit */
        continue ;                                        /* next loop element */
    answer $\leftarrow$ IsInDownset($S', \mathscr{D}$) ;                   /* membership check */
    if answer = FALSE then                           /* if new set is not in $\mathscr{D}$ */
        continue ;                                        /* next loop element */
    $c \leftarrow c +$ SymLexSubsetRS($n, \mathscr{D}, \mathfrak{G}, S'$) ;                 /* recurse */
return c;
```

Algorithm 5: A specialized implementation of the combination of LexSubsetRS and SymRS for orbits of subsets in a downset $\mathscr{D}$ of $2^{[n]}$ (cf. [24])

A new combination of the methods in [24] and [14] (for subsets) turned out to be the fastest method for the enumeration of cocircuits and circuits in the test cases of this paper. For even smaller group orders (like in the thousands) the overhead of all the generator-based methods may not pay off. Hence, a new taylor-made *element-based* method is developed in Section 3 exactly for those cases. This turned out to be the fastest method for the enumeration of triangulations.

Switch tables [14] are non-standard and are therefore introduced in the following. Given a subgroup $\mathfrak{G}$ of $\mathfrak{S}_n$, a *switch table* is a function $\operatorname{st}(\cdot,\cdot) : [n] \times [n] \to \mathfrak{G}$ with either $\operatorname{st}(i, j) = \pi$ so that $\pi(j) = i$ for some $j > i$ and $\pi(k) = k$ for all $k < i$, if such a $\pi \in \mathfrak{G}$ exists, or $\operatorname{st}(i, j) = \mathrm{id}$ otherwise. A switch table need not be unique. The entries $\operatorname{st}(i, j)$ are called *switches*. An entry of a switch table is *trivial* if it is the identity. A row of a switch table is *effective* if it contains at least one non-trivial switch. The set effRowSet of all row indices i so that $\operatorname{st}(i,\cdot)$ is effective is called the *effective row set* of $\operatorname{st}(\cdot,\cdot)$. The (non-empty) set of column indices of non-trivial switches in an effective row $i \in \mathrm{effRowSet}$ is the *effective column set* of i, denoted by $\mathrm{effColSet}(i)$. For the trivial group, there are no effective rows, for which we adopt the convention that $\max \mathrm{effRowSet} = -\infty$.

One key property of a switch table is that each element of $\mathfrak{G}$ is an essentially unique (up to the insertion of trivial factors) product of switches consisting of at most one non-trivial switch from each row. Not all switches can contribute to a lex-decreasing switch-product, so that not all products need to be considered. A straight-forward version of the algorithm for subsets is presented in Algorithm 6. Details for general vectors instead of subsets – which can be seen as special, namely characteristic, vectors – can be found in [14]. The algorithm uses global auxiliary data, which is given by a switch table $\mathsf{ST} = \operatorname{st}(\cdot,\cdot)$ for $\mathfrak{G}$.

The membership-test in `IsInDownset` can be difficult. In cases, where the downset to be enumerated up to symmetry has only been implicitly defined as a downset by a set of really interesting subsets together with all its subsets, `IsInDownset` in fact needs to answer the question whether or not a subset is a subset of a really interesting set. This problem arises in all of the applications in this paper and has to be solved for each application seperately. In section 4, variants of `SymLexSubsetRS` will be designed where this membership test is relaxed in order to make faster progress at the cost of accepting deadends in the enumeration.

Problem Statement In this paper, the following questions concerning the algorithm `SymLexSubsetRS` and some variants are studied in detail:

- How can the subroutine `IsLexMin` of `SymLexSubsetRS` be implemented efficiently *in general*?
- How can the subroutines of `SymLexSubsetRS` and its variants be implemented efficiently *for each of the applications*. More specifically:
  - How can one effectively prune subsets of vectors that are not lex-contained in a circuit or cocircuit, respectively?
  - How can one effectively prune subsets of simplices that are not lex-contained in a triangulation of the point configuration?

```
Algorithm: IsLexMin_viaSwitches($i, S, S', \mathfrak{G}, \mathsf{ST}$)
Input: an integer $i \in [1, n]$, a subset $S$ (the original subset), a subset $S'$ (the subset mapped
       by a partial switch product), a subgroup $\mathfrak{G}$ of $\mathfrak{S}_n$, and a switch table ST of $\mathfrak{G}$
Output: TRUE if $S = \text{lex-min}\,\mathfrak{G}_{[i-1]}(S)$ and FALSE otherwise
/* check for empty set:                                                    */
if $S = \emptyset$ then
    return TRUE;
/* beyond the effective row set?                                           */
if $i > \max \text{effRowSet}(\mathsf{ST})$ then
    return $(S' \not<_{\text{lex}} S)$;
/* recursively check a switch product with leading identity:              */
if IsLexMin_viaSwitches($i+1, S, S', \mathfrak{G}, \mathsf{ST}$) = FALSE then
    return FALSE;
/* collect "good" switches lex-decreasing $S'$:                             */
$\mathsf{GS} \leftarrow \{ j \in \text{effColSet}(i) \mid i \notin S', j \in S' \}$;
if $\mathsf{GS} = \emptyset$ then
    /* collect "good" switches not preventing future lex-decrease:        */
    $\mathsf{GS} \leftarrow \{ j \in \text{effColSet}(i) \mid |S' \cap \{i, j\}| \in \{0, 2\} \}$;
for $j \in \mathsf{GS}$ do
    $S'' \leftarrow \mathsf{ST}[i][j](S')$ ;                                /* map subset */
    if $S'' <_{lex} S$ then
        return FALSE ;                    /* decreasing switch product found */
    /* recurse with mapped subset:                                          */
    if IsLexMin_viaSwitches($i+1, S, S'', \mathfrak{G}, \mathsf{ST}$) = FALSE then
        return FALSE;
return TRUE ;                          /* no decreasing switch product found */
```

Algorithm 6: The switch-table method from [14] specialized to subsets

The term *efficiently* here does *not* mean *polynomial-time*. It rather indicates that compared to naive approaches the number of steps is reduced as much as possible. The resulting algorithms are, in general, not (counting/enumerating/listing for cocircuits and circuits as well as counting/enumerating for triangulations) or not known to be (listing for triangulations) output-polynomial.

3 On Lexicographically Minimal Elements in Subset-Orbits

This section provides mathematical structures that help to decide whether or not a subset is lexicographically minimal in its orbit. For some of the upcoming arguments the following characterization of the lexicographic order on k-element subsets of $[n]$ is used.

Lemma 1 (Lexicographic order on subsets)**.** *Let $n \in \mathbb{N}$. Moreover, let S and R be k-element subsets of $[n]$. Then S is lexicographically smaller than R if the minimal element of their symmetric difference is in S. In formulae:*

$$S <_{lex} R \iff \min(S \triangle R) \in S, \tag{1}$$

where $\min \emptyset = \infty$.

The following lemma states that if subsets are built element-by-element with backtracking and only the lexicographically minimal ones in their orbits are followed, then one will reach all subsets that are lexicographically minimal in their orbits. This result was already presented in [24]. Here, some more details for the proof are provided.

Lemma 2 (cf. [24])**.** *Let S be a non-empty subset of $[n]$ and $S^- := S \setminus \{\max S\}$. Then, for all subgroups $\mathfrak{G}$ of $\mathfrak{S}_n$ we have:*

$$S = \text{lex-min}\, \mathfrak{G}(S) \Rightarrow S^- = \text{lex-min}\, \mathfrak{G}(S^-) \tag{2}$$

Proof. Let S be lex-min in its $\mathfrak{G}$-orbit. Assume, for the sake of contradiction, that S^- is not lex-min in its $\mathfrak{G}$-orbit. Then, there is a set R which is lex-smaller than S^- and a permutation $\pi \in \mathfrak{G}$ with $\pi(S^-) = R$. In particular, S^- and R are non-empty and non-identical. That means, $\min(S^- \triangle R) =: r \in R \setminus S^-$. Moreover, $\pi(\max S) \notin R$, by the bijectivity of any permutation. Consider $R^+ := R \cup \{\pi(\max S)\}$.

Since $\max S > \min(S^- \setminus R) > \min(R \setminus S^-) = \min(S^- \triangle R) = r$, we know that $\min(S \setminus R) > r$. Moreover, $R \subset R^+$ implies $\min(S \setminus R^+) \geq \min(S \setminus R) > r$. Since $r \in R \setminus S^-$ and $r < \max S$, it follows that $r \in R \setminus S \subseteq R^+ \setminus S$. Thus, $\min(R^+ \setminus S) \leq r < \min(S \setminus R^+)$, and, hence, $\min(S \triangle R^+) \in R^+$. Thus, by definition, R^+ is lex-smaller than S with $\pi(S) = R^+$: contradiction. □

Next, two methods for the lex-min check are presented: one for "small" symmetry groups (order small compared to degree) based on the new concept of *critical-element tables* and one for "large" symmetry groups (order large compared to degree) based on a modified use of switch tables.

First, the method for small groups is presented. It is based on the new structure of *critical-element tables*. The third application in this paper concerns most often smaller symmetry groups with relatively high degree that can easily be enumerated first. How can one reduce the number of evaluations of actions on subsets in that check? Remember that the orbits' lexicographically-minimal subsets are built element-by-element. That is, for the check of whether or not a given m-subset S is lexicographically minimal in its orbit, one can exploit that its predecessor-subset S^- with $m-1$ elements is already known to be lexicographically minimal in its orbit. The new question is: has the addition of the new maximal element led to the existence of a permutation that lexicographically decreases (*lex-decreases*, for short) the new subset S in its orbit?

The new idea in this paper is to keep the information about *why* the predecessor subset S^- is lexicographically minimal in its orbit. The reason is that for each permutation $\pi \in \mathfrak{G}$ one has

$$\min\bigl(S^- \mathbin{\triangle} \pi(S^-)\bigr) \notin \pi(S^-). \tag{3}$$

These minimal elements of the symmetric differences of subsets and their images are critical for the question at hand, which motivates the following definition:

Definition 1. Let $n \in \mathbb{N}$ and $\mathfrak{G}$ be a subgroup of $\mathfrak{S}_n$. For a given subset S the *critical-element table with respect to* S is defined as follows:

$$\operatorname{critelem}_S \colon \begin{cases} \mathfrak{G} & \to & [n] \cup \{\infty\}, \\ \pi & \mapsto & \min\bigl(S \mathbin{\triangle} \pi(S)\bigr), \end{cases} \tag{4}$$

where $\min \emptyset := \infty$. The function value $\operatorname{critelem}_S(\pi)$ is called the *critical element of* π *with respect to* S.

From the critical-element table, some important of properties of the action of a permutation π on the given subset S can be derived easily:

Lemma 3. *Let $n \in \mathbb{N}$ and $\mathfrak{G}$ be a subgroup of $\mathfrak{S}_n$. Moreover, let S be a non-empty subset of $[n]$. Then:*

(i) The stabilizer of S in $\mathfrak{G}$ is the set of permutations with critical element ∞.

(ii) S is lexicographically minimal in its $\mathfrak{G}$-orbit if and only if $\operatorname{critelem}_S(\pi) \notin \pi(S)$ *for all* $\pi \in \mathfrak{G}$.

(iii) Let $S^- := S \setminus \{\max S\}$. Moreover, assume that S^- is lexicographically minimal in its $\mathfrak{G}$-orbit. Then a permutation $\pi \in \mathfrak{G}$ lex-decreases S if and only if one of the following cases occurs:

I. $\operatorname{critelem}_{S^-}(\pi) = \infty$ *and* $\pi(\max S) < \max S$

II. $\operatorname{critelem}_{S^-}(\pi) \in S^-$ *and* $\pi(\max S) < \operatorname{critelem}_{S^-}(\pi)$ *or*

III. $\operatorname{critelem}_{S^-}(\pi) \in S^-$, $\pi(\max S) = \operatorname{critelem}_{S^-}(\pi)$ *and* $\operatorname{critelem}_S(\pi) \in \pi(S)$ □

Call the application of this the *critical-element method* for checking lexicographic minimality of a subset in its orbit.

The crucial gain of this lemma is the following: given the critical-element table with respect to S^-, one can check lexicographic minimality of S in its orbit without actually computing $\pi(S)$, with the only exception when π maps the new element of S exactly to the critical element of S^-. And this exception roughly happens for a $\frac{1}{n}$-fraction of the permutations, on average.

There are now two ways to implement the critical-element method.

1. iterate over all permutations and apply Lemma 3 to each of them (the *iteration-based critical-element method*; see Algorithm 11 in Section 4 for a possible implementation);
2. first, from certain fixed, preprocessed subsets of permutations, compute the subsets of permutations that
   (a) certainly lexicographically decrease the given subset, and if empty,
   (b) possibly lexicographically decrease the given subset.

   If the subset of certainly lexicographically decreasing permutations is non-empty, the subset is not lexicographically minimal in its orbit; if it is empty, iterate over the possibly lexicographically decreasing permutations and apply Lemma 3 to only those (the *set-based critical-element method*; see Algorithm 12 in Section 4 for a possible implementation).

In the sequel, some details for the set-based method are explained.[2] The following structures of the possibly lexicographically decreasing permutations for a specific subset. The first three structures can be preprocessed prior to the enumeration. The fourth structure has to be updated for each enumeration node.

Definition 2. For $n \in \mathbb{N}$ and a subgroup $\mathfrak{G}$ of $\mathfrak{S}_n$, the *hit-element classification of* $\mathfrak{G}$ is defined as

$$\text{hitclass:} \left\{ \begin{array}{rcl} [n] \times [n] & \to & 2^{\mathfrak{G}}, \\ (i,j) & \mapsto & \{\pi \in \mathfrak{G} : \pi(i) = j\}. \end{array} \right. \tag{5}$$

The *increasing-element classification of* $\mathfrak{G}$ is defined as

$$\text{incclass:} \left\{ \begin{array}{rcl} [n] & \to & 2^{\mathfrak{G}}, \\ i & \mapsto & \{\pi \in \mathfrak{G} : \pi(i) > i\} = \bigcup_{k=i+1}^{n} \text{hitclass}(i,k). \end{array} \right. \tag{6}$$

[2]The approach is motivated by the fact that with a set structure based on dynamic bitstrings it is possible to compute unions, intersections, differences, and symmetric differences of sets fast in practice. Nota bene: From a complexity standpoint, bitstrings only gain something if their length is uniformly bounded. However, in practice the reduction of the runtime by a constant factor by using dynamic bitstrings for set operations is not irrelevant. Moreover, operations on dynamic bitstrings residing consecutively in memory are more cache coherent than data structures that are scattered in main memory.

The *decreasing-element classification of* $\mathfrak{G}$ is defined as

$$\text{decclass}\colon \begin{cases} [n]\times[n] & \longrightarrow \quad 2^{\mathfrak{G}}, \\ (i,j) & \longmapsto \quad \{\pi\in\mathfrak{G} : \pi(i) < j\} = \bigcup_{k=1}^{j-1} \text{hitclass}(i,k). \end{cases} \tag{7}$$

Moreover, for a subset S of $[n]$, the *critical-element classification of* $\mathfrak{G}$ *with respect to* S is defined as

$$\text{critclass}_S\colon \begin{cases} [n]\cup\{\infty\} & \longrightarrow \quad 2^{\mathfrak{G}}, \\ i & \longmapsto \quad \{\pi\in\mathfrak{G} : \text{critelem}_S(\pi) = i\}. \end{cases} \tag{8}$$

The next lemma characterizes lexicographical minimality in an orbit by intersections of certain decreasing-element and critical-element classifications.

Lemma 4. *Let* $n\in\mathbb{N}$ *and* $\mathfrak{G}$ *a subgroup of* $\mathfrak{S}_n$. *Moreover, let* S *be a non-empty subset of* $[n]$, *and let* $S^- = S\setminus\{\max S\}$ *be lexicographically minimal in its* $\mathfrak{G}$*-orbit. Then* S *is not lexicographically minimal in its* $\mathfrak{G}$*-orbit if and only if at least one of the following cases occurs:*

I. *The set* $\text{decclass}(\max S, \max S)\cap\text{critclass}_{S^-}(\infty)$ *is non-empty.*

II. *There is an element* $i\in S^-$ *such that the set* $\text{decclass}(\max S, i)\cap\text{critclass}_{S^-}(i)$ *is non-empty.*

III. *There is an* $i\in S^-$ *and a permutation* π *in the set* $\text{hitclass}(\max S, i)\cap\text{critclass}_{S^-}(i)$ *such that* $\text{critelem}_S(\pi)\in\pi(S)$. □

The set-based method has the advantage that the checks that potentially lead to an immediate answer can be done prior to the complicated cases, whereas in the iteration method it depends on the order of the permutations when the complicated cases have to be handled. And whether or not an order of the permutations is advantageous heavily depends on the subset at hand. The disadvantage of the set-based method is that for large n especially the decreasing-element classification can grow large. Intermediate preprocessing structures are possible trading speed for size. Corresponding details, however, are not discussed any further in this work, since this is rather a topic of software-engineering.

For the applications presented in this paper the following observations can be made: For enumerating triangulations (one prominent example $\mathbf{\Delta}_6\times\mathbf{\Delta}_2$ has $|\mathfrak{G}| = 30{,}240$ and $n = 35{,}721$), the iteration method is mostly faster, whereas for the large cases in the enumeration of circuits and cocircuits (our largest example, the cocircuits of the 9-cube $\mathbf{C}_9$, has $|\mathfrak{G}| = 185{,}794{,}560$ and $n = 512$) the set-based method is significantly quicker.

However, for the latter application with a large group order compared to the degree the second new method based on switch tables is even faster than the set-based critical-element method. This method is explained in the following. It works by combining switch tables with the recursive algorithm in [24]. The recursive algorithm in [24] answers a slightly more general question: for two given subsets S_{img} and S_{org} with the same number of elements, does there exist a permutation $\pi\in\mathfrak{G}$ with $\pi(S_{\text{img}}) <_{\text{lex}} S_{\text{org}}$? The answer is certainly "no" if the subsets or their complements are empty. The answer is "yes" if the some element of S_{img} can be mapped to something smaller than the minimal element $s_{\min}$ of S_{org}. The answer is “no” if all elements of S_{img} are mapped by $\mathfrak{G}$ to something strictly larger than $s_{\min}$. And if some $\pi \in \mathfrak{G}$ maps some element $k \in S_{\text{img}}$ exactly to $s_{\min}$, the answer is given by answering the same question recursively for $\pi(S_{\text{img}}) \setminus \{s_{\min}\}$, $S_{\text{org}} \setminus \{s_{\min}\}$, and the group $\mathfrak{G}_{[s_{\min}]}$. Switch tables allow to run this recursive algorithm without the administration of new stabilizer groups. The new adaption is based on the following.

Lemma 5. *Consider a switch table* $\operatorname{st}(\cdot,\cdot)$ *for a subgroup* $\mathfrak{G}$ *of* $\mathfrak{S}_n$*. Let* $1 \le i \le n$*, and let* $\mathfrak{G}_{[i-1]}$ *be the point-wise stabilizer of* $[i-1]$ *in* $\mathfrak{G}$*. Moreover, let* S_{org} *and* S_{img} *be subsets of* $[n] \setminus [i-1]$ *with identical cardinality. Then:*

(i) If $S_{\text{org}} = \emptyset$ *or* $S_{\text{org}} = [n] \setminus [i-1]$*, then there is no permutation* $\pi \in \mathfrak{G}_{[i-1]}$ *with* $\pi(S_{\text{img}}) <_{lex} S_{\text{org}}$*.*

(ii) If $\mathfrak{G}$ *is the trivial group or* $i > \max(\text{effRowSet})$*, then there is a permutation* $\pi \in \mathfrak{G}_{[i-1]}$ *with* $\pi(S_{\text{img}}) <_{lex} S_{\text{org}}$ *if and only if* $S_{\text{img}} <_{lex} S_{\text{org}}$*.*

(iii) If $i \in S_{\text{org}}$*, then:*

(a) If $i \in S_{\text{img}}$ *and there is a permutation* $\pi' \in \mathfrak{G}_{[i]}$ *with* $\pi'\big(S_{\text{img}} \setminus \{i\}\big) <_{lex} S_{\text{org}} \setminus \{i\}$*, or*

(b) if there is a non-trivial switch $\operatorname{st}(i, j_i)$ *with* $j_i \in S_{\text{img}}$ *and a permutation* $\pi' \in \mathfrak{G}_{[i]}$ *with* $\pi'\big(\operatorname{st}(i, j_i)(S_{\text{img}} \setminus \{j_i\})\big) <_{lex} S_{\text{org}} \setminus \{i\}$*,*

then there is a permutation $\pi \in \mathfrak{G}_{[i-1]}$ *with* $\pi(S_{\text{img}}) <_{lex} S_{\text{org}}$*. If none of these two cases occurs, then there is no permutation* $\pi \in \mathfrak{G}_{[i-1]}$ *with* $\pi(S_{\text{img}}) <_{lex} S_{\text{org}}$*.*

(iv) If $i \notin S_{\text{org}}$*, then:*

(a) If $i \in S_{\text{img}}$*, or*

(b) $\text{effColSet}(i) \cap S_{\text{img}} \neq \emptyset$*, or*

(c) there is a permutation $\pi' \in \mathfrak{G}_{[i]}$ *with* $\pi'\big(S_{\text{img}}\big) <_{lex} S_{\text{org}}$*, or*

(d) there is a switch $\operatorname{st}(i, j_i)$ *with* $j_i \notin S_{\text{img}}$ *and a permutation* $\pi' \in \mathfrak{G}_{[i]}$ *with* $\pi'\big(\operatorname{st}(i, j_i)(S_{\text{img}})\big) <_{lex} S_{\text{org}}$*,*

then there is a permutation $\pi \in \mathfrak{G}_{[i-1]}$ *with* $\pi(S_{\text{img}}) <_{lex} S_{\text{org}}$*. If none of these four cases occurs, then there is no permutation* $\pi \in \mathfrak{G}_{[i-1]}$ *with* $\pi(S_{\text{img}}) <_{lex} S_{\text{org}}$*.*

In particular, for $i = 1$ *and* $S_{\text{img}} = S_{\text{org}}$ *these conditions characterize whether there is a permutation* $\pi \in \mathfrak{G}$ *with* $\pi(S_{\text{org}}) <_{lex} S_{\text{org}}$*.*

Proof. There is a permutation $\pi \in \mathfrak{G}_{[i-1]}$ with $\pi(S_{\text{img}}) <_{\text{lex}} S_{\text{org}}$ if and only if there is a switch product $\operatorname{st}(n, j_n) \cdots \operatorname{st}(i, j_i)$ with $\big(\operatorname{st}(n, j_n) \cdots \operatorname{st}(i, j_i)\big)(S_{\text{img}}) <_{\text{lex}} S_{\text{org}}$, by [14].

Case (i) and (ii) are straight-forward.

Case (iii) (a) is formally required to account for the identity switch in row i; the argument is then the same as for the next case. In case (iii) (b), consider the case when there is a non-trivial switch $\operatorname{st}(i, j_i)$ with $j_i \in S_{\text{img}}$ and a permutation $\pi' \in \mathfrak{G}_{[i]}$ with $\pi'\big(\operatorname{st}(i, j_i)(S_{\text{img}} \setminus \{j_i\})\big) <_{\text{lex}} S_{\text{org}} \setminus \{i\}$. Consider $\pi := \pi' \cdot \operatorname{st}(i, j_i)$. Then, $\pi(j_i) = i = \min(S_{\text{org}})$, since $\pi' \in \mathfrak{G}_{[i]}$ stabilizes i. Because

$j_i \in S_{\mathrm{img}}$, it follows that $\pi(S_{\mathrm{img}}) <_{\mathrm{lex}} S_{\mathrm{org}}$ if and only if $\pi'(S_{\mathrm{img}} \setminus \{j_i\}) <_{\mathrm{lex}} S_{\mathrm{org}} \setminus \{i\}$, which is the case by the choice of π'. If there is no non-trivial switch $\mathrm{st}(i, j_i)$ with $j_i \in S_{\mathrm{img}}$, then all switch products $\mathrm{st}(n, j_n)\cdots\mathrm{st}(i, j_i)$ map all elements of S_{img} to elements strictly larger than $i = \min(S_{\mathrm{org}})$, and S_{org} is therefore strictly lex-smaller than any subset in the $\mathfrak{G}_{[i-1]}$-orbit of S_{img}. If, moreover, for all switches $\mathrm{st}(i, j_i)$ with $j_i \in S_{\mathrm{img}}$ the $\mathfrak{G}_{[i]}$-orbit of $\mathrm{st}(i, j_i)(S_{\mathrm{img}} \setminus \{j_i\})$ contains no lex-smaller element than $S_{\mathrm{org}} \setminus \{i\}$, then no switch product $\mathrm{st}(n, j_n)\cdots\mathrm{st}(i, j_i)$ with $j_i \in S_{\mathrm{img}}$ can lex-decrease S_{img} below S_{org}.

In case (iv)(a) the minimal element of S_{org} is strictly larger than the minimal element i of S_{img}, thus $S_{\mathrm{img}} <_{\mathrm{lex}} S_{\mathrm{org}}$ so that $\pi = \mathrm{id}$ will do the job. In case (iv)(b) there is a switch $\mathrm{st}(i, j_i)$ mapping an element of S_{img} to $i < \min(S_{\mathrm{org}})$, so $\pi = \mathrm{st}(i, j_i) = \mathrm{id}\cdots\mathrm{id}\cdot\mathrm{st}(i, j_i)$ maps S_{img} to a lex-smaller subset than S_{org}. The cases (iv)(c) and (d) are essentially analogous to case (iii)(a) and (b).

The final assertion follows from backward induction on i, rooted at the case $i = n$ with the trivial group $\mathfrak{G}_{[n]}$. □

Call the application of Lemma 5 the *modified switch-table method* for checking lexicographic minimality of a subset in its orbit; see Algorithm 13 in Section 4 for a possible implementation.

4 Algorithms

In this section, the following are introduced and analyzed:

- Variants of `SymLexSubsetRS` (Algorithm 5 in Section 2) supporting global and local auxiliary data, counting only maximal elements, minimal non-elements, and feasible elements, resp., and utilizing *semi-deciding* algorithms to prune the enumeration, i.e., roughly speaking, algorithms that correctly answer "TRUE" (i.e., "prune") only if a subset is not a subset of an interesting set and "FALSE" (i.e., "continue") if a subset might or might not be a subset of an interesting set.

- Three new ways to implement `IsLexMin` in those variants. Recall that this subroutine checks whether or not a subset is lex-min in its orbit.

4.1 Variants of Symmetric Lexicographic Subset Reverse Search

First, the algorithm `SymLexSubsetRS_withData` is presented, which is a slightly modified form of `SymLexSubsetRS`. The reason is that for most non-trivial problems auxiliary data has to be computed and to be kept in memory. The are two principal ways auxiliary data can be made available. Data that depends on the current subset in the reverse-search tree is stored in a node of the reverse-search tree together with the subset. Data independent of the subset can be stored globally, e.g., together with the problem data.

To keep track of this in the following, specify by $N = (S_N, \mathbf{L}_N)$ a node in the reverse-search tree consisting of a subset $S_N \in 2^{[n]}$ and a subset-specific collection of data $\mathbf{L}_N$, which formally is just a tupel of mathematical objects. The global collection of data is denoted by $\mathbf{G}$.

```
Algorithm: SymLexSubsetRS_withData($n$, $\mathscr{D}$, $\mathfrak{G}$, $\mathbf{G}$, $N$)
Input: $n \in \mathbb{N}$, a downset $\mathscr{D}$ of $2^{[n]}$, a subgroup $\mathfrak{G}$ of the automorphism group of $\mathscr{D}$, global
        data $\mathbf{G}$, a node $N = (S, \mathbf{L})$ with canonical $S \in \mathscr{D}$ and $\mathbf{L}$ local data
Output: the number of canonical $S'$ in $\mathscr{D}$ with $S \subseteq_{\text{lex}}$ lex-min $S'$
output $S$ and $c \leftarrow 1$ ;                                          /* count $S$ */
for $i = \max(S)+1, \ldots, n$ do        /* ordered traversal of new maximal elements */
    $S' \leftarrow S \cup \{i\}$ ;                             /* add a new element on the right */
    default initialize $\mathbf{L}'$ ;                       /* prepare a local data structure */
    $(\text{answer}, \mathbf{L}') \leftarrow$ IsLexMin($S', \mathfrak{G}, \mathbf{G}, \mathbf{L}, \mathbf{L}'$) ;                 /* lex-min check */
    if answer = FALSE then          /* if new set is not lex-min in its orbit */
        continue ;                                        /* next loop element */
    $(\text{answer}, \mathbf{L}') \leftarrow$ IsInDownset($S', \mathscr{D}, \mathbf{G}, \mathbf{L}, \mathbf{L}'$) ;           /* membership check */
    if answer = FALSE then                               /* if new set is not in $\mathscr{D}$ */
        continue ;                                        /* next loop element */
    $N' \leftarrow (S', \mathbf{L}')$ ;                                         /* build new node */
    $c \leftarrow c +$ SymLexSubsetRS_withData($n$, $\mathscr{D}$, $\mathfrak{G}$, $\mathbf{G}$, $N'$) ;           /* recurse */
return c;
```

Algorithm 7: A variant of symmetric lexicographic subset reverse search with explicit use of global and local auxiliary data

Any implementation of SymLexSubsetRS_withData must specify the exact structure of **L** and **G**. It is desirable that the local data for a new node can be computed during one or both of the rather expensive subroutines IsLexMin and IsInDownset. A possible such layout can be seen in Algorithm 7. The run-time complexity depends on the run-time complexities of the subroutines and can be derived readily from the underlying reverse search structure – it is in $O\big(n(\tau(\texttt{IsLexMin}) + \tau(\texttt{IsInDownset}))|\mathscr{D}/\mathfrak{G}|\big)$, assuming initializing local data and building a new node are dominated by the other subroutines.

At times, not the cardinality of the downset $\mathscr{D}$ up to symmetry is of interest but only the number of its maximal elements. This is, e.g., the case for the enumeration of cocircuits up to symmetry. In such situations, the downset $\mathscr{D}$ is only implicitly defined as the downset of all subsets of the subsets one is really interested in. This models the process building interesting objects from scratch element-by-element. In the ordered reverse-search tree of subsets considered in the algorithms so far, the added elements are always larger than the current maximal element. Therefore, the following notions are defined:

Definition 3. For a subset $S \in \mathscr{D}$ an *expansion of* S is an element $i \in [n] \setminus S$ so that $S \cup \{i\} \in \mathscr{D}$. A *right-expansion* is an expansion i with $i > \max S$. The subset S is *maximal* if there is no expansion for it, and it is *right-maximal* if there is no right-expansion for it. The subset S is *right-completable* if there is a set S' that is maximal in $\mathscr{D}$ so that S' lex-contains S.

By definition, each right-expansion of a set lex-contains that set. Algorithm 8 shows the adapted algorithm enumerating the maximal elements in a downset. It might appear that the leaves of the enumeration tree automatically correspond to maximal subsets in $\mathscr{D}$. This, however, is unfortunately not the case – the leaves are, by construction, only right-maximal: Consider an arbitrary maximal non-empty subset S in $\mathscr{D}$. Then, e.g., the enumeration branch starting with the second smallest element in S will have $S \setminus \min S$ as one of its leaves, which is obviously not maximal in D. Thus, a maximality check has to be performed on the leaves, and non-maximal leaves are simply ignored for the count. Sometimes the reverse-search tree can be pruned by semi-deciding whether or not a subset can be right-completed. The subroutine `SemiIsNotRightComp` in Algorithm 8 takes care of that. Whenever it can be detected locally that an expansion is unavoidable on the path to a maximal subset, one can skip the traversal of supersets not containing it. This idea is supported by the subroutine `IsInEachMax`. The run-time complexity is in $O\big((n(\tau(\texttt{IsLexMin}) + \tau(\texttt{IsInDownset}) + \tau(\texttt{SemiIsNotRightComp}) + \tau(\texttt{IsInEachMax})) + \tau(\texttt{IsMaxInDownset}))|\mathscr{D}^{\text{nonprunable}}/\mathfrak{G}|\big)$, where $\mathscr{D}^{\text{nonprunable}}$ is the subset of $\mathscr{D}$ containing subsets for which `SemiIsNotRightComp` returns "FALSE".

Other applications are rather interested in objects that can be better represented as the minimal subsets *not* contained in a downset. The enumeration of all circuits up to symmetry is an example for such a use-case.

Definition 4. For a subset $R \in 2^{[n]} \setminus \mathscr{D}$ a *reduction of R* is an element $i \in R$ so that $R \setminus \{i\} \in 2^{[n]} \setminus \mathscr{D}$. The subset R is *co-minimal w.r.t. $\mathscr{D}$* if it contains no reduction. It is *right-co-minimal* if its maximal element is not a reduction. A subset S in $\mathscr{D}$ is *right-exitable w.r.t. $\mathscr{D}$* if there is a set R that is co-minimal w.r.t. $\mathscr{D}$ so that R lex-contains S.

The idea for a reverse-search algorithm is to adapt `SymLexSubsetRSMax_withData`: add elements until the first non-member is met and check afterwards if the resulting set is a minimal non-member. Algorithm 9 shows a possible layout for this. The subroutine `SemiIsNotRightExit` can help to prune the tree. For the enumeration of circuits as in Section 8 there is no sensible such option known, but for other applications there may be one. The run-time complexity is in $O\big(n(\tau(\texttt{IsLexMin}) + \tau(\texttt{IsInDownset}) + \tau(\texttt{SemiIsNotRightExit}) + \tau(\texttt{IsCominOfDownset}))|\mathscr{D}^{\text{nonprunable}}/\mathfrak{G}|\big)$, where $\mathscr{D}^{\text{nonprunable}}$ is the subset of $\mathscr{D}$ containing subsets for which `SemiIsNotRightExit` returns "FALSE".

Occasionally, the representation of interesting objects as the maximal or co-minimal elements of a downset leads to a very difficult membership test. For example, it is NP-complete to decide whether or not a set of simplices is a subset of any triangulation. See Section 9 for more details. Then, it can be preferable to travers an auxiliary downset with simpler membership test that contains all feasible subsets and check for extendability to a feasible subset separately.

In the following, distinguish for a subset S of a feasible subset between an *expansion* (see Definition 3) of S by an element, which remains in $\mathscr{D}$, and an *extension* of S by an element, which remains a subset of a feasible subset. Most interesting is the case where the feasible subsets form an *antichain* in $[n]$, i.e., no feasible subset contains any other feasible subset.

Definition 5. Let $\mathscr{F}$ be an antichain in $2^{[n]}$. Members of $\mathscr{F}$ are called *feasible subsets.* Each $S \in \mathscr{F}$ is said to *represent a solution. Extendable subsets* are subsets of feasible subsets. A subset S is *right-extendable* if it is extendable to a feasible subset S' so that S' lex-contains S. In this case, S' is called a *feasible right-completion* of S.

```
Algorithm: SymLexSubsetRSMax_withData($n, \mathscr{D}, \mathfrak{G}, \mathbf{G}, N$)
Input: $n \in \mathbb{N}$, a downset $\mathscr{D}$ of $2^{[n]}$, a subgroup $\mathfrak{G}$ of the automorphism group of $\mathscr{D}$, global
        data $\mathbf{G}$, a node $N = (S, \mathbf{L})$ with a canonical $S \in \mathscr{D}$ and $\mathbf{L}$ local data
Output: the number of canonical $S'$ maximal in $\mathscr{D}$ with $S \subseteq_{\text{lex}} S'$
$c \leftarrow 0$ ;                                              /* do not count $S$ yet */
for $i = \max(S)+1, \dots, n$ do          /* ordered traversal of new maximal elements */
    $S' \leftarrow S \cup \{i\}$ ;                                /* add a new element on the right */
    default initialize $\mathbf{L}'$ ;                            /* prepare a local data structure */
    (answer, $\mathbf{L}'$) $\leftarrow$ IsLexMin($S', \mathfrak{G}, \mathbf{G}, \mathbf{L}, \mathbf{L}'$) ;              /* lex-min check */
    if answer = FALSE then             /* if new set is not lex-min in its orbit */
        continue ;                                             /* next loop element */
    (answer, $\mathbf{L}'$) $\leftarrow$ IsInDownset($S', \mathscr{D}, \mathbf{G}, \mathbf{L}, \mathbf{L}'$) ;          /* membership check */
    if answer = FALSE then                                   /* if new set is not in $\mathscr{D}$ */
        continue ;                                             /* next loop element */
    (answer, $\mathbf{L}'$) $\leftarrow$ SemiIsNotRightComp($S', \mathscr{D}, \mathbf{G}, \mathbf{L}, \mathbf{L}'$) ;       /* completability */
    if answer = TRUE then            /* if new set is not right-completable */
        continue ;                                             /* next loop element */
    $N' \leftarrow (S', \mathbf{L}')$ ;                                        /* build new node */
    $c \leftarrow c +$ SymLexSubsetRSMax_withData($n, \mathscr{D}, \mathfrak{G}, \mathbf{G}, N'$) ;           /* recurse */
    (answer, $\mathbf{L}'$) $\leftarrow$ IsInEachMax($S', \mathscr{D}, \mathbf{G}, \mathbf{L}, \mathbf{L}'$) ;       /* check unavoidability */
    if answer = TRUE then          /* if new set is in each right-completion */
        break ;                                                      /* exit the loop */
if $c > 0$ then                                          /* if supersets in $\mathscr{D}$ found */
    return $c$ ;                          /* return no. of max. supersets of $S$ in $\mathscr{D}$ */
if IsMaxInDownset($S, \mathbf{G}, \mathbf{L}$) then                         /* if $S$ is maximal in $\mathscr{D}$ */
    output $S$ and return 1 ;                                 /* return the count for $S$ */
```

Algorithm 8: A variant of symmetric lexicographic subset reverse search for maximal subsets with explicit use of global and local auxiliary data

If only a small fraction of the non-elements in a subset lead to an element of the downset, then the iteration over all new maximal elements followed by membership tests can lead to an unnecessarily large number of loop traversals. In the enumeration of triangulations this case occurs, since, in general, out of the many simplices not in a partial triangulation only a few can be added. In such cases it may be possible to compute all possible right-expansions of a subset before the loop and iterate only over those. The following notion supports this idea.

Definition 6. The *right-expansion sequence* $E(S)$ of S is the sequence $i_1 < \cdots < i_{|E(S)|}$ of all i_k, $k = 1, \dots, |E(S)|$, with $i_1 > \max S$ and $S \cup \{i_k\} \in \mathscr{D}$.

For the enumeration of feasible subsets with expensive exact extendability check it is desirable to prune the enumeration tree as soon as one knows that a subset is not right-

```
Algorithm: SymLexSubsetRSComin_withData($n, \mathscr{D}, \mathfrak{G}, \mathbf{G}, N$)
Input: $n \in \mathbb{N}$, a downset $\mathscr{D}$ of $2^{[n]}$, a subgroup $\mathfrak{G}$ of the automorphism group of $\mathscr{D}$, global
        data $\mathbf{G}$, a node $N = (S, \mathbf{L})$ with canonical $S \in \mathscr{D}$ and $\mathbf{L}$ local data
Output: the number of canonical $S'$ cominimal w.r.t. $\mathscr{D}$ with $S \subseteq_{\text{lex}} S'$
$c \leftarrow 0$ ;                                  /* as a member, $S$ cannot be co-minimal */
for $i = \max(S)+1, \dots, n$ do          /* ordered traversal of new maximal elements */
    $S' \leftarrow S \cup \{i\}$ ;                             /* add a new element on the right */
    default initialize $\mathbf{L}'$ ;                         /* prepare a local data structure */
    (answer, $\mathbf{L}'$) $\leftarrow$ IsLexMin($S', \mathfrak{G}, \mathbf{G}, \mathbf{L}, \mathbf{L}'$) ;                           /* lex-min check */
    if answer = FALSE then             /* if new set is not lex-min in its orbit */
        continue ;                                             /* next loop element */
    (answer, $\mathbf{L}'$) $\leftarrow$ IsInDownset($S', \mathscr{D}, \mathbf{G}, \mathbf{L}, \mathbf{L}'$) ;                /* membership check */
    if answer = FALSE then                           /* if new set is right-co-minimal */
        if IsCominOfDownset($S', \mathbf{G}, \mathbf{L}, \mathbf{L}'$) then               /* if set is co-minimal */
            output $S'$ and $c \leftarrow c+1$ ;                                   /* count $S'$ */
            continue ;                                         /* next loop element */
    (answer, $\mathbf{L}'$) $\leftarrow$ SemiIsNotRightExit($S', \mathscr{D}, \mathbf{G}, \mathbf{L}, \mathbf{L}'$) ;               /* exitability */
    if answer = TRUE then                        /* if new set is not right-exitable */
        continue ;                                             /* next loop element */
    $N' \leftarrow (S', \mathbf{L}')$ ;                                            /* build new node */
    $c \leftarrow c +$ SymLexSubsetRSComin_withData($n, \mathscr{D}, \mathfrak{G}, \mathbf{G}, N'$) ;           /* recurse */
return $c$;
```

Algorithm 9: A variant of symmetric lexicographic subset reverse search for co-minimal subsets with explicit use of global and local auxiliary data

extendable. This paper suggests an incomplete check that returns TRUE if a subset is not right-extendable and FALSE if the subset might be right-extendable. Algorithm 10 shows a possible pseudo-code for this method that will be used in the enumeration of all triangulations of a point configuration in Section 9. If $e^{\max}$ is the largest cardinality of any expansion sequence, then the run-time complexity is in $O\big((e^{\max}(\tau(\texttt{IsLexMin}) + \tau(\texttt{Expand}) + \tau\,\texttt{SemiIsNotRightExt}) + \tau(\texttt{IsFeasible}))|\mathscr{D}^{\text{nonprunable}}/\mathfrak{G}|\big)$, where $\mathscr{D}^{\text{nonprunable}}$ is the subset of $\mathscr{D}$ containing subsets for which SemiIsNotRightExt returns "FALSE".

4.2 New Checks for the Lexicographic Minimality of Subsets

Next, the findings from Section 3 are used to specify suitable local data to accelerate the lex-min check in orbits. In particular, the critical-element-table critelem_S of a subset will be used as local data stored together with S in a node. It can be seen that one can update the critical-element table for the next node during its usage in the lex-min check.

First, the iterative method IsLexMin_viaIter($S', \mathfrak{G}, \mathbf{G}, \mathbf{L}, \mathbf{L}'$) is described, where $\mathbf{L}$ contains CET, which is a function-value table of critelem_S. It neither needs global data $\mathbf{G}$ nor the

```
Algorithm: SymLexSubsetRSFeas_withData($n, \mathscr{D}, \mathscr{F}, \mathfrak{G}, \mathbf{G}, N$)
Input: $n \in \mathbb{N}$, a downset $\mathscr{D}$ of $2^{[n]}$, a subset $\mathscr{F} \subseteq \mathscr{D}$ of feasible subsets, a subgroup $\mathfrak{G}$ of the
       automorphism group of $\mathscr{D}$, global data $\mathbf{G}$, a node $N = (S, \mathbf{L})$ with canonical $S \in \mathscr{D}$ and
       $\mathbf{L}$ local data encompassing the right-expansion sequence $E(S)$
Output: the number of canonical $S'$ feasible in $\mathscr{D}$ with $S \subseteq_{\text{lex}} S'$
if IsFeasible($S, \mathscr{F}, \mathbf{G}, \mathbf{L}$) then                                  /* if $S$ is feasible */
    output $S$ and return 1 ;                                                         /* count $S$ */
$c \leftarrow 0$ ;                                                                    /* do not count $S$ */
for $j = 1, \dots, |E(S)|$ do                           /* ordered traversal of right-expansions */
    default initialize $\mathbf{L}'$ ;                            /* prepare a local data structure */
    (break, $S', \mathbf{L}'$) $\leftarrow$ Expand($S, E(S)_j, \mathbf{G}, \mathbf{L}, \mathbf{L}'$) ;          /* expand to the right */
    if break = TRUE then                                 /* if expansion returns to stop */
        break ;                                                              /* exit loop */
    (answer, $\mathbf{L}'$) $\leftarrow$ IsLexMin($S', \mathfrak{G}, \mathbf{G}, \mathbf{L}, \mathbf{L}'$) ;                  /* lex-min check */
    if answer = FALSE then             /* if new set is not lex-min in its orbit */
        continue ;                                                 /* next loop element */
    /* incomplete right-extendability check:                                          */
    (answer, $\mathbf{L}'$) $\leftarrow$ SemiIsNotRightExt($S', \mathscr{D}, \mathscr{F}, \mathbf{G}, \mathbf{L}, \mathbf{L}'$) ;
    if answer = TRUE then           /* if new set is certainly not right-ext.  */
        continue ;                                                 /* next loop element */
    $N' \leftarrow (S', \mathbf{L}')$ ;                                                  /* build new node */
    $c \leftarrow c +$ SymLexSubsetRSFeas_withData($n, \mathscr{D}, \mathfrak{G}, \mathbf{G}, N'$) ;          /* recurse */
return $c$ ;
```

Algorithm 10: A variant of symmetric lexicographic subset reverse search for feasible subsets with pruning by an incomplete extendability check

initialized local data $\mathbf{L}'$ for S'. In case the answer is "TRUE", the complete updated critical-element table of S' is stored in CET' representing a function-value table of $\text{critelem}_{S'}$, which will be part of $\mathbf{L}'$. In case the answer is "FALSE", CET' will only partially be updated. However, this is irrelevant, since this answer leads to immediate pruning. The function iterates over the whole symmetry group but is quite lean inside the loop. Algorithm 11 shows a possible listing in pseudo-code. Its run-time complexity amortized over all possible instances is in $O(|\mathfrak{G}|)$ under the assumption that on average for only a $\frac{1}{n}$-fraction of the permutations the added new maximal element of the new subset is mapped exactly to the critical element of the previous subset.

Next, the set-based variant `IsLexMin_viaSets`($S', \mathfrak{G}, \mathbf{G}, \mathbf{L}, \mathbf{L}'$) is described, where $\mathbf{L}$ contains CEC, which is a function-value table of critclass_S. Global data $\mathbf{G}$ containing ($\mathsf{HEC}, \mathsf{IEC}, \mathsf{DEC}$) are needed, which are function-value tables for hitclass, incclass, and decclass, respectively. It does not need the initialized local data $\mathbf{L}'$ for S'. Algorithm 12 shows the pseudo-code for this.

Finally, the modified switch-table method for larger symmetry groups is implemented.

```
Algorithm: IsLexMin_viaIter($S'$, $\mathfrak{G}$, CET)
Input: a set $S'$, a subgroup $\mathfrak{G}$ of $\mathfrak{S}_n$, and the critical-element table $\mathsf{CET} = \mathrm{critelem}_S$
       of $S = S' \setminus \max S'$
Output: $(\mathsf{TRUE}, \mathsf{CET}' = \mathrm{critelem}_{S'})$ if $S' = \text{lex-min}\,\mathfrak{G}(S')$ and $(\mathsf{FALSE}, -)$ otherwise
$\mathsf{CET}' \leftarrow \mathsf{CET}$;                       /* copy the old critical-element table */
for $\pi \in \mathfrak{G}$ do                                            /* iterate over $\mathfrak{G}$ */
    $j \leftarrow \mathsf{CET}[\pi]$;                                /* retrieve critical element */
    if $j = \infty$ then                                         /* if $\pi$ stabilizes $S$ */
        if $\pi(\max S') < \max S'$ then                        /* if $\pi$ decreases $\max S'$ */
            return $(\mathsf{FALSE}, -)$;                              /* $\pi$ is lex-decreasing */
        else if $\pi(\max S') > \max S'$ then                   /* if $\pi$ increases $\max S'$ */
            $\mathsf{CET}'[\pi] \leftarrow \max S'$;                       /* update critical element */
    else                                                /* $\pi$ strictly lex-increases $S$ */
        if $\pi(\max S') < j$ then                    /* if $\pi$ decreases $\max S'$ beyond $j$ */
            return $(\mathsf{FALSE}, -)$;                              /* $\pi$ is lex-decreasing */
        else if $\pi(\max S') = j$ then           /* if $\pi$ maps $\max S'$ to crit. element */
            $j' \leftarrow \min\big(S' \triangle \pi(S')\big)$;                /* compute critical element of $S'$ */
            if $j' \in \pi(S')$ then              /* if new critical element is in $\pi(S')$ */
                return $(\mathsf{FALSE}, -)$;                          /* $\pi$ is lex-decreasing */
            else                            /* if new critical element is not in $\pi(S')$ */
                $\mathsf{CET}'[\pi] \leftarrow j'$;                    /* update critical-element table */
return $(\mathsf{TRUE}, \mathsf{CET}')$;
```

Algorithm 11: The iteration-based critical-element method

Algorithm 13 is a utilization of Lemma 5 from Section 3. The advantage compared to Algorithm 6 is that it recursively removes elements from the subsets that play no role in the lex-comparison. The advantage compared to the algorithm in [24] is that one only uses a single group representation, namely a switch table, which avoids the creation and deletion of non-trivial local data structures, namely stabilizers, during the recursion.

It is clear that for the critical-element method a very large order of the symmetry group is prohibitive, so that in those cases the modified switch table method is mandatory. For groups of moderate order that can be stored completely in main memory, it is difficult to predict precisely which method is faster. This is mainly due to the fact that in an analysis of the recursion for the modified switch-table method it is difficult to exploit the group structure.

Therefore, as a simplification, a *hyper-amortized* analysis is presented indicating the parameter values for which the critical-element method is typically preferable. The idea is based on a drastically simplified probability model for the objects involved. In hyper-amortized analysis, the input subsets are considered uniformly and independently distributed among *all* possible subsets, not considering their extendability to feasible subsets

```
Algorithm: IsLexMin_viaSets($S', \mathfrak{G}$, (HEC, IEC, DEC), CEC)
Input: a set $S'$, a subgroup $\mathfrak{G}$ of $\mathfrak{S}_n$ represented by its hit-element, increasing-element, and
       decreasing-element classifications HEC, IEC, and DEC, the critical-element
       classification CEC $=$ critclass$_S$ of $S = S' \setminus \max S'$
Output: (TRUE, CEC$' =$ critclass$_{S'}$) if $S' = $ lex-min $\mathfrak{G}(S')$ and (FALSE, –) otherwise
if DEC[$\max S'$][$\max S'$] $\cap$ CEC[$\infty$] $\neq \emptyset$ then                /* check for case I */
    return (FALSE, –) ;                                          /* $S'$ is not lex-min */
if $\exists i \in S$ : DEC[$\max S'$][$i$] $\cap$ CEC[$i$] $\neq \emptyset$ then              /* check for case II */
    return (FALSE, –) ;                                          /* $S'$ is not lex-min */
CEC$'$ $\leftarrow$ CEC ;            /* copy the old critical-element classification */
for $j \in S$ do                                                    /* iterate over $S$ */
    for $\pi \in$ CEC[$j$] $\cap$ HEC[$\max S'$][$j$] do                  /* check $j$ for case III */
        $j' \leftarrow \min\big(S' \triangle \pi(S')\big)$ ;                 /* compute critical element of $S'$ */
        if $j' \in \pi(S')$ then               /* if new critical element is in $\pi(S')$ */
            return (FALSE, –) ;                              /* $\pi$ is lex-decreasing */
        else                           /* if new critical element is not in $\pi(S')$ */
            CEC$'$[$j$] $\leftarrow$ CEC$'$[$j$] $\setminus \{\pi\}$ ;                   /* update classification */
            CEC$'$[$j'$] $\leftarrow$ CEC$'$[$j'$] $\cup \{\pi\}$ ;                  /* update classification */
for $\pi \in$ IEC[$\max S'$] $\cap$ CEC[$\infty$] do          /* permutations no longer stabilizing */
    CEC$'$[$\infty$] $\leftarrow$ CEC$'$[$\infty$] $\setminus \{\pi\}$ ;                     /* update classification */
    CEC$'$[$\max S'$] $\leftarrow$ CEC$'$[$\max S'$] $\cup \{\pi\}$ ;                /* update classification */
return (TRUE, CEC$'$);
```

Algorithm 12: The set-based critical-element method

or their dependence on the earlier subsets. Finally, instead of a probability model for the possible permutation *groups*, a probability model for permutation *subsets* with uniform probabilities is considered. This probability model ignores that the subsets occurring as inputs actually are feasible subsets in a given downset and that they have been reached by the enumeration algorithm somehow. Moreover, it ignores that the actual sets of permutations form groups. Thus, the respective analysis cannot give the amortized effort for a particular problem instance. It can, however, be roughly interpreted as a higher-aggregated information about the average effort when the respective methods are applied to all conceivable problem instances with similar parameters, where all subsets are equally likely to occur as feasible subsets in *some* instance and where all permutations are individually equally likely to occur in *some* group. To stress the reference to the uniform probability model all expected values with respect to this model are called *typical values.*

For the analysis it is important to identify from which operations the main effort results. Profiling of actual code gives the hint that three types of operations cause the main part of the computation times: applying a permutation to a subset or a singleton, passing a copy of a subset as an argument to a recursive call, and lex-comparing a subset or a singleton with

```
Algorithm: IsLexMin_viaSwitchesMod($i, S, S', \mathfrak{G}$, ST)
Input: an integer $i \in [1,n]$, a subset $S$ of $\{i,\dots,n\}$ (the original subset), a subset $S'$ of
        $\{i,\dots,n\}$ (the image of $S$ under a partial switch product), a subgroup $\mathfrak{G}$ of $\mathfrak{S}_n$, and a
        switch table ST of $\mathfrak{G}$
Output: TRUE if lex-min $\mathfrak{G}_{[i-1]}(S') \not<_{\text{lex}} S$ and FALSE otherwise
if $|S| = 0$ or $|S| = n - i + 1$ then                    /* no or all elements?  */
    return TRUE ;                             /* subsets identical and invariant */
if $i > \max \text{effRowSet}(\text{ST})$ then        /* no non-trivial permutation in group?  */
    return $(S' \not<_{\text{lex}} S)$ ;                                  /* subset invariant */
/* case distinction whether $i \in S$:                                           */
if $i \in S$ then
    if $i \in S'$ then
        /* recursively check a switch product with leading identity:      */
        if IsLexMin_viaSwitchesMod($i+1, S \setminus \{i\}, S' \setminus \{i\}, \mathfrak{G}$, ST) = FALSE then
            return FALSE;
    /* travers all switches mapping an element of $S'$ to $i$:                */
    for $j \in \text{effColSet}(i) \cap S'$ do
        if IsLexMin_viaSwitchesMod($i+1, S \setminus \{i\}, \text{ST}[i][j](S' \setminus \{j\}), \mathfrak{G}$, ST) = FALSE then
            return FALSE;
    return TRUE ;                          /* no decreasing switch product found */
else
    /* compare minimal elements using $\min(S) > i$:                           */
    if $i \in S'$ then
        return FALSE;
    /* check for a switch mapping an element of $S'$ to $i$:                  */
    if $\text{effColSet}(i) \cap S' \neq \emptyset$ then
        return FALSE;
    /* recursively check a switch product with leading identity:          */
    if IsLexMin_viaSwitchesMod($i+1, S, S', \mathfrak{G}$, ST) = FALSE then
        return FALSE;
    /* travers all switches mapping a non-element of $S'$ to $i$:             */
    for $j \in \text{effColSet}(i) \setminus S'$ do
        if IsLexMin_viaSwitchesMod($i+1, S, \text{ST}[i][j](S'), \mathfrak{G}$, ST) = FALSE then
            return FALSE;
    return TRUE ;                          /* no decreasing switch product found */
```

Algorithm 13: The modified switch-table method

its image under a permutation. Applying a permutation π to a subset S means generating a new subset consisting of all $\pi(i)$ with $i \in S$. Passing a subset is essentially a special case with $\pi(i) = i$. Lex-comparing a subset or a singleton with its image requires the comparison of some the elements with image elements, which will not be counted extra, since for each comparison there is at least one image computation. The only substantial effort that does not lead to single-element evaluations is the lex-comparison of two subsets, which happens in exactly one spot in the modified switch-table method. In order to unify the effort estimation, we count m single-element evaluations for the lex-comparison of two m-subsets. In this generalized sense, the typical number of *single-element evaluations* $\pi(i)$ is investigated for permutations $\pi \in \mathfrak{G}$ and elements $i \in S$ during the execution of the algorithms.

In the following, the effort of `IsLexMin` for either method is estimated. In "FALSE"-instances the typical run-time is difficult to tell because it is not clear how early a lex-decreasing permutation will be found. Therefore, it is assumed for the analysis that any premature answer does not lead to an immediate stop; rather the algorithm is carried out until all relevant (depending on the method) permutations have been processed. This is equivalent to the typical effort for *counting* the lex-decreasing permutations.

The analysis of the naive method (check each permutation whether or not it lex-decreases the subset) and the critical-element method is straight-forward.

Theorem 1. *Consider subsets $\mathfrak{G}$ of $\mathfrak{S}_n$ with order k and degree n. Then:*

(i) The typical number $\sigma_{\text{nve}}(m)$ of single-element evaluations of the naive method on instances with an m-element subset S_{org} of `IsLexMin` *is $\sigma_{\text{nve}}(m) = km$.*

(ii) The typical number $\sigma_{\text{cet}}(m)$ of single-element evaluations of the iteration-based critical-element method on instances with an m-element subset S_{org} of `IsLexMin_viaIter` *is $\sigma_{\text{cet}}(m) = k\frac{n+m-1}{n} = \frac{n+m-1}{nm}km$.*

Proof. For the naive method, k permutations need to be applied to the m-element subset. For the critical-element method, typically $\frac{1}{n}k$ permutations map the new maximal element of S_{org} to the critical element, so that, in the worst case, m single-element evaluations are necessary to compute the new critical element from scratch. Moreover, typically $\frac{n-1}{n}k$ permutations do not map the maximal element of S_{org} to the critical element, incurring only one single-element evaluation each. □

Recall that the modified switch-table method takes two subsets of identical cardinality as input and decides whether or not the second can be lex-decreased below the first. The analysis for the modified switch-table now works in two steps. First, for a *given* switch-table the typical number of single-element evaluations is analyzed on two subsets chosen uniformly and independently at random.[3] Second, for a *typical* switch table of a random

[3]Note that this means already a simplification because for the applications of the modified switch-table in this paper, S_{org} and S_{img} are not stochastically independent. On the contrary: in the top-level call, they are identical. The hope is that over all instances that might ever be dealt with in any recursion level, they are sufficiently independent.

subset of permutations the typical number of non-trivial switches in each row and the typical maximal effective row are analyzed.

It is often more instructive to estimate the factor by which a (non-trivial) method for `IsLexMin` is faster than the naive method.

Definition 7. Let met be a method for `IsLexMin`. Then *relative effort* $\kappa_{\mathrm{met}}(\ell)$ of met is defined as the typical number of single-element evaluations $\sigma_{\mathrm{met}}(\ell)$ of met divided by the typical number $\sigma_{\mathrm{nve}}(\ell)$ of single-element evaluations of the naive method on random ℓ-element subsets.

Example 1. By Theorem 1, the relative effort of the critical-element method $\kappa_{cet}(\ell)$ is $\frac{n+m-1}{nm} \leq \frac{2}{m}$, which is a speed-up linear in $m < n$. □

It will turn out that the following parameters are crucial for the efficiency of a switch table. They all contribute to measures for how many times the various branches in the recursions are typically traversed.

Definition 8. Consider a switch table ST for a group of permutations of degree n with maximal effective row r and c_i non-trivial switches in row i, $i = 1, 2, \ldots, r$. Moreover, consider subsets of cardinality ℓ with $0 \leq \ell \leq m < n$.

The *level-i order* $k_{[i]}$ of ST is defined as the number of switch-products in ST from rows at least i, that is

$$k_{[i]} := \prod_{j=i}^{r} (c_j + 1). \tag{9}$$

With this, the order k of the group equals $k_{[1]}$.

The *level-i subset density and the level-i subset codensity of* ST are defined for $0 < \ell < n - i + 1$ as

$$\delta_{i,\ell} := \frac{\ell}{n-i+1} \text{ and } \bar{\delta}_{i,\ell} := \frac{n-i+1-\ell}{n-i+1} = 1 - \delta_{i,\ell}\text{, respectively.} \tag{10}$$

For $\ell \leq 0$ and $\ell \geq n - i + 1$ define the exceptional values

$$\delta_{i,\ell} := \bar{\delta}_{i,\ell} := 0. \tag{11}$$

The *level-i transitivity gap of* ST is defined for $0 \leq \ell \leq n - i - c_i$ as

$$\vartheta_{i,\ell} := \frac{\big(n-i+1-(c_i+1)\big)_\ell}{(n-i+1)_\ell}, \tag{12}$$

where $(x)_\ell := x(x-1)\cdots(x-\ell+1)$, as usual. For $\ell > n-i+1-(c_i+1)$ the transitivity gap is defined to be zero. In algorithm `IsLexMin_viaSwitchesMod`, call the branch at recursion level i with $i \in S_{\mathrm{org}}$ the *level-i element branch* and the branch with $i \notin S_{\mathrm{org}}$ the *level-i non-element branch.* The *level-i reduction factor of the element branch of* ST is defined as

$$\alpha^{(1)}_{i,\ell} := \delta^2_{i,\ell}, \tag{13}$$

and the *level-i reduction factor of the non-element branch of* ST is defined as

$$\alpha^{(0)}_{i,\ell} := \bar{\delta}_{i,\ell}\,\vartheta_{\ell,}. \tag{14}$$

For $p \in \mathbb{N}$, a *branch-selection vector of length p* is a binary vector $\mathbf{b} = (b_1, b_2, \dots, b_p) \in \{0,1\}^p$, where () is the only branch selection vector for $p = 0$. Let $\mathscr{B}(p)$ be the set of all branch-selection vectors of length p with $\mathscr{B}(p) = \emptyset$ for all $p < 0$. Let $\lambda(\mathbf{b})$ be the number of ones in $\mathbf{b}$, and denote by $\lambda_j(\mathbf{b})$ the number of ones in $\mathbf{b}$ among the first $j-1$ coordinates of $\mathbf{b}$, $1 \le j \le p$.

For $i \in [r]$ and a branch-selection vector of length $p \le r-i+1$ let its *level-i branch weight* $w_{i,\ell}(\mathbf{b})$ be the following product of p reduction factors, where an empty product ($p = 0$) is defined to be one:

$$w_{i,\ell}(\mathbf{b}) := \prod_{t=1}^{p} \alpha^{(b_t)}_{i+t-1,\ell-\lambda_t(\mathbf{b})}. \tag{15}$$

Finally, let its *node weight* be

$$v_\ell(\mathbf{b}) := (\ell - \lambda(\mathbf{b}))^+ = \max(\ell - \lambda(\mathbf{b}), 0). \tag{16}$$

These notions are motivated by the following: The subset density and codensity determine the probabilities with which the two branches in the modified switch-table method are entered. The exceptional zero-values for both model the cases where the modified switch-table method returns TRUE without entering any branch. The transitivity gaps indicate the fraction of ℓ-subsets for which no image of a switch in row i of ST contains i. For example, in a switch table of a transitive group, the level-1-transitivity gap is zero even for $\ell = 1$ (one-element subsets), because any element can be mapped to 1. The reduction factors compute the probabilities that one of the switches in a row has to be processed during the execution of the algorithm at the corresponding recursion level. A branch selection vector of length p encodes for recursion levels $i, i+1, \dots, i+p-1$ and a fixed S_{org} which of the branches is entered by the algorithm: A "1" selects the element-branch, a "0" selects the non-element branch. In this sense, all paths in the recursion tree are indexed by branch-selection vectors. The branch weights are products of reduction factors, where the first index (indicating the row of ST) is increased with each new factor, and the second index (indicating the cardinality of a processed subset) is decreased directly after an element branch has been entered.

With these parameters, all of which can easily be derived directly from the switch table and the subsets under consideration, the following can be said about its efficiency:

Theorem 2. *Let* ST *be a switch table for permutations in $\mathfrak{S}_n$ with maximal effective row r and c_i non-trivial switches in row i for $i = 1, 2, \dots, r$. Let $\mathscr{B}(i)$ be the set of all branch-selection vectors of length i, $1 \le i \le r$.*

Then, the typical number $\sigma_{\text{mst}}(m)$ of single-element evaluations for two independently random m-subsets $S_{\text{org}}, S_{\text{img}}$ is

$$\sigma_{\text{mst}}(m) = \Big(\sum_{\mathbf{b}\in\mathscr{B}(r)} w_{1,m}(\mathbf{b})v_m(\mathbf{b})\Big)k + \sum_{j=1}^{r}\bigg(\Big(\sum_{\mathbf{b}\in\mathscr{B}(j)} w_{1,m}(\mathbf{b})v_m(\mathbf{b})\Big)\frac{c_j}{c_j+1}\cdot\frac{k}{k_{[j+1]}}\bigg) \tag{17}$$

Remark 1. Since the rth summand of the second sum is actually $\frac{c_r}{c_r+1}$ times the first summand, an alternative expression for $\sigma_{\mathrm{mst}}(m)$ is:

$$\sigma_{\mathrm{mst}}(m) = \Big(1 + \frac{c_r}{c_r+1}\Big)\Big(\sum_{\mathbf{b}\in\mathscr{B}(r)} w_{1,m}(\mathbf{b})v_m(\mathbf{b})\Big)k \tag{18}$$

$$+\sum_{j=1}^{r-1}\bigg(\Big(\sum_{\mathbf{b}\in\mathscr{B}(j)} w_{1,m}(\mathbf{b})v_m(\mathbf{b})\Big)\frac{c_j}{c_j+1}\cdot\frac{k}{k_{[j+1]}}\bigg). \tag{19}$$

Proof. Let ST be as in the assumptions of the theorem. For $1 \le i \le r$ and $1 \le \ell \le m < n$ call $f_i(\ell)$ the number of single-element evaluations the modified switch-table method with start row i typically needs on two random ℓ-element subsets S_{org} and S_{img} of $[n]\backslash[i-1]$. For $i = r+1$, no non-trivial switches are available, and after one lex-comparison of the given ℓ-element subsets incurring ℓ single-element evaluations the answer is immediate. Consequently, $f_{r+1}(\ell) = \ell$. Moreover, if $\ell = 0$ and $\ell = n-i+1$ both sets are either the empty or the complete subset, thus fixed under any action of a switch in row i and higher, and the recursion subtree needs no further single-element evaluations, leading to $f_i(0) = f_i(n-i+1) = 0$.

The element-branch in the modified switch-table method has $i \in S_{\mathrm{org}}$. Given a random S_{org}, this branch is entered with probability $\frac{\ell}{n-i+1} = \delta_{i,\ell}$.

In the element branch, the identity is considered only if $i \in S_{\mathrm{img}}$, and non-trivial switches $\mathsf{ST}[i][j]$ are considered only if $j \in S_{\mathrm{img}}$. For fixed $j \ge i$ one has $j \in S_{\mathrm{img}}$ with probability $\frac{\ell}{n-i+1} = \delta_{i,\ell}$.[4] Thus, only a fraction of $\delta_{i,\ell}$ of the (c_i+1) many switches including one identity in row i are processed at all. Processing a switch incurs $\ell-1$ single-element evaluations for the non-trivial switches plus a recursive call with two $\ell-1$-element subsets for row $i+1$ of the switch table for the c_i non-trivial switches plus the identity. The typical number of single-element evaluations in the element branch is, therefore, $\delta_{i,\ell}\big(c_i(\ell-1)+(c_i+1)f_{i+1}(\ell-1)\big)$.

In the non-element branch with $i \notin S_{\mathrm{org}}$, the answer is immediately FALSE in this branch whenever $i \in S_{\mathrm{img}}$ or there exists a non-trivial switch $\mathsf{ST}[i][j]$ with $j \in S_{\mathrm{img}}$; in this case there are no subsequent single-element evaluations in this branch. If all ℓ elements of $S_{\mathrm{img}} \subseteq \{i, i+1, \ldots, n\}$ are among the trivial columns $j > i$, then the remaining switches have to be processed. Therefore, with probability $\frac{(n-i+1-(c_i+1))_\ell}{(n-i+1)_\ell} = \vartheta_{i,\ell}$ neither the c_i non-trivial switches nor the identity lead to an immediate FALSE in this branch. This event incurs ℓ single-element evaluations for all non-trivial switches plus a recursive call with two ℓ-element subsets with row $i+1$ for all non-trivial switches and the identity. Hence, the typical number of single-element evaluations in the non-element branch is $\vartheta_{i,\ell}\big(c_i\ell + (c_i+1)f_{i+1}(\ell)\big)$.

Summarized, the following recursion for $f_i(\ell)$ is obtained with $f_1(m) = \sigma_{\mathrm{mst}}(m)$:

$$f_i(\ell) = \alpha^{(1)}_{i,\ell}\big(c_i(\ell-1) + (c_i+1)f_{i+1}(\ell-1)\big) + \alpha^{(0)}_{i,\ell}\big(c_i\ell + (c_i+1)f_{i+1}(\ell)\big). \tag{20}$$

[4]At this point a difference occurs for $S_{\mathrm{img}} = S_{\mathrm{org}}$, which occurs in the lex-min check for the case $i = 1$ and $\ell = m$. Then, $1 \in S_{\mathrm{org}}$ implies (with probability one) $1 \in S_{\mathrm{img}}$, and, therefore, the probability that the identity has to be processed is 1 instead of $\delta_{1,m}$, changing the expression $\delta^2_{1,m}(c_1+1)$ to $\delta_{1,m}(\delta_{1,m}\,c_1+1)$ in the results below. This increases the typical number of single-element evaluations in the first recursion level. The difference is the larger the smaller c_1 is. Since this leads to more case-distinctions, in this paper only the analysis for two independently random subset is carried out.

Recall that the final comparison of ℓ-subsets at maximal recursion level $r+1$ as well as the empty-set case $\ell = 0$ and the full-set case $\ell = n - i + 1$ lead to the boundary conditions

$$f_{r+1}(\ell) = \ell \text{ for all } 1 \le \ell \le m, \quad f_i(0) = f_i(n-i+1) = 0 \text{ for all } 1 \le i \le r. \tag{21}$$

In order to prove the theorem, it is sufficient show that the following function g is a solution to the recursion for f, since the claim of the theorem can then be expressed as $\sigma_{\text{mst}}(m) = g_1(m)$:

$$g_i(\ell) := \Bigg(\sum_{\mathbf{b}\in\mathscr{B}(r-i+1)} w_{i,\ell}(\mathbf{b})v_\ell(\mathbf{b})\Bigg)k_{[i]} + \sum_{j=i}^{r}\Bigg(\Bigg(\sum_{\mathbf{b}\in\mathscr{B}(j-i+1)} w_{i,\ell}(\mathbf{b})v_\ell(\mathbf{b})\Bigg)\frac{c_j}{c_j+1}\cdot\frac{k_{[i]}}{k_{[j+1]}}\Bigg). \tag{22}$$

To this end, note that the following recursion holds for all $1 \le i \le r$ and $1 \le \ell \le m$ by definition for the products of branch weights and node weights:

$$\sum_{\mathbf{b}\in\mathscr{B}(i)} w_{i,\ell}(\mathbf{b})v_\ell(\mathbf{b}) = \alpha^{(1)}_{i,\ell}\sum_{\mathbf{b}\in\mathscr{B}(i-1)} w_{i+1,\ell-1}(\mathbf{b})v_{\ell-1}(\mathbf{b}) + \alpha^{(0)}_{i,\ell}\sum_{\mathbf{b}\in\mathscr{B}(i-1)} w_{i+1,\ell}(\mathbf{b})v_\ell(\mathbf{b}). \tag{23}$$

Moreover, the following recursion holds for the level-i orders:

$$k_{[i]} = (c_i+1)k_{[i+1]}. \tag{24}$$

The boundary condition $g_{r+1}(\ell) = \ell$ holds true, because the boundary cases $k_{[r+1]} = 1$, $w_{i,\ell}(()) = 1$, and $v_\ell(()) = \ell$ imply

$$\begin{aligned}
&g_{r+1}(\ell) && (25)\\
&= \Bigg(\sum_{\mathbf{b}\in\mathscr{B}(0)} w_{i,\ell}(\mathbf{b})v_\ell(\mathbf{b})\Bigg)k_{[r+1]} + \sum_{j=r+1}^{r}\Bigg(\Bigg(\sum_{\mathbf{b}\in\mathscr{B}(j-r)} w_{i,\ell}(\mathbf{b})v_\ell(\mathbf{b})\Bigg)\frac{c_j}{c_j+1}\cdot\frac{k_{[i]}}{k_{[j+1]}}\Bigg) && (26)\\
&= w_{r+1,\ell}(())v_\ell(()) && (27)\\
&= \ell. && (28)
\end{aligned}$$

The boundary conditions $g_i(0) = 0$ and $g_i(n-i+1) = 0$ hold true, because $\delta_{i,n-i+1} = \bar{\delta}_{i,n-i+1} = 0$ leading to zero reduction factors throughout. Moreover, $g_i(\ell)$ satisfies the recursion formula for $1 \le \ell \le m$ and $1 \le i \le r$, because

$$\begin{aligned}
g_i(\ell) &= \Bigg(\sum_{\mathbf{b}\in\mathscr{B}(r-i+1)} w_{i,\ell}(\mathbf{b})v_\ell(\mathbf{b})\Bigg)k_{[i]} + \sum_{j=i}^{r}\Bigg(\sum_{\mathbf{b}\in\mathscr{B}(j-i+1)} w_{i,\ell}(\mathbf{b})v_\ell(\mathbf{b})\Bigg)\frac{c_j}{c_j+1}\cdot\frac{k_{[i]}}{k_{[j+1]}} && (29)\\
&\overset{(23),(24)}{=} \Bigg(\alpha^{(1)}_{i,\ell}\sum_{\mathbf{b}\in\mathscr{B}(r-i)} w_{i+1,\ell-1}(\mathbf{b})v_{\ell-1}(\mathbf{b}) + \alpha^{(0)}_{i,\ell}\sum_{\mathbf{b}\in\mathscr{B}(r-i)} w_{i+1,\ell}(\mathbf{b})v_\ell(\mathbf{b})\Bigg)(c_i+1)k_{[i+1]} && (30)\\
&\quad + \sum_{j=i+1}^{r}\Bigg(\Bigg(\alpha^{(1)}_{i,\ell}\sum_{\mathbf{b}\in\mathscr{B}(j-i)} w_{i+1,\ell-1}(\mathbf{b})v_{\ell-1}(\mathbf{b}) + \alpha^{(0)}_{i,\ell}\sum_{\mathbf{b}\in\mathscr{B}(j-i)} w_{i+1,\ell}(\mathbf{b})v_\ell(\mathbf{b})\Bigg) && (31)\\
&\qquad\qquad \cdot\frac{c_j}{c_j+1}\cdot\frac{(c_i+1)k_{[i+1]}}{k_{[j+1]}}\Bigg) && (32)
\end{aligned}$$

$$+\sum_{\mathbf{b}\in\mathscr{B}(1)} w_{i,\ell}(\mathbf{b})v_\ell(\mathbf{b})\frac{c_i}{c_i+1}\cdot\frac{(c_i+1)k_{[i+1]}}{k_{[i+1]}} \tag{33}$$

$$=\alpha^{(1)}_{i,\ell}\Bigg(\sum_{\mathbf{b}\in\mathscr{B}(r-i)} w_{i+1,\ell-1}(\mathbf{b})v_{\ell-1}(\mathbf{b})\Bigg)(c_i+1)k_{[i+1]} \tag{34}$$

$$+\alpha^{(0)}_{i,\ell}\Bigg(\sum_{\mathbf{b}\in\mathscr{B}(r-i)} w_{i+1,\ell}(\mathbf{b})v_{\ell}(\mathbf{b})\Bigg)(c_i+1)k_{[i+1]} \tag{35}$$

$$+\alpha^{(1)}_{i,\ell}\sum_{j=i+1}^{r}\Bigg(\sum_{\mathbf{b}\in\mathscr{B}(j-i)} w_{i+1,\ell-1}(\mathbf{b})v_{\ell-1}(\mathbf{b})\Bigg)\frac{c_j}{c_j+1}\cdot\frac{(c_i+1)k_{[i+1]}}{k_{[j+1]}} \tag{36}$$

$$+\alpha^{(0)}_{i,\ell}\sum_{j=i+1}^{r}\Bigg(\sum_{\mathbf{b}\in\mathscr{B}(j-i)} w_{i+1,\ell}(\mathbf{b})v_{\ell}(\mathbf{b})\Bigg)\frac{c_j}{c_j+1}\cdot\frac{(c_i+1)k_{[i+1]}}{k_{[j+1]}} \tag{37}$$

$$+c_i\,\underbrace{w_{i,\ell}((1))}_{=\alpha^{(1)}_{i,\ell}}\underbrace{v_\ell((1))}_{=\ell-1}+c_i\,\underbrace{w_{i,\ell}((0))}_{=\alpha^{(0)}_{i,\ell}}\underbrace{v_\ell((0))}_{=\ell} \tag{38}$$

$$=\alpha^{(1)}_{i,\ell}(c_i+1)g_{i+1}(\ell-1)+\alpha^{(0)}_{i,\ell}(c_i+1)g_{i+1}(\ell)+\alpha^{(1)}_{i,\ell}\,c_i(\ell-1)+\alpha^{(0)}_{i,\ell}\,c_i(\ell). \tag{39}$$

Thus, g satisfies recursion (20), which completes the proof. □

Theorem 2 can be used to derive simpler upper and lower bounds for the application of the modified switch-table method. In order to prepare this, some monotonicity properties of the quantities involved in Theorem 2 and Remark 1, respectively, are presented. Note, that the monotonicity properties below do not imply any kind of monotonicity for the function $f_i(\ell)$ from the previous proof. In fact, it is not monotone at all, as plotting experiments easily reveal. Thus, the exact evaluation of the recursion in Theorem 2 is the key to simpler bounds.

Lemma 6. *For all $1\le j\le r\le m$, all $1\le i\le j$, all $m-j+1\le\ell\le m$, and all $\mathbf{b}\in\mathscr{B}(j)$ the following hold:*

(i) $\delta_{1,m-j+1}\le\delta_{i,\ell}\le\delta_{j,m}$.

(ii) $\bar{\delta}_{j,m}\le\bar{\delta}_{i,\ell}\le\bar{\delta}_{1,m-j+1}$.

(iii) $\vartheta_{j,m}\le\vartheta_{i,\ell}\le\vartheta_{1,m-j+1}$.

(iv) $\alpha^{(1)}_{1,m-j+1}\le\alpha^{(1)}_{i,\ell}\le\alpha^{(1)}_{m,j}$.

(v) $\alpha^{(0)}_{j,m}\le\alpha^{(0)}_{i,\ell}\le\alpha^{(0)}_{1,m-j+1}$.

(vi) $\left(\alpha^{(1)}_{1,m-j+1}\right)^{\lambda(\mathbf{b})}\left(\alpha^{(0)}_{j,m}\right)^{j-\lambda(\mathbf{b})}\le w_{1,m}(\mathbf{b})\le\left(\alpha^{(1)}_{j,m}\right)^{\lambda(\mathbf{b})}\left(\alpha^{(0)}_{1,m-j+1}\right)^{j-\lambda(\mathbf{b})}$.

(vii) $m-j\le v_m(\mathbf{b})\le m$.

Proof. All claims follow directly from the definitions of the respective quantities. □

From this, the following conclusion can be drawn.

Corollary 1. *Let* ST *be a switch table for permutations in* $\mathfrak{S}_n$ *with maximal effective row* r *and* c_i *non-trivial switches in row* i *for* $i = 1, 2, \dots, r$. *Let* $\mathscr{B}(i)$ *be the set of all branch-selection vectors of length* i, $1 \le i \le r$.

Then, the typical number $\sigma_{\text{mst}}(m)$ *of single-element evaluations for two independently random* m*-subsets* $S_{\text{org}}, S_{\text{img}}$ *with* $m \ge r$ *is bounded as follows:*

$$\Big(1 + \frac{c_r}{c_r+1}\Big)\Big(\alpha^{(1)}_{1,m-r+1} + \alpha^{(0)}_{r,m}\Big)^r k(m-r) \tag{40}$$

$$+ \sum_{j=1}^{r-1}\Big(\alpha^{(1)}_{1,m-j+1} + \alpha^{(0)}_{j,m}\Big)^j \frac{c_j}{c_j+1} \cdot \frac{k(m-j)}{k_{[j+1]}} \tag{41}$$

$$\le \sigma_{\text{mst}}(m) \tag{42}$$

$$\le \Big(1 + \frac{c_r}{c_r+1}\Big)\Big(\alpha^{(1)}_{r,m} + \alpha^{(0)}_{1,m-r+1}\Big)^r km \tag{43}$$

$$+ \sum_{j=1}^{r-1}\Big(\alpha^{(1)}_{j,m} + \alpha^{(0)}_{1,m-j+1}\Big)^j \frac{c_j}{c_j+1} \cdot \frac{km}{k_{[j+1]}}. \tag{44}$$

Proof. Substituting the monotonicity properties 6 (vi) and (vii) for branch and node weights into the formula in Remark 1 yields the following:

$$\Big(1 + \frac{c_r}{c_r+1}\Big)\Bigg(\sum_{\lambda=0}^{r}\binom{\lambda}{r}\Big(\alpha^{(1)}_{1,m-r+1}\Big)^{\lambda}\Big(\alpha^{(0)}_{r,m}\Big)^{r-\lambda}\Bigg)(m-r)k \tag{45}$$

$$+ \sum_{j=1}^{r-1}\Bigg(\sum_{\lambda=0}^{j}\binom{\lambda}{r}\Big(\alpha^{(1)}_{1,m-j+1}\Big)^{\lambda}\Big(\alpha^{(0)}_{j,m}\Big)^{j-\lambda}\Bigg)(m-j) \cdot \frac{c_j}{c_j+1} \cdot \frac{k}{k_{[j+1]}} \tag{46}$$

$$\le \sigma_{\text{mst}}(m) \tag{47}$$

$$\le \Big(1 + \frac{c_r}{c_r+1}\Big)\Bigg(\sum_{\lambda=0}^{r}\binom{\lambda}{r}\Big(\alpha^{(1)}_{r,m}\Big)^{\lambda}\Big(\alpha^{(0)}_{1,m-r+1}\Big)^{r-\lambda}\Bigg)mk \tag{48}$$

$$+ \sum_{j=1}^{r-1}\Bigg(\sum_{\lambda=0}^{j}\binom{\lambda}{r}\Big(\alpha^{(1)}_{j,m}\Big)^{\lambda}\Big(\alpha^{(0)}_{1,m-j+1}\Big)^{j-\lambda}\Bigg)m \cdot \frac{c_j}{c_j+1} \cdot \frac{k}{k_{[j+1]}}. \tag{49}$$

The assertion now follows directly from an application of the binomial theorem. □

One key learning can be derived right away: Whenever the sum of the two respective largest possible reduction factors in the element and non-element branches is strictly smaller than one (which is clear for identical parameters and is the common case otherwise), then the gain of the switch table is exponential in r at the cost of an overhead that is essentially linear in r. For a small r the overhead may outweigh the gain. For a large r the gain will in most cases outweigh the overhead. Note that conclusions based on asymptotics in r alone may be misleading, since r is not an unbounded, free input parameter but bounded by n

and determined by both the structure of the symmetry group and the order of the elements in $[n]$. In the third application of this paper (triangulations, see Section 9), r is often small, as will be shown later in this section.

The following result shows that if the reduction factors are uniformly close to $\frac{1}{4}$, then the gain of the switch table is substantial.

Corollary 2. *Let* ST *be a switch table for a subgroup $\mathfrak{G}$ of $\mathfrak{S}_n$ with maximal effective row r and c_i non-trivial switches in row i, $i = 1,2,\dots,r$.*

Assume there is a $\mu \in [1,2)$ with

- $\alpha^{(1)}_{i,\ell} \le \frac{\mu}{4}$ *and* $\alpha^{(0)}_{i,\ell} \le \frac{\mu}{4}$ *for all $i = 1,2,\dots,r$ and $\ell = m-r+1, m-r+2,\dots,m$;*
- $c_i > 0$ *for all $i = 1,2,\dots,r$.*

Then, the relative effort $\kappa_{mst}(m)$ of ST *is at most $(r+1)\left(\frac{\mu}{2}\right)^r$, i.e.,*

$$\sigma_{\mathrm{mst}}(m) \le (r+1)\left(\frac{\mu}{2}\right)^r km = (r+1)\left(\frac{\mu}{2}\right)^r \sigma_{\mathrm{nve}}(m). \tag{50}$$

Moreover, these assumptions are, for example, satisfied if $\delta_{r,m} = \frac{m}{n-r+1} \le \frac{\sqrt{2-\mu}}{2}$ and

$$c_i + 1 \ge \left(1 - \sqrt[m-i+1]{\frac{\mu(n-i+1)}{4(n-m)}}\right)(n-i+1) \textit{ for all } i = 1,2,\dots,r. \tag{51}$$

Proof. The starting point is Corollary 1. Since $c_i > 0$ for all $i = 1,2,\dots,r$, the level-$i+1$ order $k_{[i+1]}$ is at least 2^{r-i}. This implies $\frac{k}{k_{[j+1]}} \le \left(\frac{1}{2}\right)^{r-j} k$. Moreover, $\frac{c_j}{c_j+1} < 1$. Therefore:

$$\sigma_{\mathrm{mst}}(m) \le \left(1 + \frac{c_r}{c_r+1}\right)\left(\alpha^{(1)}_{r,m} + \alpha^{(0)}_{1,m-r+1}\right)^r km \tag{52}$$

$$+ \sum_{j=1}^{r-1}\left(\alpha^{(1)}_{j,m} + \alpha^{(0)}_{1,m-j+1}\right)^j \frac{c_j}{c_j+1} \cdot \frac{km}{k_{[j+1]}} \tag{53}$$

$$\le 2 \cdot \left(\frac{\mu}{2}\right)^r km + \left(\sum_{j=1}^{r-1}\left(\frac{\mu}{2}\right)^j \cdot 1 \cdot \left(\frac{1}{2}\right)^{r-j}\right) km \tag{54}$$

$$\overset{(\mu\ge1)}{\le} \left(2\left(\frac{\mu}{2}\right)^r + \sum_{j=1}^{r-1}\left(\frac{\mu}{2}\right)^j\left(\frac{\mu}{2}\right)^{r-j}\right) km \tag{55}$$

$$= \left(2\left(\frac{\mu}{2}\right)^r + (r-1)\left(\frac{\mu}{2}\right)^r\right) km \tag{56}$$

$$= (r+1)\left(\frac{\mu}{2}\right)^r km. \tag{57}$$

If the modified switch-table method is called on m-element subsets, then the minimal ℓ in a reduction factor of level i is $m-i+1$ and the maximal ℓ is m. Therefore, $\delta_{i,\ell} \le \delta_{i,m} \le \frac{\sqrt{2-\mu}}{2}$ implies $\alpha^{(1)}_{i,\ell} = \delta^2_{i,\ell} \le \frac{2-\mu}{4} \le \frac{1}{4} \le \frac{\mu}{4}$. Moreover, $\frac{\mu}{4} < \frac{1}{2} \le 1 - \frac{\sqrt{2-\mu}}{2} \le 1 - \delta_{i,\ell} =$

$\bar{\delta}_{i,\ell} \leq \bar{\delta}_{i,m-i+1} = \frac{n-i+1-(m-i+1)}{n-i+1} = \frac{n-m}{n-i+1}$. This implies $\frac{\mu(n-i+1)}{4(n-m)} < 1$, and, hence, $\sqrt[\ell]{\frac{\mu(n-i+1)}{4(n-m)}}$ is monotonically increasing in ℓ.

With this, one can conclude for all $\ell = m, m-1, \ldots, m-i+1$ and all $i = 1, 2, \ldots, r$:

$$c_i + 1 \geq \left(1 - \sqrt[m-i+1]{\frac{\mu(n-i+1)}{4(n-m)}}\right)(n-i+1) \tag{58}$$

$$\Rightarrow c_i + 1 \geq \left(1 - \sqrt[\ell]{\frac{\mu}{4\bar{\delta}_{i,\ell}}}\right)(n-i+1) \tag{59}$$

$$\Rightarrow c_i + 1 \geq (n-i+1) - (n-i+1)\sqrt[\ell]{\frac{\mu}{4\bar{\delta}_{i,\ell}}} \tag{60}$$

$$\Rightarrow \left(\frac{n-i+1-(c_i+1)}{n-i+1}\right)^{\ell} \leq \frac{\mu}{4\bar{\delta}_{i,\ell}} \tag{61}$$

$$\Rightarrow \bar{\delta}_{i,\ell}\left(\frac{\left(n-i+1-(c_i+1)\right)_{\ell}}{(n-i+1)_{\ell}}\right) \leq \frac{\mu}{4} \tag{62}$$

$$\Rightarrow \alpha^{(0)}_{i,\ell} = \bar{\delta}_{i,\ell}\vartheta_{i,\ell} \leq \frac{\mu}{4}. \tag{63}$$

□

The following result shows that if the reduction factors are not too much smaller than $\frac{1}{2}$ and r is sufficiently smaller than m, then the switch table may even be slower than the naive method.

Corollary 3. *Let* ST *be a switch table for a subgroup* $\mathfrak{G}$ *of* $\mathfrak{S}_n$ *with maximal effective row* r *and* c_i *non-trivial switches in row* i, $i = 1, 2, \ldots, r$.

Assume that there is a $\mu \in (0, 1]$ *with* $\alpha^{(1)}_{i,\ell} \geq \frac{\mu}{2}$ *and* $\alpha^{(0)}_{i,\ell} \geq \frac{\mu}{2}$ *for all* $i = 1, 2, \ldots, r$ *and* $\ell = m-r, m-r+1, \ldots, m$. *Then, the relative effort* $\kappa_{mst}(m)$ *of* ST *is at least* $\frac{3}{2}\mu^r\left(1-\frac{r}{m}\right)$, *i.e.,*

$$\sigma_{\text{mst}}(m) \geq \frac{3}{2}\mu^r\left(1-\frac{r}{m}\right)km = \frac{3}{2}\mu^r\left(1-\frac{r}{m}\right)\sigma_{\text{nve}}(m). \tag{64}$$

In particular, if additionally $\mu > \sqrt[r]{\frac{2}{3}}$ *and* $r < m\left(1-\frac{2}{3\mu^r}\right)$, *then the modified switch-table method typically needs more single-element evaluations on* m*-element subsets than the naive method.*

Proof. The starting point is again Corollary 1. The fact that $c_r > 0$ implies $\frac{c_r}{c_r+1} \geq \frac{1}{2}$. Therefore:

$$\sigma_{\text{mst}}(m) \geq \left(1 + \frac{c_r}{c_r+1}\right)\left(\alpha^{(1)}_{1,m-r+1} + \alpha^{(0)}_{r,m}\right)^r k(m-r) \tag{65}$$

$$+ \sum_{j=1}^{r-1}\left(\alpha^{(1)}_{1,m-j+1} + \alpha^{(0)}_{j,m}\right)^j \frac{c_j}{c_j+1} \cdot \frac{k(m-j+1)}{k_{[j+1]}} \tag{66}$$

$$\geq \frac{3}{2} \cdot \mu^r \cdot 2^r \cdot \left(\frac{1}{2}\right)^r k(m-r) \tag{67}$$

$$\geq \frac{3}{2}\mu^r k(m-r) \tag{68}$$

$$= \frac{3}{2}\mu^r \left(1 - \frac{r}{m}\right) km. \tag{69}$$

The additional claim follows from solving $\frac{3}{2}\mu^r\left(1-\frac{r}{m}\right) > 1$. □

Since the assumptions in both Corollaries 2 and 3 are somewhat unrealistic (uniformly bounded reduction factors are rare in practice), the focus of the following is on bounds that actually explain the difference in performance for important use cases, namely the enumeration of circuits and triangulations of hypercubes.

A closer inspection of the non-element reduction factors reveals that the transitivity of a symmetry group has a positive influence on the performance of the modified switch-table method. An important example for this are the symmetry groups of hypercubes: for each pair of vertices of the hypercube there is a permutation in its symmetry group that maps one vertex to the other. Switch tables allow for a somewhat continuous measure for the transitivity of a symmetry group: the level-i transitivity gaps. A switch table encodes a transitive group if and only if $n-(c_1+1) = 0$. If an m-element subset shall be checked for lexicographic minimality in its orbit, then it is more relevant whether the level-1 transitivity gap is zero. Such a zero-transitivity gap is already guaranteed whenever $n-(c_1+1) < m$. This immediately annihilates the 50 % branch weights of all branch-selection vectors starting with a "0" and guarantees that the first reduction factor in the element branch is active for all remaining branches. Similarly, if also the level-2 transitivity gap is zero, then another 25 % of branch weights disappears with an active second reduction factor of the element branch, etc. The reduction factors in the element branch are particularly small if m is small compared to n. The following corollary presents an analysis of the resulting relative effort for level-1 transitivity gaps of zero.

Corollary 4. *Let* ST *be a switch table for a subgroup $\mathfrak{G}$ of $\mathfrak{S}_n$ with maximal effective row r and c_i non-trivial switches in row i, $i = 1,2,\dots,r$ so that $\vartheta_{1,m} = 0$. This is, e.g., the case whenever $\mathfrak{G}$ is transitive. Denote by $r(j)$ the number of effective rows with index strictly larger than j.*

Define $\mu := \frac{n-1}{n}$ and assume that $\delta^2_{m,r} + \mu^2 \leq 1$. Then:

$$\kappa_{mst}(m) \leq \left(1 + \sum_{j=1}^{r} \frac{1}{2^{r(j)}} \cdot \frac{c_j}{c_j+1}\right)\delta^2_{1,m} \leq 3\,\delta^2_{1,m}. \tag{70}$$

Proof. First note that both $\bar{\bar{\delta}}_{i,\ell} \le \mu$ and $\vartheta_{i,\ell} \le \mu$, whence $\alpha^{(0)}_{i,\ell} \le \mu^2$ for all $i = 1,2,\dots,r$ and $\ell = 1,2,\dots,m$. The assumptions and Lemma 6 guarantee that for a branch-selection vector $\mathbf{b} = \binom{1}{\mathbf{b}'}$ with $\mathbf{b}' \in \mathscr{B}(j-1)$ for $j = 1,2,\dots,r-1$ its weight is bounded from above as follows:

$$w_{1,m}(\tbinom{1}{\mathbf{b}'}) \le \delta^2_{1,m} \cdot \left(\delta^2_{r,m}\right)^{\lambda(\mathbf{b}')} \cdot \left(\mu^2\right)^{r-1-\lambda(\mathbf{b}')}. \tag{71}$$

All branch-selection vectors with $b_1 = 0$ have weight zero. Thus, the first sum in Remark 1 can be bounded as follows:

$$\begin{aligned}
&\Big(1+\frac{c_r}{c_r+1}\Big)\sum_{\mathbf{b}\in\mathscr{B}(r)} w_{1,m}(\mathbf{b})v_m(\mathbf{b})k && (72)\\
&\le k\Big(1+\frac{c_r}{c_r+1}\Big)\alpha^{(1)}_{1,m}\sum_{\mathbf{b}'\in\mathscr{B}(r-1)} w_{1,m}(\mathbf{b}')v_m(\mathbf{b}') && (73)\\
&\le km\Big(1+\frac{c_r}{c_r+1}\Big)\delta^2_{1,m}\sum_{\lambda=0}^{r-1}\binom{r-1}{\lambda}\left(\delta^2_{r,m}\right)^{\lambda}\left(\mu^2\right)^{r-1-\lambda} && (74)\\
&= km\left[\Big(1+\frac{c_r}{c_r+1}\Big)\delta^2_{1,m}\left(\delta^2_{r,m}+\mu^2\right)^{r-1}\right] && (75)\\
&\le km\left[\Big(1+\frac{c_r}{c_r+1}\Big)\delta^2_{1,m}\right]. && (76)
\end{aligned}$$

For the second sum, $\frac{k}{k_{[j+1]}} \le \frac{k}{2^{r(j)}}$ for all $j = 1,2,\dots,r-1$, since $c_j + 1 \ge 2$ for all effective rows j. Therefore:

$$\begin{aligned}
&\sum_{j=1}^{r-1}\sum_{\mathbf{b}\in\mathscr{B}(j)} w_{1,m}(\mathbf{b})v_m(\mathbf{b})\frac{c_j}{c_j+1}\cdot\frac{k}{k_{[j+1]}} && (77)\\
&\le k\,\alpha^{(1)}_{1,m}\sum_{j=1}^{r-1}\frac{1}{2^{r(j)}}\cdot\frac{c_j}{c_j+1}\sum_{\mathbf{b}'\in\mathscr{B}(j-1)} w_{1,m}(\mathbf{b}')v_m(\mathbf{b}') && (78)\\
&\le k\,\delta^2_{1,m}\sum_{j=1}^{r-1}\frac{1}{2^{r(j)}}\cdot\frac{c_j}{c_j+1}\sum_{\mathbf{b}'\in\mathscr{B}(j-1)} w_{1,m}(\mathbf{b}')v_m(\mathbf{b}') && (79)\\
&\le km\,\delta^2_{1,m}\sum_{j=1}^{r-1}\frac{1}{2^{r(j)}}\cdot\frac{c_j}{c_j+1}\sum_{\lambda=0}^{j-1}\binom{j-1}{\lambda}\left(\delta^2_{r,m}\right)^{\lambda}\left(\mu^2\right)^{j-1-\lambda} && (80)\\
&= km\,\delta^2_{1,m}\sum_{j=1}^{r-1}\frac{1}{2^{r(j)}}\cdot\frac{c_j}{c_j+1}\left(\delta^2_{r,m}+\mu^2\right)^{j-1} && (81)\\
&= km\,\delta^2_{1,m}\sum_{j=1}^{r-1}\frac{1}{2^{r(j)}}\cdot\frac{c_j}{c_j+1}. && (82)
\end{aligned}$$

In total:

$$\sigma_{\mathrm{mst}}(m) \le km\left[\Big(1+\sum_{j=1}^{r}\frac{1}{2^{r(j)}}\cdot\frac{c_j}{c_j+1}\Big)\delta^2_{1,m}\right]. \tag{83}$$

This implies the assertion. For the simplified, weaker upper bound, note that $\frac{c_j}{c_j+1} \leq 1$ for each effective row j and $\frac{c_j}{c_j+1} = 0$ for each trivial row j. Therefore, for s effective rows a restriction of the sum to effective row indices, reversing the summation, and extension to the infinite series yields the claim as follows:

$$\sum_{j=1}^{r} \frac{1}{2^{r(j)}} \cdot \frac{c_j}{c_j+1} \leq \sum_{j=1}^{s} \frac{1}{2^{s-j}} = \sum_{j=0}^{s-1} \frac{1}{2^j} \leq \sum_{j=0}^{\infty} \frac{1}{2^j} = 2. \tag{84}$$

□

Example 2. Consider the point configuration consisting of the set of vertices of the d-dimensional hypercube $\mathbf{C}_d$. It has $n = 2^d$ points and a transitive symmetry group of order $d!2^d$, generated by all permutations of coordinates and a 0/1-flip of an arbitrary coordinate. If one is interested, e.g., in the circuits (see Section 8) of $\mathbf{C}_d$, then all subsets to be considered contain at least 4 and at most $m = d+2$ points. Consider an order of the points so that, recursively, $\mathbf{C}_d = \begin{pmatrix} \mathbf{C}_{d-1} & \mathbf{C}_{d-1} \\ 0 & 1 \end{pmatrix}$ in non-homogeneous coordinates or with a row of ones at the top in homogeneous coordinates. Then

$$\mathbf{C}_d = \begin{pmatrix} \mathbf{C}_{d-2} & \mathbf{C}_{d-2} & \mathbf{C}_{d-2} & \mathbf{C}_{d-2} \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \tag{85}$$

There is exactly one symmetry that point-wise stabilizes the first 2^{d-2} columns, namely flipping the last two coordinates. Similarly, flipping either of the last two coordinates with the third-to-last coordinate yields exactly two symmetries that stabilize the first 2^{d-3} columns from the left, and so on. For $d = 6$ this yields 1, 2, 3, 4, 5 non-trivial switches in rows 17, 9, 5, 3, 2, respectively, and 63 non-trivial switches in row 1 (because of transitivity). Since $k = 46{,}080 = 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 \cdot 64$, there are no further non-trivial switches. Therefore, $r = 17$. Moreover, for $m = 8$, one obtains $\delta_{1,m} = \frac{8}{64} = \frac{1}{8}$, $\delta_{r,m} = \frac{8}{48} = \frac{1}{6}$, and $\mu = \frac{63}{64}$. Therefore, $\delta_{r,m}^2 + \mu^2 = \frac{1}{6^2} + \frac{63^2}{64^2} \leq 1$. With this, Corollary 4 yields:

$$\begin{aligned} \kappa_{mst}(8) &\leq \left(1 + \frac{1}{1} \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{2}{3} + \frac{1}{4} \cdot \frac{3}{4} + \frac{1}{8} \cdot \frac{4}{5} + \frac{1}{16} \cdot \frac{5}{6} + \frac{1}{32} \cdot \frac{63}{64} \right) \cdot \frac{1}{64} & (86) \\ &< 0.035. & (87) \end{aligned}$$

Hence, the modified switch-table method for the lex-min checks for 8-element subsets in the 6-cube (relevant for circuits) typically needs less than 3.5 % of the effort of the naive method. The simplified upper bound (which does not need the detailed data of the switch table) still yields $\kappa_{mst}(8) \leq \frac{3}{64} < 0.047$. Compare this to the relative effort of the iteration-based critical-element method, which, by Theorem 1, is always $\frac{n+m-1}{nm}$, in this case $\frac{71}{512} > 13.8\,\%$. So, it can be assumed a-priori that for 8-element subsets in the 6-cube the modified switch-table method will be typically at least almost four times as fast (in terms of single-element evaluations) as the critical-element method. The superiority in terms of single-element evaluations of the modified switch-table method in the *enumeration* of circuits in the 6-cube (see Table 1) is much more pronounced: First, Corollary 4 is not very tight in order to arrive at a simpler formula. Second, during the enumeration most checks run on subsets with fewer than 8 elements (some circuits have fewer elements, and the enumeration has to check at least all lex-subsets of circuits), which reduces the speed-up of the critical-element method. Third, in this example $m < r$, so that all branch-selection vectors with more than m one-components lead to summands that are zero (which was ignored in the estimates above, since taking this into account leads to ugly binomial tails). The superiority of the modified switch-table method in terms of CPU times is much less apparent. This can be attributed to the fact that the modified switch-table method incurs a larger effort for the adminstration of local data structures and the recursion compared to the structurally simple, purely iterative critical-element method. In particular, in the critical-element method many single-element evaluations are only read-outs from a consecutive array while in the modified switch-table method many single-element evaluations are used to generate new subsets, incurring a memory management overhead. □

It may seem that the modified switch-table method should be the superior method in most cases. However, in the following it is shown that for large degrees and small orders so that the non-trivial elements in the switch table are extremely sparse the critical-element method can be substantially better.

Corollary 5. *Let* ST *be a switch table for a subgroup* $\mathfrak{G}$ *of* $\mathfrak{S}_n$ *with maximal effective row* r *and* c_i *non-trivial switches in row* i, $i = 1, 2, \dots, r$.

Assume that there is a $\mu \in (0, 1)$ *such that*

- $m \le \mu(n - r + 1)$
- $c_i + 1 \le \mu(n - i + 1 - m + 1)$ *for all* $i = 1, 2, \dots, r$.

Then, the relative effort $\kappa_{mst}(m)$ *of* ST *is at least* $\frac{3}{2}(1-\mu)^{r(m+1)}$*, i.e.,*

$$\sigma_{\mathrm{mst}}(m) \ge \frac{3}{2}(1-\mu)^{r(m+1)} k m. \tag{88}$$

Proof. First, note that $m \le \mu(n - r + 1)$ implies $\bar{\bar{\delta}}_{i,\ell} \ge \bar{\bar{\delta}}_{r,m} \ge (1 - \mu)$. Moreover, $c_i + 1 \le \mu(n - i + 1 - m + 1)$ implies $\vartheta_{i,\ell} \ge \vartheta_{i,m} \ge (1 - \mu)^m$.

The starting point for the lower bound is now again Remark 1, where all branch-selection vectors are ignored except the zero vector. Moreover, since c_i might be zero for $i = 2, 3, \dots, r-1$ only the expressions with branch-selection vectors of length r are taken into account.

With this one obtains:

$$\sigma_{\text{mst}}(m) \geq \Big(1+\frac{c_r}{c_r+1}\Big) w_{1,m}(\mathbf{0}) v_m(\mathbf{0}) \tag{89}$$

$$\geq km\Big(1+\frac{c_r}{c_r+1}\Big)\prod_{t=1}^{r} \alpha_{t,m}^{(0)} \tag{90}$$

$$\geq km\Big(1+\frac{c_r}{c_r+1}\Big)\prod_{t=1}^{r} (\bar{\delta}_{t,m}\,\vartheta_{t,m}) \tag{91}$$

$$\geq km\Big(1+\frac{c_r}{c_r+1}\Big)\Big((1-\mu)(1-\mu)^m\Big)^r \tag{92}$$

$$\geq km\frac{3}{2}\Big((1-\mu)(1-\mu)^m\Big)^r \tag{93}$$

$$\geq \frac{3}{2}(1-\mu)^{r(m+1)} km. \tag{94}$$

□

Thus, for a very small μ it may happen that the exponential speed-up in r of the modified switch-table method is outperformed by the linear speed-up in m of the critical-element method. For the enumeration of triangulations this is the case in all instances computed in this paper. More specifically: The result explains why for triangulations of the 4-cube, the critical-element method is faster, as is shown in the following example.

Example 3. Consider the point configuration consisting of the set of vertices of the d-dimensional hypercube $\mathbf{C}_d$, in particular for $d = 4$. If one is interested, e.g., in triangulations (see Section 9), then the relevant action of its symmetry group of order $k = 384$ is the action on the full-dimensional simplices. Concrete computer calculations show the following. There are 3008 such simplices. Thus, the degree here is $n = 3008$. Note that $384 < 3008$ already implies that the symmetry group is not acting transitively on simplices. Moreover, for the order of points as in Example 2 and the lexicographic order on simplices a switch table has $r = 3$ with $c_1 = 15$, $c_2 = 11$, and $c_3 = c_r = 1$. Finally, triangulations of $\mathbf{C}_4$ have between 16 and 24 simplices. Corollary 5 can, therefore, be applied to this particular switch table, e.g., with $\mu = \frac{1}{100}$. Then, in the best-case $m = 24$ for the modified switch-table method one obtains:

$$\kappa_{mst}(24) \geq \frac{3}{2}\left(\frac{99}{100}\right)^{3\cdot 25} > 70\,\%. \tag{95}$$

In contrast to this, in the worst-case $m = 16$ for the iteration-based critical-element method one obtains:

$$\kappa_{cet}(16) \leq \frac{3008+16-1}{3008\cdot 16} < 7\,\%. \tag{96}$$

Thus, for triangulations one can a-priori guarantee that for the lex-min check on 16- to 24-element subsets of simplices in the 4-cube the critical-element method will typically be at least 10 times as fast (in terms of single-element evaluations) as the modified switch-table method. Again, in Table 1 it can be seen that inside the *enumeration* of triangulations of the 4-cube the critical-element method is only slightly more than 3 times as fast (in terms of single-element evaluations) as the modified switch-table method, because of the many checks of smaller subsets of triangulations. As before, in terms of CPU times the superiority of the critical-element method is more pronounced, probably because of its overall simpler implementation structure. $\square$

So far, a switch table was considered given, and the input subsets were considered random. Now, a switch table on a random k-subset $\mathfrak{G}$ of permutations is discussed. The suitable probability space for this is $(\Omega, 2^\Omega, \mathbb{P}[\cdot])$, where Ω is the set of all k-subsets $P \in \binom{\mathfrak{S}_n}{k}$ of permutations $\pi \in \mathfrak{S}_n$ with $|\Omega| = \binom{n!}{k}$ and $\mathbb{P}[P] = \frac{1}{|\Omega|}$ for all $P \in \binom{\mathfrak{S}_n}{k}$.

Motivated by the applications in this paper (see the examples in Sections 7 through 9), the focus in the following is on orders k that are much smaller than the number $n!$ of all permutations in $\mathfrak{S}_n$. For these cases, a binomial approximation is added based on sampling k permutations uniformly and independently at random with replacement to generate a sequence of k (not necessarily distinct) permutations, which simplifies formulas.

Theorem 3. *Let $\mathfrak{G}$ be a k-subset of permutations in $\mathfrak{S}_n$ that has been chosen uniformly at random from all such k-subsets. Then, the typical number c_i of non-trivial switches in row i for $i = 1, 2, \ldots, n$ is given by*

$$\mathbb{E}[c_i] = (n-i)\left(1 - \prod_{t=0}^{k-1}\left(1 - \frac{(n-i)!}{n! - t}\right)\right) \tag{97}$$

If k is sufficiently smaller than $n!$, then a binomial approximation of this is given as follows: Let $\mathfrak{G}$ be a random sequence of k permutations in $\mathfrak{S}_n$, where each permutation has been chosen uniformly and independently at random with replacement. Then, the typical number c_i of non-trivial switches in row i for $i = 1, 2, \ldots, n$ is given by

$$\mathbb{E}[c_i] = (n-i)\left(1 - \left(1 - \frac{1}{(n)_i}\right)^k\right). \tag{98}$$

Proof. Call $A_i(j)$ the set of all permutations $\pi \in \mathfrak{S}_n$ that stabilize $1, 2, \ldots, i-1$ and map a given $j > i$ to i. Since for permutations in $A_i(j)$ exactly i images are fixed and the others are arbitrary, there are $(n-i)!$ permutations in $A_i(j)$. The number of k-subsets of permutations that contain no permutation in $A_i(j)$ is, therefore, $\binom{n!-(n-i)!}{k}$. Let $\mathscr{C}_i(j)$ be the event containing those k-subsets of permutations for which row i and column j in a switch table of $\mathfrak{G}$ is non-trivial. The cardinality of $\mathscr{C}_i(j)$ is $\binom{n!}{k} - \binom{n!-(n-i)!}{k}$. Hence, the probability $\mathbb{P}[\mathscr{C}_i(j)]$ that row i and column j of a switch table for a random k-subset of permutations is non-trivial can be computed as follows.

$$\mathbb{P}[\mathscr{C}_i(j)] = \frac{\binom{n!}{k} - \binom{n!-(n-i)!}{k}}{\binom{n!}{k}} \tag{99}$$

$$= 1 - \frac{\binom{n!-(n-i)!}{k}}{\binom{n!}{k}} \tag{100}$$

$$= 1 - \frac{\big(n! - (n-i)!\big)_k}{(n!)_k} \tag{101}$$

$$= 1 - \prod_{t=0}^{k-1} \frac{n! - (n-i)! - t}{n! - t} \tag{102}$$

$$= 1 - \prod_{t=0}^{k-1} \frac{(n! - t) - (n-i)!}{n! - t} \tag{103}$$

$$= 1 - \prod_{t=0}^{k-1} \left(1 - \frac{(n-i)!}{n! - t}\right) \tag{104}$$

$$\tag{105}$$

Using the indicator variable $\mathbf{1}_{\mathscr{C}_i(j)}$ of $\mathscr{C}_i(j)$ and the linearity of expectation, the typical number of non-trivial switches $\mathbb{E}[c_i]$ in row i of a switch-table for $\mathfrak{G}$ is given by

$$\mathbb{E}[c_i] = \sum_{j=i+1}^{n} \mathbb{E}[\mathbf{1}_{\mathscr{C}_i(j)}] = \sum_{j=i+1}^{n} \mathbb{P}[\mathscr{C}_i(j)] = (n-i)\left(1 - \prod_{t=0}^{k-1}\left(1 - \frac{(n-i)!}{n! - t}\right)\right). \tag{106}$$

This equals the first assertion. If k is sufficiently smaller than $n!$, then the k factors in the product are all approximately equal to $1 - \frac{(n-i)!}{n!} = 1 - \frac{1}{(n)_i}$, which yields the assertion about the binomial approximation. □

The formula shows that for small orders k and large degrees n there are typically only very few non-trivial switches in rows $i > 1$, which leads to large level-i transitivity gaps and, therefore, a smaller performance gain of a switch table.

For the following bound on the maximal effective row the considerations are based on the binomial approximation. It shows that in the absence of special structure the maximal effective row r is usually small when the degree n is large compared to the order k of the symmetry group. Of course, in such cases k is, in particular, much smaller than $n!$, so that a binomial approximation is justified. Recall that for a small r the linear overhead of a switch table may outweigh the exponential gain; therefore, this is an important finding.

Theorem 4. *The typical maximal effective row r of a switch table for a sequence of k permutations of degree n chosen uniformly and independently at random with replacement is at most $1 + \frac{2k}{n}$. In particular, whenever the degree is at least twice the order, then the typical maximal effective row r is at most two.*

Note that a group may have special structure leading to more effective rows in a switch table. The claim is about the hyper-amortized average for $k \ll n!$ only and provides a hint what to expect.

Proof. Call a random permutation *i-non-trivial* for $i = 2, \ldots, n$ if it could be chosen as a non-trivial switch in row i or larger of a switch table. Note that the probability $\mathbb{P}[\pi\ i\text{-non-trivial}]$ is given by the probability that π stabilizes $1, 2, \ldots, i-1$, which yields

$$\mathbb{P}[\pi\ i\text{-non-trivial}] = \frac{1}{n(n-1)\cdots(n-i+2)}. \tag{107}$$

Moreover, the number b_i of i-non-trivial permutations is typically $\mathbb{E}[b_i] = k\mathbb{P}[\pi\ i\text{-non-trivial}]$ (approximation by the expectation in a binomial distribution). The maximal effective row r is at least i if and only if there is at least one i-non-trivial permutation. Consequently:

$$\mathbb{E}[b_i] = \sum_{j=0}^{k} j\mathbb{P}[b_i = j] \geq \sum_{j=1}^{k} \mathbb{P}[b_i = j] = \mathbb{P}[b_i \geq 1] = \mathbb{P}[r \geq i]. \tag{108}$$

This leads to the following estimate:

$$\begin{aligned}
\mathbb{E}[r] &= \sum_{i=1}^{n} i\mathbb{P}[r = i] && (109)\\
&= \sum_{i=1}^{n} \mathbb{P}[r \geq i] && (110)\\
&= 1 + \sum_{i=2}^{n} \mathbb{P}[r \geq i] && (111)\\
&\leq 1 + \sum_{i=2}^{n} \mathbb{E}[b_i] && (112)\\
&= 1 + \sum_{i=2}^{n} k\mathbb{P}[\pi\ i\text{-non-trivial}] && (113)\\
&= 1 + k\sum_{i=2}^{n} \frac{1}{n(n-1)\cdots(n-i+2)} && (114)\\
&= 1 + \frac{k}{n} + \frac{k}{n}\sum_{i=3}^{n} \frac{1}{(n-1)\cdots(n-i+2)} && (115)\\
&\leq 1 + \frac{k}{n}\sum_{i=2}^{n} \left(\frac{1}{2}\right)^{i-2} && (116)\\
&= 1 + \frac{k}{n}\sum_{i=0}^{n-2} \left(\frac{1}{2}\right)^{i} && (117)\\
&\leq 1 + \frac{2k}{n}. && (118)
\end{aligned}$$

For the special case $n \geq 2k$, consequently, $\mathbb{E}[r] \leq 2$. □

The key learning of the previous theorems beyond the complicated formulas is that the typical efficiency of the modified switch-table method becomes better (ceteris paribus) as the order increases compared to the degree, and vice versa.

The analysis is qualitatively confirmed by the computational experiments for the applications in this paper. Table 1 shows for the circuits of the 6-cube $\mathbf{C}_6$ and the triangulations of the 4-cube $\mathbf{C}_4$ the qualitative consistency of Corollaries 4 and 5 with computational experience single-threaded on an M1Max machine (for details on the computational environment see Section 6).

comput.	n	k	r	m	CPU time [s]			# realized single-element evaluations		
					cei	ces	mst	cei	ces	mst
$\mathbf{C}_6$ circuits	64	46,080	17	4–8	14	7	3	4,259,939,467	498,266,262	50,735,629
$\mathbf{C}_4$ triang's	3008	384	3	16–24	5	-	32	796,884,547	-	2,673,110,644

Table 1: Spot-light comparison of methods for lex-min checks cei = "critical-element method via iteration", ces = "critical-element method via sets", and mst = "modified switch-table method" (single-threaded)

Remark 2. The analysis in this section neglects the necessary memory-management overhead for non-trivial temporary local data-structures for subsets. In `TOPCOM`, e.g., this overhead – which in naive implementations can dominate anything else – is more prevalent in the modified switch-table method than in the critical-element method.

4.3 Parallelization

The enumeration of trees is much easier to parallelize than general enumeration, since the enumerations of distinct subtrees do not influence each other.

How to explore reverse-search trees in parallel using multiple processes with only little inter-process communication was proposed in [3]. The core idea is *budgeted load balancing*. That means, a coordinating process assigns to each worker process the root of a subtree and a limited budget of nodes to enumerate in that subtree. The enumeration results in that subtree roots are stored with the worker process. After the completion of its budget, the worker process returns the control to the coordinating process, which in turn merges the results from the worker process, including any unprocessed nodes in the subtree, to the results obtained so far. The coordinating process then assigns to each idle worker process a new subtree root and a new budget. Note that, because of independent enumerations in each subtree, there can be no guarantee anymore for the order in which the objects of interest are output. However, the "is-canonical" check implemented via the lex-min check in `SymLexSubsetRS` remains correct since lex-minimality and, thus, canonicalicity does not depend on when a subset orbit was found for the first time.

The big advantage of tree-based parallelization is that such a parallelization scheme can carry out the actual enumeration and the merge of each worker's results completely lock-free. (The experience with various load-balancing mechanisms shows that a lock-free implementation is the single most important success factor for fast parallelizations of enumeration algorithms like the ones in this paper.)

`TOPCOM` uses a modified version of this paradigm using a lean shared memory for multiple threads in one process instead of multiple processes: the coordinating thread updates a count of currently unassigned subtree roots. Each worker thread reads out this count after each discovery of a new node. If the count falls below the number of worker threads (a threshold that can be configured), the worker thread stops and returns its results – including the unprocessed nodes in the subtree – as well as the control to the coordinating thread. The read-out of the count can be done lock-free because it does not matter at which node exactly the worker thread interrupts.

In TOPCOM the new method is called *workbuffer control*. It was used throughout the computational experiments. The advantage of workbuffer control compared to budgeted load balancing is that the number of unprocessed subtree roots (which must be stored in the coordinating process) turns out to be much smaller for large problems. Moreover, the worker processes usually stop less often unnecessarily this way. The cpu times of budgeted load balancing versus workbuffer control did not show any significant differences in the examples of this paper.

5 Applications: Common Preliminaries

In this section, some basic notions and notation are summarized for the common setup of the applications presented in the upcoming sections.

Consider a point or vector configuration of rank r with n points or vectors. It is represented by an $r \times n$-matrix $\mathbf{A}$ containing the coordinates $\mathbf{a}_1, \dots, \mathbf{a}_n \in \mathbb{R}^r$ of the points (as homogeneous coordinates) or vectors as columns. The number $c := n - r$ is the *corank* of $\mathbf{A}$. For a subset S of column indices of $\mathbf{A}$ denote by $\mathbf{A}_{*,S}$ the submatrix consisting of the columns with increasing indices in S. Moreover, for a matrix $\mathbf{M}$ in column-echelon form denote by $\mathbf{M}_{\mathrm{NZ}}$ its submatrix of non-zero columns.

The usual language of linear algebra can be adapted to subsets of indices as follows.

Definition 9. A subset $B \in [n]$ is *spanning* if $\operatorname{rank}(\mathbf{A}_{*,B}) = r$, otherwise it is *not-spanning* or *coplanar*. A coplanar subset B with $\operatorname{rank} B = r - 1$ is *hyperplanar*. It is *independent* if $\operatorname{rank}(\mathbf{A}_{*,B}) = |B|$, otherwise it is *dependent*. A *basis* is an independent spanning subset. A *simplex* (in the context of triangulations) is a basis S such that $\operatorname{cone}(\mathbf{A}_{*,S})$ is a pointed polyhedral cone, i.e., the origin is a vertex of it. An $r-1$-subset of a simplex is a *simplex facet* or simply a *facet* whenever no confusion with facets of $\mathbf{A}$ can arise.

The first application deals with cocircuits, which can be seen as hyperplanes spanned by the configuration.

Definition 10. Any inclusion-maximal coplanar subset C_0^* of column indices of the configuration $\mathbf{A}$ is called *(the zero part of) a cocircuit*. A *cocircuit signature* of C_0^* is a map $\sigma^* \colon [n] \to \{-, 0, +\}$ with $C_0^* = \sigma^{-1}(\{0\})$ so that there is a $\mathbf{c} \in \mathbb{R}^r$ with $\mathbf{c}^T \mathbf{a}_i = 0$ for all $i \in (\sigma^*)^{-1}(\{0\})$, $\mathbf{c}^T \mathbf{a}_i > 0$ for all $i \in (\sigma^*)^{-1}(\{+\})$, and $\mathbf{c}^T \mathbf{a}_i < 0$ for all $i \in (\sigma^*)^{-1}(\{-\})$. Here, $C_+^* := (\sigma^*)^{-1}(\{+\})$ is called the *positive part*, and $C_-^* := (\sigma^*)^{-1}(\{-\})$ is called the *negative part* of the cocircuit signature σ^*. By elementary linear algebra, there are exactly two *opposite* cocircuit signatures of C_0^*. Moreover, these two signatures are uniquely determined by any hyperplanar subset of C_0^*. The pair (C_+^*, C_-^*) is a *signed cocircuit*.

Intuitively, a cocircuit is the subset of *all* elements lying on a hyperplane spanned by *some* of the elements in $\mathbf{A}$. Counting all cocircuits is, therefore, the same as counting all hyperplanes spanned by elements of the configuration.

Another application is concerned with circuits, which can be seen as unique intersection points of subpolytopes spanned by the configuration.

Definition 11. Any inclusion-minimal dependent subset C of the column indices of $\mathbf{A}$ is *(the support of) a circuit.* A *circuit signature* of a circuit C is a map $\sigma\colon [n] \to \{-,0,+\}$ so that $C = \sigma^{-1}(\{-,+\})$ and $\sum_{i\in\sigma^{-1}(\{+\})} \lambda_i \mathbf{a}_i = \sum_{i\in\sigma^{-1}(\{-\})} \lambda_i \mathbf{a}_i$ for suitable $\lambda_i > 0$, $i = 1,2,\dots,n$. Here, $C_+ := \sigma^{-1}(\{+\})$ is called the *positive part,* $C_- := \sigma^{-1}(\{-\})$ the *negative part,* and $C_0 := \sigma^{-1}(\{0\})$ the *zero-part* of the circuit signature σ. By elementary linear algebra, there are exactly two *opposite* circuit signatures of C. The pair (C_+, C_-) is a *signed circuit.*

Symmetries of a configuration are the (co-)circuit-maintaining permutations of elements.

Definition 12. A permutation $\pi \in \mathfrak{S}_n$ is a *(combinatorial) symmetry of* $\mathbf{A}$, if the following holds for each pair (S,R) of subsets of $[n]$: (S,R) is a signed cocircuit of $\mathbf{A}$ if and only if $\big(\pi(S),\pi(R)\big)$ is a signed cocircuit of $\mathbf{A}$, or, equivalently, (S,R) is a signed circuit of $\mathbf{A}$ if and only if $\big(\pi(S),\pi(R)\big)$ is a signed circuit of $\mathbf{A}$. A symmetry is *affine* if it is induced by an affine isomorphism $f : \operatorname{aff}(\mathbf{A}) \to \operatorname{aff}(\mathbf{A})$ via $\pi(i) = j$ if and only if $f(\mathbf{a}_i) = \mathbf{a}_j$. The *(combinatorial) symmetry group* $\operatorname{Aut}(\mathbf{A}) = \operatorname{Aut}^{\text{comb}}(\mathbf{A})$ *of* $\mathbf{A}$ is the group of all (combinatorial) symmetries of $\mathbf{A}$. The *affine symmetry group* $\operatorname{Aut}^{\text{aff}}(\mathbf{A})$ is the group of all affine symmetries.

By this definition, it makes sense to enumerate all (co-)circuits of $\mathbf{A}$ up to (combinatorial) symmetry.

The third application in this paper deals with triangulations of $\mathbf{A}$. It is advantageous to use a well-known characterization of triangulations of point configurations as the definition [8, Cor. 4.1.32]. In the context of triangulations, bases are usually called *simplices.*

Definition 13. Two simplices S_1 and S_2 are *intersecting properly* if there is no signed circuit C with $C_+ \subseteq S_1$ and $C_- \subseteq S_2$. A simplex facet F is *interior* if there is a cocircuit C^*_0 with $F \subseteq C^*_0$ so that C^*_+ and C^*_- are both non-empty. A non-empty subset $\mathscr{T}$ of simplices is *covering* if for each interior facet F of some simplex $S \in \mathscr{T}$ there is another simplex $S' \in \mathscr{T}$ containing F. A *triangulation* is a non-empty covering subset $\mathscr{T}$ of pairwise properly intersecting simplices.

Proper intersection of two simplices S_1 and S_2 roughly means geometrically that the convex hulls $\operatorname{conv}\mathbf{A}_{*,S_1}$ and $\operatorname{conv}\mathbf{A}_{*,S_2}$ intersect in a common (possibly empty) face of both. The covering property together with proper intersection roughly means geometrically that all facets of a simplex in $\mathscr{T}$ interior in $\mathbf{A}$ are covered by simplices from boths sides in $\mathscr{T}$.

Whether or not a subset of simplices of $\mathbf{A}$ is a triangulation depends only on the combinatorics of $\mathbf{A}$ as given by its set of (co-)circuits [8, Thm. 4.1.31]. Therefore:

Lemma 7. *For any combinatorial symmetry* π *of* $\mathbf{A}$ *and any subset* T *of* $[n]$ *one has that* T *indexes a triangulation of* $\mathbf{A}$ *if and only if the set* $\pi(T)$ *indexes a triangulation of* $\mathbf{A}$. □

By this lemma, it makes sense to enumerate all triangulations of $\mathbf{A}$ up to (combinatorial) symmetry. Note that additional restrictions on the triangulations to be enumerated (like regularity or unimodularity) can break this property. In that case, the symmetries have to be restricted to such symmetries that maintain the required structures imposed on the triangulations to be enumerated. To enumerate regular triangulations up to symmetry the symmetries have to be restricted to affine symmetries (maintaining the convexity of liftings), and for the enumeration of unimodular triangulations the symmetries have to be restricted to isometric symmetries (maintaining the volumes of simplices). In Section 9.4 further examples for interesting restrictions can be found that require a restriction of the combinatorial symmetries of **A** to a smaller subgroup.

A	dimension	# points	# symmetries
$\mathbf{C}_5$	5	32	38,040
$\mathbf{C}_6$	6	64	46,080
$\mathbf{C}_7$	7	128	645,120
$\mathbf{C}_8$	8	256	10,321,920
$\mathbf{C}_9$	9	512	185,794,560
$\boldsymbol{\Delta}(8,3)$	7	56	40,320
$\boldsymbol{\Delta}(8,4)$	7	70	40,320
$\boldsymbol{\Delta}(9,3)$	8	84	362,880
$\boldsymbol{\Delta}(9,4)$	8	126	362,880
$\boldsymbol{\Delta}(10,3)$	9	120	3,628,800
$\boldsymbol{\Delta}(10,4)$	9	210	3,628,800
$\boldsymbol{\Delta}(10,5)$	9	252	3,628,800

Table 2: Parameters of the configurations for the enumeration of (co-)circuits

In the computational results, the following point configurations are mentioned. Since some results depend on the order of points, we also mention the order used in this paper. The homogenization of coordinates is not explicitly mentioned in the following:

- The d-dimensional point configuration $\mathbf{C}_d$ is the d-dimensional *hypercube*. The 2^d points are ordered in this paper such that $\mathbf{C}_d = \begin{pmatrix} \mathbf{C}_{d-1} & \mathbf{C}_{d-1} \\ 0 & 1 \end{pmatrix}$ with $\mathbf{C}_0 := ()$ (which can be interpreted as the only point in $\mathbb{R}^0$).

- The $n+n'$-dimensional point configuration $\boldsymbol{\Delta}_n \times \boldsymbol{\Delta}_{n'}$ is the *product of simplices* of dimensions n and n'. The $(n+1)(n'+1)$ points are ordered in this paper such that $\boldsymbol{\Delta}_n \times \boldsymbol{\Delta}_{n'} = \begin{pmatrix} I_{n+1} & I_{n+1} & \cdots & I_{n+1} \\ e_1 & e_2 & \cdots & e_{n'+1} \end{pmatrix}$.

- The d-dimensional point configuration $\boldsymbol{\Delta}(d,k)$ is the d-dimensional *hypersimplex* with coordinate sum k. The points are ordered lexicographically in this paper.

- The d-dimensional point configuration $k\boldsymbol{\Delta}_d$ is the *dilated simplex* in dimension d with *dilation factor* k; it may have interior points. The points are ordered lexicographically in this paper. The number of subregular/regular/unimodular triangulations of $3\boldsymbol{\Delta}_3$ were computed for the first time in [14] with the help of high-performance computing with `mptopcom` – the largest number of triangulations computed at that time.

- The d-dimensional point configuration $\mathbf{C}(n,d)$ is the d-dimensional *cyclic polytope* with n vertices. The points are ordered in this paper with increasing first coordinate.

- The 3-dimensional point configurations *icosahedron* and *dodecahedron* consist of the vertices of the regular polytopes corresponding to the respective Platonic solids with

$\mathbf{A}$	dimension	# points	# symmetries
$\mathbf{C}_4$	4	16	384
$\boldsymbol{\Delta}(6,3)$	5	20	1440
$\boldsymbol{\Delta}(7,2)$	6	21	5040
$\boldsymbol{\Delta}_6 \times \boldsymbol{\Delta}_2$	8	21	30,240
$\boldsymbol{\Delta}_4 \times \boldsymbol{\Delta}_3$	7	20	2880
$\boldsymbol{\Delta}_5 \times \boldsymbol{\Delta}_3$	8	24	17,280
$\boldsymbol{\Delta}_4 \times \boldsymbol{\Delta}_4$	8	25	28,800
$3\boldsymbol{\Delta}_3$	3	20	24
$4\boldsymbol{\Delta}_3$	3	35	24
$3\boldsymbol{\Delta}_4$	4	35	120
icosahedron	3	12	120
pseudo-icosahedron	3	12	24
dodecahedron	3	20	120
pyritohedron	3	20	24
$\mathbf{P}(A_2)$	2	7	12
$\mathbf{P}(A_3)$	3	13	48
$\mathbf{P}(A_4)$	4	21	240
$\mathbf{P}(A_5)$	5	31	1440
$\mathbf{C}(n,2k)$	$2k$	n	$2n$
$\mathbf{C}(n,2k-1)$	$2k-1$	n	2

Table 3: Parameters of the configurations for the enumeration of triangulations

12 and 20 vertices, respectively. There are no rational coordinates for them, but all their (co-)circuits can still be computed exactly using algebraic field extensions. There are no results in this paper that depend on the order of points.

- The 3-dimensional point configuration *pseudo-icosahedron* consists of the vertices of an approximation of the regular icosahedron by rational coordinates generated by ratios of Fibonacci-numbers approximating the golden ratio. In this paper, the exact coordinates are chosen and ordered as follows:

$$\begin{pmatrix} 0 & 0 & 21 & -21 & 13 & -13 & 13 & -13 & 21 & -21 & 0 & 0 \\ 21 & 21 & 13 & 13 & 0 & 0 & 0 & 0 & -13 & -13 & -21 & -21 \\ 13 & -13 & 0 & 0 & 21 & 21 & -21 & -21 & 0 & 0 & 13 & -13 \end{pmatrix}$$

- The 3-dimensional point configuration *pyritohedron* consists of the vertices of a *pseudo-dodecahedron* in the same sense known from cristallography. In this paper, the exact coordinates are chosen and ordered as follows:

$$\begin{pmatrix} -1 & 1 & -1 & -1 & 0 & -3/4 & -3/2 & 0 & -3/4 & 3/2 & -3/2 & 3/4 & 0 & 3/2 & 3/4 & 0 & 1 & 1 & -1 & 1 \\ -1 & -1 & -1 & 1 & -3/4 & -3/2 & 0 & -3/4 & 3/2 & 0 & 0 & -3/2 & 3/4 & 0 & 3/2 & 3/4 & -1 & 1 & 1 & 1 \\ -1 & -1 & 1 & -1 & -3/2 & 0 & -3/4 & 3/2 & 0 & -3/4 & 3/4 & 0 & -3/2 & 3/4 & 0 & 3/2 & 1 & -1 & 1 & 1 \end{pmatrix}$$

- The *n-dimensional full root polytope* consists of the vertices of the n-polytope $\mathbf{P}(A_n) = \operatorname{conv}\{e_j - e_i : 1 \leq i, j \leq n+1\} \subset \mathbb{R}^{n+1}$ in ambient $n+1$-space together with the origin.

The points are ordered in this paper as $(e_j - e_i, e_i - e_j)$ for (i, j) ordered lexicographically and transformed to full rank by Gaussian elimination. Of particular interest are the numbers of *central* triangulations of full root polytopes, i.e., the ones in which each simplex contains the origin. For details about the enumeration of triangulations, see Section 9. In [16] these numbers are computed up to $\mathbf{P}(A_4)$ to classify the combinatorial types of *polytropes*. This classification was achieved earlier in [31] via Gröbner fans. Among the central triangulations of full root polytopes, the centrally symmetric ones have attracted recent attention as well. No results have been published so far about $\mathbf{P}(A_5)$. The new TOPCOM enumeration results for the number of central and centrally symmetric triangulations of $\mathbf{P}(A_5)$ presented in Section 9.4 have meanwhile been utilized for the classification of certain finite metrics in [9], where also the relevance of these numbers is explained.

6 Applications: Computational Environment

For all computational tests a C++-implementation in `TOPCOM` beta-version `1.2.0b` was used. See [25] for a paper on an earlier version of `TOPCOM`.

The computer is an Intel(R) Xeon(R) CPU E5-2690, 2.90GHz (384 GB RAM) with 2 sockets of 8 cores each and two virtual threads per core. The operating system was `Ubuntu Linux 5.4.0-172-generic` with the C++compiler from `gcc version 9.4.0 (Ubuntu 9.4.0-1ubuntu1 20.04.2)` Unless state otherwise, for each run 16 threads were used on an otherwise idle machine since hyper-threading has no advantages for computationally demanding tasks.

For some computations, also an Apple MacBookPro (2021) was used with M1Max (64 GB RAM) with 8 performance cores and 2 efficiency cores. The operating systems ranged over the time of the experiments from `MacOSX Sonoma 14.5 (23F79)` with the C++-compiler `Apple clang version 15.0.0 (clang-1500.3.9.4)` through `MacOSX Sequoia 15.6.1` with the C++-compiler `Apple clang version 17.0.0 (clang-1700.4.4.1)`. Unless stated otherwise, this computer was run with 8 threads. Computational results achieved with this computer are marked with “M1Max”.

No multiple runs were executed for the experiments in this paper, since for the larger instances this simply would have taken too much time. Therefore, the cpu times could vary a little. Since the cpu times in this paper are several times faster than for any earlier effort, this does not affect the relevance of the results.

7 Application I: Cocircuits

In this section it is shown how all cocircuits of a point configuration **A** can be enumerated up to symmetry by exploring the downset $\mathscr{D}$ of all coplanar subsets using `SymLexSubsetRSMax_withData` (Algorithm 8).

7.1 Specializations

In order to make the application of SymLexSubsetRSMax_withData explicit, its subroutines IsInDownset, SemiIsNotRightComp, IsInEachMax, and IsMaxInDownset must be formally specified. Recall that in these subroutines one can utilize global data (preprocessed prior to the enumeration) and local data (updated in each enumeration node) to speed up the computations and avoid duplicate work.

In this particular case, the local data **L** consists of a matrix M that is a column-echelon form of $\mathbf{A}_S$. It can be computed by Gaussian elimination on columns from left to right. Storing it allows us to avoid the repetition of identical eliminations in matrices with identical initial segments of columns. The local data of the node of a subset S' with $S = S' \setminus \max S'$ is initialized as the augmented matrix $(\mathsf{M}, \mathbf{A}_{*,\max S'})$. The global data used is the full matrix **A**.

```
Algorithm: IsInDownset($S'$, $\mathscr{D}$, $\mathbf{A}$, $\mathsf{M}$, $\mathsf{M}'$)
Input: A subset $S'$ of column indices of A, the downset $\mathscr{D}$ of coplanar subsets, the
        configuration A, a column-echelon form M of $\mathbf{A}_{*,S'\setminus\max S'}$, and the augmented matrix
        $\mathsf{M}' = (\mathsf{M}, \mathbf{A}_{*,\max S'})$ (not yet in column-echelon form, in general)
Output: (TRUE, M') with M' a column-echelon form of $\mathbf{A}_{*,S'}$ if $\mathrm{rank}(\mathsf{M}') \le r-1$ and (FALSE, M')
        otherwise
M' ← colechForm(M') ;                                  /* compute column-echelon form */
if $|S'| < r$ or $\mathsf{M}'_{*,r} = \mathbf{0}$ then                 /* if at most $r-1$ non-zero columns */
    return (TRUE, M') ;                                              /* $S$ is coplanar */
else
    return (FALSE, M') ;                                             /* $S$ is spanning */
```

Algorithm 14: Check whether a subset of columns is coplanar

The implementation of SemiIsNotRightComp in Algorithm 15 is based on the following theorem.

Theorem 5. *Let S be right-completable to (the zero-set of) a cocircuit C_0^* of* **A**. *Then:*

(i) For the set $R := \{i \in [n] : i > \max S\}$ the rank bound $\mathrm{rank}(\mathbf{A}_{*,S\cup R}) \ge r-1$ holds.

(ii) For all $i \in [n] \setminus S$ with $i < \max S$ the rank bound $\mathrm{rank}(\mathbf{A}_{*,S}) < \mathrm{rank}(\mathbf{A}_{*,S\cup\{i\}})$ holds.

Proof. For item (i) assume that S is right-completable to a maximal and, thus, maximally hyperplanar subset S' with $\mathrm{rank}(\mathbf{A}_{*,S\cup R}) < r-1$. Then, since $S' \setminus S \subseteq R$, we have $\mathrm{rank}(\mathbf{A}_{*,S'}) < r-1$, contradicting the fact that S' is hyperplanar.

For item (ii) assume there is an $i \in [n] \setminus S$ with $i < \max S$ so that $\mathrm{rank}(\mathbf{A}_{*,S}) = \mathrm{rank}(\mathbf{A}_{*,S\cup\{i\}})$, and assume there is a maximal hyperplanar subset S' that is a right-completion of S. Then, i is not in S', and $\mathrm{rank}(\mathbf{A}_{*,S'\cup\{i\}}) = \mathrm{rank}\big(\mathbf{A}_{*,(S'\setminus S)\cup(S\cup\{i\})}\big) = \mathrm{rank}\big(\mathbf{A}_{*,(S'\setminus S)\cup S}\big) = \mathrm{rank}(\mathbf{A}_{*,S'})$, contradicting the maximality of S'. □

Call the use of the semi-check based on this theorem *rank-pruning*; skipping this semi-check is called *no-pruning* for later reference. Moreover, call the set $\mathscr{D}^{\text{nonprunable}}$ of subsets

```
Algorithm: SemiIsNotRightComp($S', \mathscr{D}, \mathbf{A}, \mathsf{M}, \mathsf{M}'$)
Input: A coplanar subset $S'$ of column indices of $\mathbf{A}$, the downset $\mathscr{D}$ of coplanar subsets, the
        configuration $\mathbf{A}$, a column-echelon form $\mathsf{M}$ of $\mathbf{A}_{*,S'\setminus\max S'}$, and a column-echelon form
        $\mathsf{M}'$ of the augmented matrix $\mathbf{A}_{*,S'}$
Output: $(\mathsf{TRUE}, \mathsf{M}')$ if $S'$ cannot be right-completed and $(\mathsf{FALSE}, \mathsf{M}')$ otherwise
$r' \leftarrow \mathrm{rank}(\mathsf{M}')$ ;                                  /* get current rank */
for $i = 1, \dots, \max S' - 1$ with $i \notin S'$ do             /* traverse left non-elements */
    $\mathsf{M}'' \leftarrow$ colechForm($\mathsf{M}', \mathbf{a}_i$) ;                  /* compute column-echelon form */
    if $\mathsf{M}''_{*,r'+1} = \mathbf{0}$ then                /* if column $i$ is in the current span */
        return $(\mathsf{TRUE}, \mathsf{M}')$ ;                            /* $S'$ not right-completable */

if $r' < r - 1$ then                                   /* if rank not yet sufficient */
    $\mathsf{M}'' \leftarrow \mathsf{M}'$ ;                      /* prepare a rank-increase checker matrix */
    $r'' \leftarrow r'$ ;                                    /* keep track of rank increase */
    for $i = \max S' + 1, \dots, n$ do                     /* traverse right non-elements */
        $\mathsf{M}'' \leftarrow$ colechForm($\mathsf{M}'', \mathbf{a}_i$) ;               /* compute column-echelon form */
        if $\mathsf{M}''_{*,r''+1} \neq \mathbf{0}$ then                              /* if $i$ increases rank */
            $r'' \leftarrow r'' + 1$ ;                                  /* update rank increase */
            if $r'' = r - 1$ then                          /* if rank increase sufficient */
                return $(\mathsf{FALSE}, \mathsf{M}')$ ;                /* $S'$ might be right-completable */

        else if $r'' + n - i < r - 1$ then                  /* if target rank unreachable */
            return $(\mathsf{TRUE}, \mathsf{M}')$ ;                         /* $S'$ not right-completable */

return $(\mathsf{FALSE}, \mathsf{M}')$ ;                          /* $S'$ might be right-completable */
```

Algorithm 15: The rank-pruning semi-check whether a coplanar subset of columns is right-completable based on a direct application of Theorem 5

in $\mathscr{D}$ for which rank-pruning returns “FALSE” the *non-prunable* subsets. In Table 4 the node counts for no-pruning and rank-pruning are compared for tiny to small examples.

In order to find out whether an expansion is in each maximal superset, one can use a similar observation: any element that does not increase the rank of the current subset can be added to each rank-$(r-1)$ superset thereof without increasing the rank. Thus, it is in each maximal superset. The pseudo-code is listed in Algorithm 16. All data necessary for this check have been computed before. Thus, this check is very fast, and there is no point in skipping it.

In order to implement the maximality check one can utilize a signature corresponding to a hyperplanar subset, which must be computed anyway. This can be done as follows.

Lemma 8. *Let S be a hyperplanar subset and let $\mathbf{B}$ be the $r \times (r-1)$-matrix of non-zero-columns of a column-echelon form of $\mathbf{A}_{*,S}$. Then, there is a unique cocircuit C_0^* containing S. Moreover, one of the two opposite signatures of C_0^* is given by*

$$\sigma_S^*(i) = \mathrm{sign}\big(\det(\mathbf{B}, \mathbf{a}_i)\big). \tag{119}$$

$\mathbf{A}$	# cocircuits	# nodes by pruning method		CPU time [s] by pruning method	
	in total	no	rank	no	rank
$\mathbf{C}_5$	3254	1,026,636	78,050	13.62	3.37
$\mathbf{\Delta}_4\times\mathbf{\Delta}_4$	460	2,018,570	87,440	37.52	5.36
$\mathbf{\Delta}(8,2)$	1661	3,133,114	131,007	50.88	6.60

Table 4: Comparison of no-pruning versus rank-pruning on tiny to small examples (single-threaded, symmetries ignored)

```
Algorithm: IsInEachMax($S', \mathscr{D}, \mathbf{A}, \mathsf{M}, \mathsf{M}'$)
Input: A subset $S'$ of column indices of $\mathbf{A}$, the downset $\mathscr{D}$ of coplanar subsets, the
       configuration $\mathbf{A}$, a column-echelon form M of $\mathbf{A}_{*,S'\setminus\max S'}$, and a column-echelon form
       $\mathsf{M}'$ of the augmented matrix $\mathbf{A}_{*,S'}$
Output: (TRUE, $\mathsf{M}'$) if each maximal superset of $S$ contains $S'$ and (FALSE, $\mathsf{M}'$) otherwise
if rank($\mathsf{M}'$) = rank(M) then                    /* $S'$ did not increase rank */
    return (TRUE, $\mathsf{M}'$) ;
else
    return (FALSE, $\mathsf{M}'$) ;
```

Algorithm 16: Check whether the expansion from S to S' is in each maximal superset of S; the local data M' remains unchanged

In particular, S is maximal and, thus, a cocircuit if and only if $\det(\mathbf{B}, \mathbf{a}_i) \neq 0$ *for all* $i \in [n]\setminus S$.

Proof. The assertion follows from the fact that the right-hand side is the sign of a linear form in $\mathbf{a}_1, \ldots, \mathbf{a}_r$ that vanishes on all points in $\mathbf{A}_{*,S}$, which was assumed to be hyperplanar. □

That is, whenever during the enumeration a right-maximal hyperplanar subset is reached, the first $r-1$ non-zero columns M_{NZ} of the corresponding matrix in the local data determine the signature of the unique cocircuit containing it.

```
Algorithm: IsMaxInDownset($S, \mathscr{D}, \mathbf{A}, \mathsf{M}$)
Input: A subset $S$ of column indices of $\mathbf{A}$, the downset $\mathscr{D}$ of coplanar subsets, the
       configuration $\mathbf{A}$, and a column-echelon form M of $\mathbf{A}_{*,S}$
Output: TRUE if $S$ is maximal in $\mathscr{D}$ and FALSE otherwise
for $i \in [n] \setminus S$ do                          /* for all elements outside $S$ */
    if $\det(\mathsf{M}_{\mathrm{NZ}}, \mathbf{a}_i) = 0$ then           /* if the signature of $i$ is zero */
        return FALSE ;                                  /* $S$ is not maximal */
return TRUE;
```

Algorithm 17: Check whether a subset of columns is maximally coplanar

7.2 Analysis

The following can be said about the efficiency of this algorithm.

Theorem 6. *Assume that the critical-element method is used for* `IsLexMin` *in Algorithm 8. Then, the run-time complexity of the resulting specialized algorithm is in* $O\big(n(|\mathfrak{G}|+nr)|\mathscr{D}^{\text{nonprunable}}/\mathfrak{G}|\big)$.

Moreover, the effectivity of rank-pruning depends on the instance. More specifically, with symmetries ignored:

(i) There is a point configuration $\mathbf{\Delta}_{r-1}^{\partial\text{c}}$ *with* $n = 2r$ *points of rank* r *with* $\frac{r(r+3)}{2}$ *cocircuits for which the total number of not right-completable subsets is in* $\Omega(2^r)$*, whereas none of the not right-completable subsets is non-prunable.*

(ii) There is a point configuration $\mathbf{\Delta}_{r-1}^{\text{dup}}$ *with* $n = 2r$ *points of rank* r *with* r *cocircuits for which the number of not right-completable subsets that are non-prunable is in* $\Omega(2^r)$.

The first item shows that, for the enumeration and listing problems, rank-pruning can lead to a spead-up from exponential to polynomial in the input and output sizes, whereas the second item shows that, in general, rank-pruning cannot guarantee a run-time polynomial in the input and output sizes.

Proof. The number of recursive calls equals the number of orbits of not-prunable subsets. For each such subset there are at most n traversals of the main loop. In each main-loop the worst-case run-time complexity is dominated by `SemiIsNotRightComp`, which computes column-echelon forms for each non-element of the current subset. This amounts to at most n column-echelon-form computations, taking at most r operations each. This proves the run-time bound.

For item (i), consider for $r \geq 3$ the point configuration $\mathbf{\Delta}_{r-1}^{\partial\text{c}}$ consisting of the $(r-1)$-dimensional standard simplex $\mathbf{\Delta}_{r-1}$ together with r copies of the barycenter $\mathbf{c}$ of its lex-min facet $\{1,2,\ldots,r-1\}$, forming the elements indexed by $r+1, r+2, \ldots, 2r$. There are r cocircuits correponding to the r facets of $\mathbf{\Delta}_{r-1}$: $r-1$ of them not containing any $\mathbf{c}$ and one containing all copies of $\mathbf{c}$. Moreover, there are $\binom{r}{r-2} = \binom{r}{2} = \frac{r(r+1)}{2}$ cocircuits containing all copies of $\mathbf{c}$ but not the lex-min facet. In any non-empty subset S call an element $i \in [n] \setminus S$ with $i < \max S$ a *gap* in S. The empty subset, by definition, has no gaps.

A coplanar subset S is not right-completable if and only if each cocircuit containing it contains a gap in S. Define for a subset S the part $R := S \cap \{1,2,\ldots,r\}$ corresponding to $\mathbf{\Delta}_{r-1}$ and the part $C := S \cap \{r+1, r+2, \ldots, 2r\}$ corresponding to the copies of $\mathbf{c}$. Call a subset $S = R \cup C$ a *deadend* if one of the following cases occurs:

- C has at least one gap.
- C has no gap, is non-empty, and $|R| \leq r-3$.
- C is empty, and R has at least three gaps.

In the following it is shown that a coplanar S is not right-completable if and only if it is a deadend. Assume S is a deadend. Then, in the first case S violates the first rank bound, in the second and third case S violates the second rank bound in Theorem 5. In particular, S is prunable. Hence, it is not right-completable.

If S is not a deadend, then, in particular, C has no gaps. Moreover, if C is non-empty, then R has $r-1$ or $r-2$ elements. If R has exactly $r-1$ elements, then R is a facet of $\mathbf{\Delta}_{r-1}$. Since S is coplanar and C is non-empty, this facet can only be the lex-min facet $\{1,2,\ldots,r-1\}$. Then, because C has no gaps, a right-expansion with all remaining copies $\{\max C+1, \max C + 2\ldots,2r\}$ of $\mathbf{c}$ leads to the cocircuit $\{1,2,\ldots,r-1,r+1,r+2,\ldots,2r\}$. If R has exactly $r-2$ elements, then $\mathbf{c}$ is affinely independent of R in any case. Thus, the same right-expansion as above leads to the cocircuit $R\cup\{r+1,r+2,\ldots,2r\}$. If C is empty, then $R=S$ has at most two gaps. If R has exactly one gap, then S is already a facet of $\mathbf{\Delta}_r$. If it is the lex-min facet, a right-expansion with all copies $\{r+1,r+2,\ldots,2r\}$ of $\mathbf{c}$ leads to a cocircuit, and otherwise S is a cocircuit already. If R has exactly two gaps, then, a right-expansion with $\{\max R+1,\max R+2,\ldots,r\}$ (leading to a rank-$r-2$-subset) followed by a right-expansion with all copies $\{r+1,r+2,\ldots,2r\}$ of $\mathbf{c}$ (all affinely independent of the elements in R) leads to a cocircuit.

Consequently, being a deadend is equivalent to being not right-completable. Since all deadends are prunable, so are all not right-completable subsets. Roughly estimated by counting the possible C's in deadends, the number of not right-completable subsets is in $\Omega(2^r)$, and all of them can be rank-pruned.

For item (ii), consider for $r\geq 3$ the point configuration $\mathbf{\Delta}_{r-1}^{\text{dup}}$ consisting of the $(r-1)$-dimensional standard simplex $\mathbf{\Delta}_{r-1}$ together with a copy of each of its elements, forming the elements indexed by $r+1,r+2,\ldots,2r$. There are r cocircuits corresponding to the r facets of the standard simplex. No subset S with $\emptyset\neq S\subseteq\{3,4,\ldots,r\}$ is right-completable, since any cocircuit containing it must also contain one of the points 1 or 2, both gaps in S. Moreover, such a subset S is non-prunable since, first, no non-element to the left is in the span of it (the first r points are independent) and, second, the non-elements to the right can sufficiently increase the rank of any such S (its possible right-expansions contain a complete copy of the standard $r-1$-simplex). The number of these subsets is $2^{r-2}-1$, which is in $\Omega(2^r)$. □

If $\mathfrak{G}$ is given as an explicit set of permutations, the run-time is polynomial in input size and the number of all non-prunable subsets up to symmetry. However, as the example in Theorem 6(ii) shows, Algorithm 17 for counting/enumeration/listing is, in general, not polynomial in the input and output sizes.

7.3 Results

The modified switch-table method in Algorithm 13 turned out to be the fastest variant of `IsLexMin` for all the larger instances. (See the end of Section 5 for explanations concerning the point configurations.) Particularly interesting is the enumeration of hyperplanes spanned by the vertices of the d-dimensional hypercube $\mathbf{C}_d$. This problem has already been studied a long time ago by Aichholzer and Aurenhammer [1]. Using cleverly a lot of structural properties of hypercubes in particular, they were able to enumerate all hyperplanes spanned by the vertices of the 8-cube, while the 9-cube's hyperplanes remained out-of-reach. In contrast to their efforts, the general-purpose algorithm of this paper could compute their numbers without using any specific knowledge about cubes. And it was able to compute the number of cocircuits (total and up to symmetry) of the 9-cube. For $\mathbf{C}_9$ compare the total number of its cocircuits to the number of all its $r-1$-subsets, which is $\binom{512}{9} = 6{,}208{,}116{,}950{,}265{,}950{,}720$ (four orders of magnitude larger) – direct signature computations for all these subsets, given today's computation power, would have been out of reach. Table 5 shows all the results.

A	# symmetries	# cocircuits up to symmetry	# cocircuits in total	# nodes	CPU time [hh:mm:ss]
$\mathbf{C}_2$	8	2	6	9	0:00:00
$\mathbf{C}_3$	48	3	20	31	0:00:00
$\mathbf{C}_4$	384	6	140	126	0:00:00
$\mathbf{C}_5$	3840	15	3254	609	0:00:00
$\mathbf{C}_6$	46,080	63	252,434	4149	0:00:01
$\mathbf{C}_7$	645,120	623	71,343,208	55,540	0:00:04
$\mathbf{C}_8$	10,321,920	22,432	86,246,755,608	2,403,058	0:03:02
*$\mathbf{C}_9$	185,794,560	3,899,720	448,691,419,804,586	530,623,381	13:30:12
*$\mathbf{\Delta}(8,3)$	40,320	56	166,420	4,644	0:00:00
*$\mathbf{\Delta}(8,4)$	80,640	83	1,105,575	9,081	0:00:00
*$\mathbf{\Delta}(9,3)$	362,880	231	10,004,154	21,034	0:00:02
*$\mathbf{\Delta}(9,4)$	362,880	2,522	359,022,180	226,077	0:00:07
*$\mathbf{\Delta}(10,3)$	3,628,800	1,337	889,205,792	113,814	0:00:21
*$\mathbf{\Delta}(10,4)$	3,628,800	87,254	178,227,172,388	6,889,144	0:07:38
*$\mathbf{\Delta}(10,5)$	7,257,600	194,489	939,079,703,204	21,423,661	1:02:09

Table 5: Computational results for the enumeration of cocircuits in hypercubes and hypersimplices using 16 threads (numbers with a "*" are new)

8 Application II: Circuits

In this section it is shown how all circuits of a point configuration **A** can be enumerated up to symmetry by exploring the downset $\mathscr{D}$ of all independent subsets using `SymLexSubsetRSComin_withData` (Algorithm 9).

8.1 Specializations

In order to use `SymLexSubsetRSComin_withData`, one has to specify how its problem-specific subroutines `IsInDownset`, `SemiIsNotRightExit`, and `IsCominOfDownset` work. Again, in these subroutines global data (preprocessed prior to the enumeration) and local data (updated in each enumeration node) can be utilized to speed up the computations and avoid duplicate work. This time, the local data will consist of a matrix in column-echelon form stacked on top of another matrix. The columns of the additional matrix yield additional information useful for circuits. For a subset S the columns of the top matrix are the columns of a column-echelon form of the sub configuration $\mathbf{A}_{*,S}$. The column in the bottom matrix contains the coefficients of a linear combination of all original columns weakly to the left that yields the column on top. If the column on top is the zero-column, then the corresponding original column can be combined linearly from original columns strictly to the left, and the signs of the coefficients in the bottom column yield the corresponding circuit signature.

Formally, the stacked matrix can be defined as follows:

Definition 14. Let $\mathbf{A}_{*,S}$ be a subset of columns of a rank-r configuration $\mathbf{A}$. For an integer $k \in [1, r+1]$ let $\mathbf{I}_k$ denote the $k \times k$-identity matrix.

A $(r+|S|) \times |S|$-matrix $\mathbf{R} = \binom{\mathbf{B}}{\mathbf{C}}$ is a *column-representation matrix of* S if it is a column-echelon form of the matrix

$$\begin{pmatrix} \mathbf{A}_{*,S} \\ \mathbf{I}_{|S|} \end{pmatrix} \tag{120}$$

Here, $\mathbf{B}$ is the *configuration part*, whereas $\mathbf{C}$ is the *coefficient part*.

Theorem 7. *Let* $\mathbf{R} = \binom{\mathbf{B}}{\mathbf{C}}$ *be a column-representation matrix of a subset* S *of column indices of* $\mathbf{A}$. *Then* S *is dependent if and only if* $\mathbf{B}_{*,|S|} = \mathbf{0}$. *Moreover,* S *is a circuit if and only if* $\mathbf{B}_{*,|S|}$ *is the first zero-column in* $\mathbf{B}$ *and* $\mathbf{C}_{|S|}$ *contains no zero entry. In that case, the signs in* $\mathbf{C}_{|S|}$ *specify one of the two possible circuits signatures of* S *restricted to its support* S.

Proof. Note that

$$\begin{pmatrix} \mathbf{A}_{*,S} \\ \mathbf{I}_{|S|} \end{pmatrix} \cdot \mathbf{I}_{|S|} = \begin{pmatrix} \mathbf{A}_{*,S} \\ \mathbf{I}_{|S|} \end{pmatrix}. \tag{121}$$

If this is transformed by admissible column operations, represented by the multiplication of a matrix $\mathbf{C}$ from the right, into column-echelon form, one has:

$$\begin{pmatrix} \mathbf{A}_{*,S} \\ \mathbf{I}_{|S|} \end{pmatrix} \cdot \mathbf{C} = \begin{pmatrix} \mathbf{B} \\ \mathbf{C} \end{pmatrix}. \tag{122}$$

with a column-representation matrix $\binom{\mathbf{B}}{\mathbf{C}}$ of S. In particular, one has for the last column:

$$\mathbf{A}_{*,S} \cdot \mathbf{C}_{*,|S|} = \mathbf{B}_{*,|S|}. \tag{123}$$

S is dependent if and only if the last column is zero, by the properties of a column-echelon form. Moreover, the entries of $\mathbf{C}_{*,|S|}$ constitute a linear dependence. Since the first rank $\mathbf{B}$ columns of $\mathbf{B}$ are linearly independent and $\mathbf{B}_{*,|S|}$ is the first zero-column in $\mathbf{B}$, one has that $\operatorname{rank}(\mathbf{A}_{*,S}) = |S| - 1$. Thus, the kernel of $\mathbf{A}_{*,S}$ is one-dimensional, and a non-zero vector in it is unique up to a non-zero scalar multiple. Therefore, the signs of $\mathbf{C}_{*,|S|}$ are unique up to sign-reversal. Thus, the non-zero components of $\mathbf{C}_{*,|S|}$ constitute the inclusion minimal support of a linear dependence among the columns in $\mathbf{A}_{*,S}$. Hence, the signs of the components of the complete vector $\mathbf{C}_{*,|S|}$ are a signature on the support of a circuit S if anxd only if there are no zero-components in it. $\square$

One can derive the information to process a node from the column-representation matrix of its subset. For the membership in the downset of independent sets this is straight-forward. Algorithm 18 shows the procedure.

```
Algorithm: IsInDownset($S', \mathscr{D}, \mathbf{A}, \mathsf{R}, \mathsf{R}'$)
Input: A subset $S'$ of column indices of $\mathbf{A}$, the downset $\mathscr{D}$ of coplanar subsets, the
        configuration $\mathbf{A}$, a column-representation matrix $\mathsf{R} = \binom{\mathsf{B}}{\mathsf{C}}$ of $S'$, and the augmented
        matrix $\mathsf{R}' = \left(\binom{\mathsf{R}}{\mathbf{0}^\top}, \binom{\mathbf{A}_{*,\max S'}}{\mathbf{e}_{|S'|}}\right)$ (not yet in column-echelon form, in general), where $\mathbf{e}_{|S'|}$ is
        the unit vector in $\mathbb{R}^{|S'|}$ with a one in its last component
Output: (TRUE, $\mathsf{R}'$) with $\mathsf{R}' = \binom{\mathsf{B}'}{\mathsf{C}'}$ a column-representation matrix of $S'$ if rank($\mathsf{R}'$) = $|S'|$ and
        (FALSE, $\mathsf{R}'$) otherwise
$\mathsf{R}' \leftarrow$ colechForm($\mathsf{R}'$) ;                    /* compute column-echelon form */
if $\mathsf{B}'_{*,|S'|} \neq \mathbf{0}$ then               /* if right-most conf-column non-zero */
    return (TRUE, $\mathsf{R}'$) ;                                /* $S'$ is independent */
else
    return (FALSE, $\mathsf{R}'$) ;                                 /* $S'$ is dependent */
```

Algorithm 18: Check whether a subset of columns is independent

Given Lemma 7 it is easy to check a subset for co-minimality. The corresponding algorithm is listed in Algorithm 19.

```
Algorithm: IsCominOfDownset($S', \mathscr{D}, \mathbf{A}, \mathsf{R}, \mathsf{R}'$)
Input: A subset $S'$ of column indices of $\mathbf{A}$, the downset $\mathscr{D}$ of independent subsets, the
        configuration $\mathbf{A}$, and local data given by a column-representation matrix $\mathsf{R}' = \binom{\mathsf{B}'}{\mathsf{C}'}$
        of $S'$
Output: TRUE if $S'$ is co-minimal in $\mathscr{D}$, FALSE otherwise
if $\mathsf{C}'_{*,|S'|}$ has zero-entries then                     /* zero-coefficients?  */
    return FALSE ;                                       /* subset is not co-min */
else
    return TRUE ;                                            /* subset is co-min */
```

Algorithm 19: Check whether a subset of columns is contained in a circuit utilizing the local data given by a column representation matrix

8.2 Analysis

Since each maximal independent subset is touched by the algorithm and each maximal independent subset is built on a unique path of length at most the rank in the enumeration tree, the efficiency of the algorithm solely depends on how many (maximal) independent sets one can find compared to the number of circuits.

Theorem 8. *Assume that the critical-element method is used for* `IsLexMin` *in Algorithm 9. Then, the run-time complexity of the resulting specialized algorithm is in $O(n(|\mathfrak{G}|+r)|\mathscr{D}/\mathfrak{G}|)$.*

Moreover:

(i) There is a point configuration $\mathbf{\Delta}^{\mathrm{c}}_{r-1}$ with $n = 2r$ points of rank r with $\frac{r(r+3)}{2}$ circuits for which the total number of maximal independent subsets is $1 + r^2$.

(ii) There is a point configuration $\mathbf{\Delta}^{\mathrm{dup}}_{r-1}$ with $n = 2r$ points of rank r with r circuits for which the number of maximal independent subsets is 2^r.

Because the number of $(r+1)$-subsets in the example configurations is $\binom{2r}{r+1}$, the algorithm achieves a speed-up in both cases compared to the naive algorithm that computes the signature of each $(r+1)$-subset. In the former case, the speed-up is substantial from exponential to polynomial, in the latter case it "only" reduces the exponential runtime of the naive algorithm by an exponential factor to a shorter exponential runtime.

Proof. The number of recursive calls equals the number of orbits of independent subsets. For each such subset there are at most n traversals of the main loop. In that loop, the lex-min check takes time in $O(|\mathfrak{G}|)$. The worst-case run-time complexity of the remaining subroutines in the main loop is in $O(r)$. This proves the run-time bound.

For item (i) consider the point configuration $\mathbf{\Delta}^{\mathrm{c}}_{r-1}$ consisting of the $(r-1)$-dimensional standard simplex $\mathbf{\Delta}_{r-1}$ together with r copies of its barycenter $\mathbf{c}$, forming the elements indexed by $r+1, r+2, \dots, 2r$. There are r circuits containing the standard simplex with an arbitrary copy of $\mathbf{c}$ and $\binom{r}{2}$ circuits consisting of two copies of $\mathbf{c}$, resulting in $\frac{r(r+3)}{2}$ circuits in total. The maximal independent subsets are the standard simplex and an arbitrary copy of $\mathbf{c}$ with an arbitrary facet of the standard simplex, leading to a total of $1 + r^2$ many maximal independent subsets.

For item (ii) consider the point configuration $\mathbf{\Delta}^{\mathrm{dup}}_{r-1}$ consisting of the $(r-1)$-dimensional standard simplex $\mathbf{\Delta}_{r-1}$ together with a copy of each of its elements, forming the elements indexed by $r+1, r+2, \dots, 2r$. There are r circuits corresponding to the r pairs of identical points. In contrast to this, any choice of a copy for the r many vertices of the standard simplex is a maximal independent subset, resulting in 2^r many of them. □

If $\mathfrak{G}$ is given as an explicit set of permutations, the runtime is polynomial in the input size and the number of all independent sets up to symmetry. However, the example in Theorem 8(ii) shows that for counting/enumeration/listing it is, in general, not polynomial in the input and output sizes.

Note that even for graphic matroids it is NP-hard to decide whether a given subset of elements is contained in a circuit [17, Section 4]. Call this NP-hard decision problem the "Extension-to-a-Circuit Problem (ECP)". There is no order involved in this result. However, given an instance to the ECP for a vector matroid given by a matrix, one can reorder the columns of the matrix to start with the subset in question. Then, all potential circuits containing the subset lex-contain it. Thus, an answer to the right-exitability problem for circuit enumeration would answer the ECP instance as well. Thus, it cannot be expected that all deadends can be avoided efficiently in our enumeration algorithm.

Even a useful semi-check for right-exitability of a subset for circuit enumeration is still unknown. Thus, for circuit enumeration, `SemiIsNotRightExit` is defined to simply return FALSE for each subset (i.e., the subset might be a left segment of a co-minimal subset of $\mathscr{D}$).

While this seems unsatisfactory at first glance, there are examples where the difficulty of showing right-non-exitability becomes plausible. Extend the standard simplex to the augmented standard simplex $(\mathbf{\Delta}_n, \mathbf{p})$ with $n+1$ points, where $\mathbf{p}$ is an additional point at the barycenter of some geometric k-face of $\mathbf{\Delta}_n$, $k = 0, \dots, n$, the *centered face*. There is exactly one circuit, namely the new point $\mathbf{p}$ together with the centered face. Whether or not any subset of the points in $(\mathbf{\Delta}_n)$ is contained in a circuit now depends on lex-containment in the centered face. Any useful semi-decision algorithm for right-non-exitability would have to find out the centered face (without actually knowing in advance that the centered face is the key object).

8.3 Results

Table 6 shows some results that could be obtained using the resulting algorithm, where again the modified switch-table method in Algorithm 13 turned out to be the fastest variant of `IsLexMin` for the larger instances (see the end of Section 5 for explanations concerning the point configurations).

A	# symmetries	# circuits up to symmetry	# circuits in total	# nodes	CPU time [hh:mm:ss]
$\mathbf{C}_2$	1	1	1	11	0:00:00
$\mathbf{C}_3$	48	3	20	40	0:00:00
$\mathbf{C}_4$	384	15	1348	219	0:00:00
$\mathbf{C}_5$	3840	186	353,616	2616	0:00:00
*$\mathbf{C}_6$	46,080	12,628	446,148,992	119,638	0:00:01
*$\mathbf{C}_7$	645,120	3,591,868	2,118,502,178,496	25,274,904	0:04:49
*$\mathbf{C}_8$	10,321,920	3,858,105,362	38,636,185,528,212,416	21,028,416,821	163:37:00
*$\mathbf{\Delta}(8,3)$	40,320	7,240	251,651,820	153,429	0:00:01
*$\mathbf{\Delta}(8,4)$	80,640	41,875	3,134,451,775	670,792	0:00:04
*$\mathbf{\Delta}(9,3)$	362,880	228,432	75,267,509,940	4,298,974	0:00:28
*$\mathbf{\Delta}(9,4)$	362,880	31,671,609	11,259,090,122,490	346,869,278	1:05:40
*$\mathbf{\Delta}(10,3)$	3,628,800	7,494,056	25,290,095,161,170	140,451,528	0:20:42
*$\mathbf{\Delta}(10,4)$	3,628,800	12,609,824,635	45,270,853,845,998,550	110,076,768,816	523:25:52

Table 6: Computational results for the enumeration of circuits in hypercubes and hypersimplices using 16 threads (numbers with a "*" are new)

9 Application III: Triangulations

In this section it is shown how all triangulations of a point configuration **A** can be enumerated up to symmetry by exploring the downset $\mathscr{D}$ of all subsets of mutually properly intersecting simplices using SymLexSubsetRSFeas_withData (Algorithm 10).

Most general-purpose enumeration algorithms for triangulations rely on an algorithm based on the flip graph of triangulations [25, 8, 14]. Since Santos found a triangulation without flips [29] it is known that one might not find all triangulations this way. There have always been hints in the literature on how to enumerate all triangulations of a point configuration by enumerating maximal cliques in the proper-intersection graph of all simplices. However, to date no implementation of this idea could ever compete with flip-based algorithms.

Here, an all new attempt is presented based on symmetric lexicographic subset reverse search for feasible subsets with pruning (Algorithm 10). In the following, a subset of pairwise properly intersecting simplices is called a *partial triangulation*. Note that, according to this definition, all triangulations are partial triangulations as well.

9.1 Specializations

As a first step towards the application of SymLexSubsetRSFeas_withData (Algorithm 10), one has to specify how triangulations are represented as subsets of some finite set. To this end, let $\mathbf{A}$ be a rank-r configuration with n elements. Let $\mathscr{S}$ be the set of feasible simplices of $\mathbf{A}$ (where "feasible" may depend on the exact enumeration task, see Section 9.4 for some examples), and let n_s be its cardinality. Similarly, let $\mathscr{R}$ be the set of all simplex-facets not contained in any facet of $\mathbf{A}$ (to be distinguished from the facets of $\mathbf{A}$) and n_f be its cardinality.

By fixing an arbitrary bijection $\mathscr{S} \to [n_s]$ to encode simplices, any triangulation $\mathscr{T}$ given by its set of maximal simplices can be represented as a subset T of $[n_s]$. While any bijection will allow to use Algorithm 10, there is a special bijection that helps to accelerate the semi-check for the non-right-extendability of a subset. For the rest of this section, let $\mathrm{idx}_s : (\mathscr{S}, <_{\mathrm{lex}}) \to ([n_s], <)$ and $\mathrm{idx}_f : (\mathscr{R}, <_{\mathrm{lex}}) \to ([n_f], <)$ be the respective order-preserving bijections. For convenience, denote the inverse functions by $\mathrm{simp} = \mathrm{idx}_s^{-1}$ and $\mathrm{facet} = \mathrm{idx}_f^{-1}$, respectively. With this notation, the subset T corresponding to a triangulation $\mathscr{T}$ is given by $T = \{s \in [n_s] : \mathrm{simp}(s) \in \mathscr{T}\}$, and the triangulation corresponding to a subset T is given by $\mathscr{T} = \{S \in \mathscr{S} : \mathrm{idx}_s(S) \in T\}$. This motivates to say that T *indexes* $\mathscr{T}$.

A brief inspection of Algorithm 10 shows that this results in the following: not only are the triangulations enumerated in the lexicographic order of $2^{[n_s]}$, but also each triangulation itself is built by adding simplices one-by-one in lexicographic order.

The downset $\mathscr{D}$ used in the method is the set of subsets indexing simplex subsets that are pairwise properly intersecting. The set $\mathscr{F}$ of feasible subsets is the set of all subsets indexing a triangulation. Since no triangulation strictly contains any other triangulation, one can apply Algorithm 10 by specializing its subroutines IsFeasible, Expand, and SemiIsNotRightExt.

Next, some notions are introduced that help to set up appropriate global and local auxiliary data, which will speed up the implementation. The global auxiliary data is defined first. It is desirable to access quickly the incidence information of spanning simplices and their facets. Given the characterization of a triangulation in Definition 13, the information about the which simplices contain which interior simplex facets is important.

Definition 15. For a rank-r-configuration **A** with n points, n_s many simplices, and n_f many simplex-facets, define the *interior-facets table* as

$$\text{intfacets}\colon \left\{ \begin{array}{rcl} [n_s] & \longrightarrow & 2^{[n_f]}; \\ s & \longmapsto & \bigl\{f \in [n_f] : \text{facet}(f) \text{ is an interior facet of simp}(s)\bigr\}, \end{array} \right. \tag{124}$$

and the reverse *convering-simplices table* as

$$\text{covsimps}\colon \left\{ \begin{array}{rcl} [n_f] & \longrightarrow & 2^{[n_s]}; \\ f & \longmapsto & \bigl\{s \in [n_s] : \text{facet}(f) \text{ is an interior facet of simp}(s)\bigr\}. \end{array} \right. \tag{125}$$

In Algorithm 10 an expansion sequence is used. In the current application such a sequence corresponds to a sequence of lexicographically greater simplices that can be added to a partial triangulation without violating proper intersection. Such a sequence can be updated more easily when for all simplices the set of all simplices that have a proper intersection with it can be accessed quickly.

Definition 16. For a rank-r-configuration **A** with n points and n_s many simplices, define the *admissibles table* as

$$\text{admsimps}\colon \left\{ \begin{array}{rcl} [n_s] & \longrightarrow & 2^{[n_s]}; \\ s & \longmapsto & \bigl\{s' \in [n_s] : \text{simp}(s') \text{ intersects properly with simp}(s)\bigr\}. \end{array} \right. \tag{126}$$

Next, some local auxiliary data is defined: with each subset indexing a partial triangulation store

- the index set of all lex-greater simplices intersecting properly with it;
- the index set of all uncovered interior facets.

To this end, define:

Definition 17. Let T index a partial triangulation. Then the *admissibles of* T are defined as

$$\text{admissibles}(T) := \bigl\{s \in [n_s] : s > s', \text{simp}(s) \text{ intersects simp}(s') \text{ properly } \forall s' \in T\bigr\}. \tag{127}$$

Moreover, define the *free interior facets of* T *as*

$$\text{freefacets}(T) := \bigl\{f \in [n_f] : \text{facet}(f) \text{ interior facet of simp}(s) \text{ for exactly one } s \in T\bigr\}. \tag{128}$$

From these data, some useful information can directly be derived.

Lemma 9. *Let T index a non-empty partial triangulation. Moreover, if* admissibles$(T) \neq \emptyset$*, let s index a simplex in* admissibles(T)*, and let T' be the expansion $T \cup \{s\}$ of T by s. Then:*

(i) T indexes a triangulation if and only if freefacets$(T) = \emptyset$.

(ii) The admissibles of T constitute the expansion sequence.

```
Algorithm: IsFeasible($T$, $\mathscr{F}$, FIF)
Input: A subset $T$ of simplex indices in $[n_s]$, a set of feasible subsets $\mathscr{F}$ (implicit), the set FIF
       of free interior facets of $T$
Output: TRUE if $T$ is a triangulation, FALSE otherwise
if FIF $= \emptyset$ then                              /* no free interior facets?  */
    return TRUE ;                        /* partial triangulation is covering */
else
    return FALSE ;                   /* partial triangulation is not covering */
```

Algorithm 20: Check whether a partial triangulation is a triangulation by checking whether all free interior facets are covered

(iii) If $\operatorname{admissibles}(T) = \emptyset$ *and* $\operatorname{freefacets}(T) \neq \emptyset$*, then T is not right-extendable.*

(iv) The admissibles of T' can be computed as the intersection

$$\operatorname{admissibles}(T') = \operatorname{admissibles}(T) \cap \operatorname{admsimps}(s). \tag{129}$$

(v) The free interior facets of T' can be computed as the symmetric difference

$$\operatorname{freefacets}(T') = \operatorname{freefacets}(T) \triangle \operatorname{intfacets}(s). \tag{130}$$

Proof. The assertions follow from straight-forward checks of definitions. □

This is essentially what was used to-date in any attempt to enumerate all triangulations of a configuration, compare [25, 8]. These methods did not scale well because without any additional pruning they would process a very large number of nodes. For later reference, call this the *no-pruning* method.

One significant improvement over this are certain necessary conditions for expansions being extensions. The discussion starts with a condition that, if false, allows to break out of the loop over all expansions immediately.

Theorem 9. *Let T index a non-empty partial triangulation with* $\operatorname{freefacets}(T)$ *and* $\operatorname{admissibles}(T)$ *both non-empty.*

Then, if $T \cup \{s'\}$ is right-extendable for some $s' \geq s$ with $s, s' \in \operatorname{admissibles}(T)$, then

$$\min\big(\operatorname{freefacets}(T)\big) \geq \min\big(\operatorname{intfacets}(s)\big). \tag{131}$$

Proof. If the minimal interior facet index of an admissible s is too large to cover the minimal free facet of T, then the same holds for each admissible $s' > s$ as well, since

$$\min\big(\operatorname{intfacets}(s')\big) \geq \min\big(\operatorname{intfacets}(s)\big) \text{ for each } s' > s. \tag{132}$$

The theorem is a formal version of this observation. □

```
Algorithm: Expand($T, s, \mathbf{G}, \mathbf{L}$)
Input: A subset $T$ of simplex indices in $[n_s]$, a new simplex index $s \in \mathsf{ADM}$, global data $\mathbf{G}$
       encompassing the interior-facets table IT and the admissibles table AT of $\mathbf{A}$, local data
       $\mathbf{L}$ of $T$ encompassing the free interior facets FIF and the admissibles ADM of $T$
Output: (break, $T', \mathbf{L}'$), where break is true if this and all future expansions cannot be
        extensions, $(T', \mathbf{L}')$ is the new node with $T' = T \cup \{s\}$, and $\mathbf{L}'$ is the correct local Data
        for $T'$
if $\min(\mathsf{FIF}) < \min(\mathsf{IT}(s))$ then                /* can $s$ cover minimal free facet?  */
    return (TRUE, –, –) ;                       /* min. free facet not coverable by $s' \geq s$ */
$T' \leftarrow T \cup \{s\}$ ;                                           /* add $s$ to the subset */
$\mathsf{FIF}' \leftarrow \mathsf{FIF} \triangle \mathsf{IT}[s]$ ;          /* free interior facets by symmetric difference */
$\mathsf{ADM}' \leftarrow \mathsf{ADM} \cap \mathsf{AT}[s]$ ;                          /* admissibles by intersection */
$\mathbf{L}' \leftarrow (\mathsf{FIF}', \mathsf{ADM}')$ ;                               /* put together local data */
return (FALSE, $T', \mathbf{L}'$) ;                                          /* return the node */
```

Algorithm 21: Expand a subset indexing a partial triangulation by a new simplex index

Since all partial triangulations are built in lexicographic order, Theorem 9 means the following: the loop over the expansion sequence in Algorithm 10 can be left as soon as the minimal interior facet index of the new simplex index is larger than the minimal free interior facet index of T, in which case $T \cup \{s\}$ cannot be right-extended. The lexicographic building order guarantees that the minimal free interior facet cannot be covered by any simplex added in the future either. The application of this necessary condition is done inside the Expand subroutine and is called *lex-breaking*.

In the sequel, some necessary conditions for right-extendability are discussed. Before going into the details, note that it is unlikely to find a complete and efficient exact characterization of right-extendability. The reason is the following. The decision problem of whether or not a partial triangulation can be extended to a triangulation is NP-hard in general. This can be derived from the NP-hardness of triangulating non-convex 3-polytopes [28], because for a general partial triangulation the uncovered part yet to be triangulated is, in general, non-convex. Here, only extensions to the right are interesting w.r.t. a certain ordering of simplices. However, since for the purpose of this decision problem one can reorder the simplices in such a way that the simplices of the given partial triangulation come first, the right-extendability problem is NP-hard as well. Thus, it is justified to resort to necessary conditions, resulting in a semi-check for pruning. First, the strongest necessary condition for right-extendability is established.

Theorem 10. *Let T index a non-empty partial triangulation. Then: If T is right-extendable, then there is a covering set of simplex indices $C \subseteq$ admissibles(T) such that*

(i) the covering set is pairwise properly intersecting: For each $s \in C$ one has for all $s' \in C \setminus \{s\}$ that $s \in$ admsimps(s'),

(ii) the covering set covers all free interior facets: For each $f \in$ freefacets(T) there is an $s \in C$

with $f \in \text{intfacets}(s)$.

Proof. Note that the set of new simplices in any feasible right-completion of T

- stems from the current admissibles of T,
- is itself properly intersecting,
- contains for each free interior facet a simplex containing that facet.

Thus, any feasible right-completion is a special case of a covering set C as in the theorem. □

Call the application of Theorem 10 *full-pruning*. Note that the existence of a covering set of simplices as in Theorem 10 does not guarantee the right-extendability, since its elements may lead to new free interior facets that have to be covered by even more simplices, which may fail at some point. It is not straight-forward how to implement full-pruning without branching-out the potential covering sets of simplices – after all, the idea of pruning is essentially to avoid branching in the first place.

Thus, the following weaker test was developped. It does not demand a covering set of simplices which is pairwise intersecting properly. Instead, for each free interior facet it is checked whether there is a covering simplex that is intersecting properly with at least one such covering simplex for each other free interior facet.

Theorem 11. *Let T index a non-empty partial triangulation. Then: If T is right-extendable, then for each $f \in \text{freefacets}(T)$ there is a multi-covering set of simplex indices $C(f) \subseteq \text{admissibles}(T)$, called the f-*covering simplices*, such that*

(i) there is an $s_f \in C(f)$ that is identical to or intersects properly with at least one $s_{f'} \in C(f')$ for all $f' \in \text{freefacets}(T)$,

(ii) for all $f \in \text{freefacets}(T)$ and all $s_f \in C(f)$, $f \in \text{intfacets}(s_f)$.

Proof. Given a covering set of simplex indices as in Theorem 10, $C(f)$ can be set to the unique simplex index in C containing f. □

Call the application of the following Theorem 11 *strong pruning*. It was this necessary condition that, for the first time ever, allowed the enumeration of all symmetry classes of triangulations for instances like the 4-cube (a standard benchmark that has 247,451 symmetry classes of triangulations) in a CPU time comparable to the CPU times of flip-based algorithms.

Given the lex-orders of simplices and facets, one can prove another necessary condition for right-extendability that can be evaluated much faster and has – surprisingly – been almost as effective for the experiments in this paper. It is based on the following theorem that is very similar to Theorem 9.

Theorem 12. *Let T index a non-empty partial triangulation with $\text{freefacets}(T)$ and $\text{admissibles}(T)$ both non-empty.*

Then, if T is right-extendable, the following holds:

$$\min\Big(\text{freefacets}(T)\Big) \geq \min\Big(\text{intfacets}\Big(\min(\text{admissibles}(T))\Big)\Big). \tag{133}$$

```
Algorithm: SemiIsNotRightExtStrong($T', \mathscr{D}, \mathscr{F}, \mathbf{G}, \mathbf{L}, \mathbf{L}'$)
Input: A subset $T'$ of simplex indices in $[n_s]$, the downset $\mathscr{D}$ of subsets indexing partial
        triangulations, the set $\mathscr{F}$ of triangulations (given implicitly by IsFeasible), global
        data $\mathbf{G}$ encompassing the covering-simplices table CS and the admissibles table AT
        of $\mathbf{A}$, local data $\mathbf{L}$ of $T$ encompassing the free interior facets FIF and the admissibles
        ADM of $T'$
Output: TRUE if $T'$ is not right-extendable, FALSE if $T'$ may be right-extendable
for $f \in$ FIF do                                   /* for all free interior facets */
    COVset[$f$] $\leftarrow$ CS[$f$] $\cap$ ADM[$T'$] ;           /* update the $f$-covering simplices */
    if COVset[$f$] $= \emptyset$ then                  /* no admissible $f$-covering simplex? */
        return TRUE ;                              /* $f$ not coverable by admissibles */
    COVSetisNew[$f$] $=$ TRUE ;                         /* $f$-covering simplices are new */
COVAnyisNew $\leftarrow$ TRUE ;                                          /* something is new */
while COVAnyisNew do                                          /* while something is new */
    for $f \in$ FIF do                                   /* for all free interior facets */
        if COVSetisNew[$f$] $=$ FALSE then          /* $f$-covering simplices not new? */
            continue ;                                             /* next loop element */
        /* collect $f$-admissible simplices:                                          */
        COVadm[$f$] $= \bigcup_{s \in \text{COVset}[f]}$ AT[$s$];
    COVAnyisNew $\leftarrow$ FALSE ;                                      /* nothing new so far */
    for $f \in$ FIF do                                   /* for all free interior facets */
        /* collect $f'$-admissibles among the $f$-covering simp's:                    */
        COVset$' = \bigcap_{f' \in \text{FIF} \setminus \{f\}}$(COVset[$f'$] $\cup$ COVadm[$f'$]);
        if COVset$' = \emptyset$ then                 /* no $f$-covering simplex admissible? */
            return TRUE ;                          /* $f$ not coverable by admissibles */
        if COVset$' \not\supseteq$ COVset[$f$] then              /* COVset[$f$] need restriction? */
            /* restrict to $f'$-admissible simplices:                                 */
            COVset[$f$] $\leftarrow$ COVset[$f$] $\cap$ COVset$'$;
            COVSetisNew[$f$] $\leftarrow$ TRUE ;           /* $f$-covering simplices are new */
            COVAnyisNew $\leftarrow$ TRUE ;                        /* something has changed */
return FALSE ;                       /* COVset[$f$] is now a multi-covering set */
```

Algorithm 22: The strong-pruning semi-check whether a partial triangulation can certainly not be right-extended to a triangulation based on a direct application of Theorem 11

Proof. If T is right-extendable, then for each free facet indexed in freefacets(T) there must be an admissible simplex indexed in the admissibles of T covering it, since admissibles(T') $\subseteq$ admissibles(T) for all $T' \supseteq T$. In particular, for the lex-minimal free facet there must be such an admissible simplex. The lex-minimal facet that can be covered by some admissible simplex is the lex-minimal facet of the lex-minimal admissible simplex. Thus, if the lex-

A	# triangulations in total	# nodes by pruning method no	strong	lex	CPU time [hh:mm:ss] by pruning method no	strong	lex
$\mathbf{C}_3$	74	2915	486	497	0.02	0.01	0.01
$\mathbf{\Delta}_3 \times \mathbf{\Delta}_2$	4488	2,385,961	29,423	29,577	1.31	0.11	0.06
$\mathbf{C}(9,4)$	357	8,627,257	4861	4926	12.87	0.08	0.02
$\mathbf{C}(10,4)$	4824	>5,180,000,000	73,085	73,259	>29:54:40.79	0.83	0.11
$\mathbf{C}(11,4)$	96,426	–	1,597,366	1,597,784	–	20.16	1.72

Table 7: Comparison of the three pruning methods no-pruning, strong-pruning, and lex-pruning for a 3-cube, a product of a tetrahedron and a triangle, and some cyclic polytopes (tiny to small instances, single-thread with symmetries ignored)

```
Algorithm: SemiIsNotRightExtLex($T', \mathscr{D}, \mathscr{F}, \mathbf{G}, \mathbf{L}, \mathbf{L}'$)
Input: A subset $T'$ of simplex indices in $[n_s]$, the downset $\mathscr{D}$ of subsets indexing partial
       triangulations, the set $\mathscr{F}$ of triangulations (given implicitly by IsFeasible), global
       data $\mathbf{G}$ encompassing the interior-facets table IT and the admissibles table AT of $\mathbf{A}$,
       local data $\mathbf{L}$ of $T$ encompassing the free interior facets FIF and the admissibles ADM
       of $T'$
Output: TRUE if $T'$ is not right-extendable, FALSE if $T'$ may be right-extendable
if $\min(\mathsf{FIF}) < \min\Big(\mathsf{IT}\big[\min(\mathsf{ADM})\big]\Big)$ then        /* min. free facet too small?  */
    return TRUE ;          /* min. free facet not coverable by admissibles */
else
    return FALSE ;              /* no contradiction to right-extendability */
```

Algorithm 23: The lex-pruning semi-check whether a partial triangulation can certainly not be right-extended to a triangulation based on a direct application of Theorem 12

minimal free facet is lex-smaller than this, then it cannot be covered by *any* admissible simplex of any superset $T' \supseteq T$, and T cannot be right-extendable. The assertion is just a translation of this into a formula. □

Call the application of Theorem 12 *lex-pruning*. In Table 7 a comparison of the node counts is presented for no-pruning and no lex-breaking, strong-pruning with lex-breaking, and lex-pruning with lex-breaking for tiny to small examples. No-pruning is not competitive, which explains the limited success of all implementations previously available. Moreover, the examples (and others not in the table) exhibit that strong-pruning is slightly more effective than lex-pruning. However, in all examples computed so far the nodes pruned extra do not justify the substantially larger effort (see Section 9.2 for an analysis of worst-case run-times).

Now, the specialized subroutines can be presented for application of Algorithm 10 to the enumeration of *all* triangulations. For the instances considered, the fastest variant of `IsLexMin` was `IsLexMin_viaIter` from Algorithm 11. This comes as no surprise (after the analysis in Section 4.2), since the degree n_s is usually large compared to the order k of the symmetry group. This time, the global and local auxiliary data are more voluminous. The global data **G** consists of the configuration **A** with hash maps AT for its admissibles and IT for its interior facets. The local data **L** stored with each subset T of simplex indices consists, besides the local data necessary for the lex-min check (in this case a critical-element table CET of T), a set ADM representing the admissible simplices of T and a set FIF representing the free interior facets of T. The feasibility check just needs part of the local data and is straight-forward (see Algorithm 20). When a node is processed, the corresponding partial triangulation is expanded by a simplex admissible for it. Thus, the set ADM in the local data of a node and ordered lexicographically yields an expansion sequence. When a new simplex is added, one can generate the new local data from the local data of T, as is described in Algorithm 21. Note that lex-breaking according to Theorem 9 might be possible.

Finally, Algorithms 22 and 23 list implementations of strong-pruning and lex-pruning, respectively, that reveal in many cases when a partial triangulation can certainly not be right-extended to a triangulation. Algorithm 22 requires some explanation. In that algorithm, for some partial triangulation T', first, for each free interior facet $f \in \mathsf{FIF}$ construct a set $\mathsf{COVset}[f]$ containing the f-covering simplices that

(a) contain the free interior facet f and

(b) intersect properly with T'.

Second, for each free interior facet $f \in \mathsf{FIF}$ construct the set $\mathsf{COVadm}[f]$ of all so-called *f-admissible simplices* that are identical to or intersect properly with at least one of the f-covering simplices. The goal is to eliminate in rounds over all $f \in \mathsf{FIF}$ all those simplices from the sets of f-covering simplices that are not f'-admissible for at least one other $f' \in \mathsf{FIF}$. As soon as no further f-covering simplex is eliminated in a complete round over all $f \in \mathsf{FIF}$, the f-covering simplices contain a multi-covering set as in Theorem 11. Since in every while-loop at least one f-covering simplex is removed, the number of while-loop traversals and, hence, the algorithm is finite.

9.2 Analysis

Let $\mathscr{D}^{\text{nonprunable}} \subseteq \mathscr{D}$ be the set of subsets satisfying the necessary conditions of Theorem 9 and Theorem 10 or 12, depending on which pruning method is chosen. Then, the following can be said about the run-time complexity of the resulting algorithm.

Theorem 13. *Assume that* `IsLexMin` *is implemented via the critical-element method. Let n_s be the number of simplices in* **A**. *Moreover, let m be the maximal number of simplices in a triangulation of* **A**. *Then, the run-time complexity of* `SymLexSubsetRSFeas_withData` *with strong-pruning is in $O\big(n_s(|\mathfrak{G}| + (r^3m^3c + r^2m^2c^2)n_s))|\mathscr{D}^{\text{nonprunable}}/\mathfrak{G}|\big)$; with lex-pruning the run-time is in $O\big(n_s(|\mathfrak{G}| + n_s + r\,m)|\mathscr{D}^{\text{nonprunable}}/\mathfrak{G}|\big)$.*

Moreover:

(i) For any $r \in \mathbb{N}$, there is a point configuration $\mathbf{\Delta}_{r-1}^{\mathrm{dup}}$ with $2r$ points of rank r whose number of triangulations up to symmetry is one and whose number of simplices n_s is 2^r, for which `SymLexSubsetRSFeas_withData` *indeed traverses its main loop 2^r times.*

(ii) For any $n \in \mathbb{N}$, there is a point configuration $\mathbf{L}_n$ with $n+2$ points of rank 2 whose number of triangulations is in $\Theta(2^n)$, whose number of right-extendable partial triangulations is in $\Theta(2^n)$, and whose number of all partial triangulations is in $\Theta\big((\frac{3+\sqrt{5}}{2})^n\big)$. In particular, asymptotically the number of all partial triangulations is $\frac{1}{2}\big(\frac{3+\sqrt{5}}{4}\big)^n$ times as large as the number of right-extendable partial triangulations. Moreover, all expandable but not right-extendable partial triangulations can be pruned by lex-pruning or strong-pruning.

(iii) For any $d \in \mathbb{N}$, there is a point Configuration $\mathbf{C}_d^*$ with $n = 2d$ points of rank $d+1$ whose number of triangulations is d, whose number of right-extendable partial triangulations is $1 + d2^{d-1}$, and whose number of all partial triangulations is $1 + d\big(2^{(2^{d-1})} - 1\big)$. In particular, the number of all partial triangulations is asymptotically $2^{(2^{d-1}-d)}$ times as large as the number of right-extendable partial triangulations. Moreover, all expandable but not right-extendable partial triangulations can be pruned by strong-pruning, but not necessarily by lex-pruning.

(iv) For any $k \in \mathbb{N}$, there is a point configuration $\mathbf{P}_k$ with $n = 4(k+3)$ points of rank 4 for which, independent of the order of points, there are at least 2^{4k} not right-extendable partial triangulations all of which can be pruned by strong-pruning.

(v) For any $k \in \mathbb{N}$, there is a point configuration $\mathbf{P}_k$ with $n = 5(k+3)$ points of rank 4 for which, independent of the order of points, there are at least 2^{5k} not right-extendable partial triangulations none of which can be pruned by full-pruning, strong-pruning or lex-pruning.

The first item shows that the number of simplices, and, therefore, the number of traversals of the main loop in `SymLexSubsetRSFeas_withData`, can be exponential in the number of triangulation *orbits*. The second and third items show to what extent (symmetries ignored) the number $|\mathscr{D}|$ of all partial triangulations can grow compared to the number of right-extendable partial triangulations and to what extent $\mathscr{D}^{\mathrm{nonprunable}}$ can be smaller than $\mathscr{D}$. The fourth item shows that, symmetries ignored and independent of the order of points, strong-pruning can lead to a pruning of an exponential number of partial triangulations, whereas the fifth item shows that, even with full-pruning, an exponential number of dead-ends in the enumeration tree is possible.

Proof. Preparations: First note that unions and symmetric differences of subsets can be implemented in (amortized) time linear in the sum of the elements of all the operands, whereas intersections can be implemented in (amortized) time linear in the number of elements of the smaller set whenever it is known which set this is. The cardinality of the set of free interior facets FIF in T' is at most rm. The best straight-forward worst-case estimate for the number of simplices in $\mathsf{AT}[s]$ admissible to a given simplex s is the number of all simplices n_s. The best straight-forward worst-case estimate for the number of f-covering simplices $f \cup \{i\}$ in $\mathsf{COVset}[f]$ for a given f is at most the number $c+1 = n-(r-1)$ of points not in f, which is in $O(c)$.

The number of elements in an expansion sequence of a non-empty subset is no larger than the number of admissibles of one of its members, which, by the preparations, is no more than the number n_s of simplices. This is the maximal number of traversals of the main loop in `SymLexSubsetRSFeas_withData`. The dominating remaining subroutine in the main loop besides `IsLexMin` and `SemiIsNotRightExt` is `Expand`. By the preparations, the updates of FIF and ADM take no more than $O(rm + n_s)$ operations. The remaining effort inside the main loop differs according to the pruning method.

For strong-pruning: Since in each while-loop traversal of strong-pruning at least one f-covering simplex is removed, the while-loop traversals are bounded by the number of simplices in $\mathsf{COVset}[f]$ summed over all $f \in \mathsf{FIF}$. Thus, the number of while-loop traversals is in $O(rmc)$. The number of for-loop traversals is in $O(rm)$ for each of the three for-loops.

The effort inside of the loops consists of

(a) the computation for $\mathsf{COVset}[f]$ for all $f \in \mathsf{FIF}$ prior to the while-loop;

(b) the computation of $\mathsf{COVadm}[f]$ for all $f \in \mathsf{FIF}$ inside the while-loop;

(c) the computation of COVset' to update $\mathsf{COVset}[f]$ for all $f \in \mathsf{FIF}$ inside the while-loop.

Concerning item (a), note that the asymptocially smaller subset in the set-intersection computation of $\mathsf{COVset}[f]$ is $\mathsf{CS}[f]$, whose number of elements is in $O(c)$. Thus, the time to compute $\mathsf{COVset}[f]$ for all $f \in \mathsf{FIF}$ is in $O(rmc)$.

Concerning item (b), note that the number of elements involved in the set-union computation of $\mathsf{COVadm}[f]$ is in $O(cn_s)$. Thus, the computation of $\mathsf{COVadm}[f]$ for all $f \in \mathsf{FIF}$ takes time in $O(rmcn_s)$. Since this happens inside the while-loop, the total effort for this is in $O\big((rmc)(rmcn_s)\big) = O(r^2m^2c^2n_s)$.

Finally, concerning (c), note that the total number of elements involved in the mixed set-union and set-intersection computation of COVset' is in $O(rmn_s)$ (for the set-intersections the respective smaller sets are unclear, whence one cannot take advantage of this), which dominates the third for-loop. Hence, the potential restriction of $\mathsf{COVset}[f]$ for all $f \in \mathsf{FIF}$ takes time in $O(r^2m^2n_s)$. Since this happens inside the while-loop, the total effort for this is in $O\big((rmc)(r^2m^2n_s)\big) = O(r^3m^3cn_s)$.

Putting everything together yields that strong-pruning takes time in $O\big((r^3m^3c + r^2m^2c^2)n_s\big)$, which dominates the effort for `Expand`. Summarized, the run-time complexity of `SymLexSubsetRSFeas_withD` for triangulations with strong-pruning is in $O\big(n_s(|\mathfrak{G}| + (r^3m^3c + r^2m^2c^2)n_s))|\mathscr{D}^{\text{nonprunable}}/\mathfrak{G}|\big)$.

For lex-pruning: Whenever for all sets sorted structures are used or their minima are cached during each modification, the run-time complexity of lex-pruning is constant, which is dominated by the effort for `Expand`. Summarized, for lex-pruning the run-time of the resulting algorithm `SymLexSubsetRSFeas_withData` is in $O\big(n_s(|\mathfrak{G}| + rm + n_s)|\mathscr{D}^{\text{nonprunable}}/\mathfrak{G}|\big)$ – an improvement so significant compared to strong-pruning that now `SemiIsNotRightExt` is not even the dominating subroutine anymore!

For item (i), define $\mathbf{\Delta}_{r-1}^{\text{dup}}$ to be the $r-1$-dimensional standard simplex $\mathbf{\Delta}_{r-1}$ with all points duplicated. Up to symmetry, there is only one triangulation. The number of simplices, however, is 2^r. In `SymLexSubsetRSFeas_withData`, each of these simplices will first be added to the empty partial triangulation before the check for lexicographic minimality can dismiss these redundant branches.

For item (ii), consider the point configuration $\mathbf{L}_n$ of rank 2 consisting of $n+2$ distinct points on a line labelled consecutively from 0 through $n+1$. It has 2^n triangulations, each corresponding to the set of used interior points. Any partial triangulation indexed by T induces a *segment pattern* $P(T) := \epsilon_0\epsilon_1\ldots\epsilon_n \in \{-1,0,1\}^{n+1}$ with the following meaning: $\epsilon_i = -1$ if $[i,i+1]$ is not covered by a segment in T, $\epsilon_i = 1$ if $[i,i+1]$ is covered by a segment in T starting at i, and $\epsilon_i = 0$ if $[i,i+1]$ is covered by a segment in T starting at some $k<i$, where $i = 0,1,\ldots,n$. Call a pattern in $\{-1,0,1\}^{n+1}$ *consistent* if it does not start with a "0" and if no "0" ever follows a "−1". Then, $P(T)$ is consistent for all partial triangulations T. Given any consistent pattern $P \in \{-1,0,1\}^{n+1}$, one can construct (from left to right) a partial triangulation T for which $P = P(T)$: A "1" starts a new segment, a "0" continues the same segment, and a "−1" starts an uncovered interval. Therefore, P is a bijection between consistent patterns in $\{-1,0,1\}^{n+1}$ and partial triangulations. The count $C(j)$ of all consistent patterns of length j is surprisingly interesting: Call $P(j)$ the number of patterns of length j not ending with a "−1" (*positive patterns*), and call $N(j)$ the number of patterns of length j ending with a "−1" (*negative patterns*). Then the consistency implies that $P(1)=1$, $N(1)=1$. In order to obtain a positive pattern, negative patterns can only be extended consistently by a "1", whereas positive patterns can be extended by a "0" or a "1". In order to obtain a negative pattern, any pattern can and must be extended by a "−1". Therefore:

$$P(j+1) = 2P(j)+N(j), \quad N(j+1) = P(j)+N(j). \tag{134}$$

Let now $F_1, F_2, F_3, F_4, \ldots$ be the Fibonacci series with $F_1 = F_2 = 1$ and $F_j = F_{j-1}+F_{j-2}$ for $j>2$. Define $P'(j) := F_{2j}$ and $N'(j) := F_{2j-1}$ with $P'(1) = F_2 = 1$ and $N'(1) = F_1 = 1$. Moreover:

$$P'(j+1) = F_{2j+2} = F_{2j+1} + F_{2j} = F_{2j} + F_{2j-1} + F_{2j} = 2P'(j) + N'(j), \tag{135}$$

$$N'(j+1) = F_{2j+1} = F_{2j} + F_{2j-1} = P'(j) + N'(j). \tag{136}$$

For all $j = 1,2,\ldots$ this proves that $P(j) = P'(j) = F_{2j}$ and $N(j) = N'(j) = F_{2j-1}$. Consequently, $C(j) = P(j)+N(j) = N(j+1) = F_{2j+1}$. By a straight-forward application of the Eigenvalue-method to the linear system of difference equations

$$\begin{pmatrix} P(j+1) \\ N(j+1) \end{pmatrix} = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} P(j) \\ N(j) \end{pmatrix} \tag{137}$$

for $P(j)$ and $N(j)$, the asymptotic growth of $C(n) = P(n)+N(n) = N(n+1)$ is exactly $(\frac{3+\sqrt{5}}{2})^n$.

A non-empty partial triangulation T is right-extendable if and only if its segment pattern $P(T)$ is consistent and has no subpattern "(−1)1". That is, right-extendable partial triangulations cover the line completely up to some right-most covered point. Hence, each subset of the points $1,2,\ldots,n+1$ gives rise to a right-extendable partial triangulation by using the maximal point as the right-most point covered by the partial triangulation (0 for the empty subset) and the remaining points to specify a complete triangulation of the line to the left of it. This induces a bijection between the set of right-extendable partial triangulations and the set of subsets of $\{1,2,\dots,n+1\}$, which has cardinality 2^{n+1}. Hence, the number of partial triangulations is, asymptotically, by an exponential factor $\left(\frac{3+\sqrt{5}}{2}\right)^n/2^{n+1} = \frac{1}{2}\left(\frac{3+\sqrt{5}}{4}\right)^n > \frac{1}{2}\left(\frac{5}{4}\right)^n$ larger than the number of right-extendable partial triangulations of $\mathbf{L}_n$.

Because of the given order of points from left to right, each "(−1)1" subpattern is detected immediately by lex-pruning: assume that the segment-pattern of an expandable but not-right-extendable partial triangulation T has the "−1" is in position $i-1$ and the "1" in position i for some $i \in \{1,\dots,n\}$. This means, the interval $[i-1,i]$ is uncovered, and the interval $[i,i+1]$ is covered by some simplex in T. Hence, T indexes a triangulation $\mathscr{T}$ that has $\{i\}$ as a free interior facet of one of its simplices $\{i,j\}$, $j>i$. The admissibles of T index, by definition, simplices that are all lex-larger than $\{i,j\}$ and intersecting properly with $\{i,j\}$. The lex-minimal admissible simplex for $\mathscr{T}$ is therefore lex-at-least $\{j,j+1\}$. The minimal facet of the minimal admissible simplex for $\mathscr{T}$ is lex-at-least $\{j\}$, which is lex-larger than the free interior facet $\{i\}$ of $\mathscr{T}$, which is equal to or lex-larger than the minimal free interior facet of $\mathscr{T}$. Consequently, T will be lex-pruned. Therefore, all the expandable but not right-extendable partial triangulations will be pruned by lex-pruning and, therefore, also by strong-pruning.

For item (iii), consider the d-dimensional regular cross polytope $\mathbf{C}^*_{d+1} = \left(\begin{smallmatrix}\mathbf{C}^*_d & \mathbf{O}_d & \mathbf{O}_d \\ 0 & 1 & -1\end{smallmatrix}\right)$ with $\mathbf{C}^*_1 = (1,-1)$ and $\mathbf{O}_d$ the origin in dimension d (coordinates may be homogenized by adding a row of ones throughout). From the fact that $\mathbf{C}^*_1$ is a segment and $\mathbf{C}^*_{d+1}$ is the one-point suspension of $\mathbf{C}^*_d$ at the origin for $d>0$ (see [8, Chp. 4]) the following can be easily derived by induction: $\mathbf{C}^*_d$ has $2d$ points in rank $d+1$ and 2^d facets. For each *antipodal diagonal* D between antipodal points $2k-1$ and $2k$, $k=1,2,\dots,d$, there is exactly one triangulation $T(D)$ using it. Each triangulation consists of 2^{d-1} many simplices. Each simplex in $T(D)$ has exactly two boundary facets (those not containing D) and $d-1$ interior facets (those containing D). The crucial property of any triangulation of a regular cross polytope is that each triangulation $T(D)$ is uniquely determined by any of its simplices, since all simplices in $T(D)$ contain D. Furthermore, only the simplices inside $T(D)$ intersect properly with each other, since all distinct antipodal diagonals intersect improperly by themselves (the origin is in the interior of each antipodal diagonal). Therefore, only the empty set (which is a partial triangulation right-extendable to any triangulation) and the $d2^{d-1}$ non-empty lex-subsets of one of its d many triangulations T can be right-extended, namely to T. In contrast to this, in total there are $d\left(2^{(2^{d-1})}-1\right)$ non-empty partial triangulations so that the total number of partial triangulations including the empty set is as claimed. Since any free interior facet f of a partial triangulation T contains some antipodal diagonal D, only simplices from $T(D)$ contain f. Because in $T(D)$ the interior facet f is contained in a unique other simplex s not in T, any multi-covering set of simplices $C(f)$ like in Theorem 11 must contain s. Therefore, such a multi-covering set exists if and only if T is a lex-subset of $T(D)$, i.e., if and only if T is right-extendable to $T(D)$. In other words, all not right-extendable partial triangulations can be pruned by strong-pruning. Note that there may be partial triangulations where the lex-min free interior facet is coverable by an admissible simplex, but not all free interior facets are. Thus, lex-pruning may miss some not right-extendable partial triangulations. (In experiments it can be seen that the number of misses is very small, though.)

For the constructions of the $\mathbf{P}_k$ in items (iv) and (v) proceed as follows. The order of points will not be relevant here, since in this particular case it can be proved that the partial triangulations involved cannot be extended at all, even if all so-far unused simplices could be used as expansions. For both items consider for a fixed $\ell \geq 3$ the point configuration $\mathbf{P} := \mathbf{P}(\ell)$ forming a prism over a regular ℓ-gon with 2ℓ points. For $i = 0, 1, \ldots, \ell - 1$ (all i-indices below are considered modulo ℓ), call the top points $\mathbf{v}_i$ (labelled by v_i), and call the bottom points $\mathbf{w}_i$ (labeled by w_i). Consider the cyclic set of diagonals $v_i w_{i+1}$ inside the ℓ quadrilateral facets of $\mathbf{P}$ labelled by F_i. These diagonals induce unique triangulations $T_i := \{v_i w_i w_{i+1}, v_i w_{i+1} v_{i+1}\}$ of all the F_i. From [26] it is known that $\mathbf{P}$ has no triangulation that uses $\bigcup_{i=0}^{\ell-1} T_i$.

Beyond each of the ℓ facets F_i, add $k+1$ distinct points labeled by $U_i := \{u_{i1}, \ldots, u_{ik}\}$ and u_i in the order of increasing distance to F_i on a line through the barycenter of F_i perpendicular to aff(F_i). This results in the point configuration $\mathbf{P}_k$ with $\ell(k+3)$ points in rank 4. Its convex hull consists of the prism over the regular ℓ-gon and ℓ pyramids over the ℓ quadrilateral facets with apices u_i.

Construct now partial triangulations triangulating the ℓ pyramids. To triangulate pyramid $F_i * u_i$, choose an arbitrary subset $V_i \subseteq U_i$ and place the points in order of increasing distance to F_i to generate a *placing triangulation* (see [8, Chp. 4]). This results in the triangulation, where the first point in V_i or u_i (if V_i is empty) is joined to T_i, and all the remaining points (including u_i) are joined to the visible boundary facets of the previous partial triangulation. For each pyramid, no two distinct subsets V_i lead to identical triangulations, since a point is a vertex of some simplex in the placing triangulation if and only if it is in V_i. Moreover, the various triangulations of the pyramids can be combined aribtrarily to a partial triangulation of all the pyramids $\bigcup_{i=1}^{\ell} (T_i * u_i)$. This results in $2^{\ell k}$ distinct partial triangulations. All these partial triangulations use the cyclic set of diagonals of the prism, i.e., none of them is extendable to a triangulation of $\mathbf{P}_k$. In particular, none of them is right-extendable.

Furthermore, all of them have $\bigcup_{i=0}^{\ell-1} T_i$ as their free interior facets. With the help of the Cayley-trick (see [12] for the general theory and [26] for the application to prims over ℓ-gons) the following can be eye-balled: for $\ell \leq 4$ not all free interior facets f can be covered by a covering set of simplices $C(f)$ as in Theorem 11, and for $\ell \geq 5$ the set of simplices $\bigcup_{i=0}^{k-1} \{v_i w_{i+1} v_{i+1} w_{i+2}\}$ is pairwise intersecting properly and covers all free interior facets. Thus, for $\ell = 4$ we obtain item (iv), and for $\ell = 5$ we obtain item (v). □

If $\mathfrak{G}$ is given as an explicit set of permutations, Theorem 13 shows that enumeration of triangulations with `SymLexSubsetRSFeas_withData` is polynomial in input size and the number of orbits of non-prunable subsets. This is because, in this case, the number of simplices is at most the number of triangulations, which is at most the number of triangulation orbits times the group order, which is at most the number of non-prunable subsets times the input size. The first example shows that this may be exponential in the input and output sizes in case $\mathfrak{G}$ is given by a set of generators, even if `IsLexMin` could be implemented in polynomial time. The second and third examples show that `SymLexSubsetRSFeas_withData` without pruning is for counting/enumeration/listing, in general, not polynomial in the input and output sizes. The third example, moreover, shows that `SymLexSubsetRSFeas_withData` with lex-breaking and strong- or lex-pruning for counting/enumeration cannot be polynomial in the input and output sizes because the exponentially many elements per found object all have to be touched by the algorithm. Whether `SymLexSubsetRSFeas_withData` with lex-breaking and strong- or lex-pruning is polynomial in the input and output sizes for listing when $\mathfrak{G}$ is given as an explicit set of permutations remains an open problem.

9.3 Results

$\mathbf{A}$	# sym's	# triangulations up to symmetry	# triangulations in total	# nodes	CPU time [hh:mm:ss]
$\mathbf{C}_4$	384	247,451	92,487,256	3,446,659	0:00:01
$\mathbf{\Delta}_6 \times \mathbf{\Delta}_2$	30,240	533,242	16,119,956,160	6,325,472	0:01:46
$\mathbf{\Delta}_4 \times \mathbf{\Delta}_3$	2880	7,402,421	21,316,106,880	116,083,390	0:02:31
*$\mathbf{\Delta}_5 \times \mathbf{\Delta}_3$	17280	25,606,173,722	442,472,050,753,920	429,725,338,124	1332:19:55
*$\mathbf{\Delta}(7,2)$	5040	37,676,752	189,355,661,460	1,397,621,560	0:47:15
*$\mathbf{\Delta}(6,3)$	1440	59,708,427	85,793,497,200	2,555,523,948	0:27:08
$3\mathbf{\Delta}_3$	24	925,148,763	22,201,684,367	7,154,329,212	0:10:56
$\mathbf{C}(14,8)$	28	2,429,751	68,007,706	187,209,582	0:00:23
$\mathbf{C}(19,14)$	38	6,515,385	247,567,074	1,265,333,660	0:04:09
$\mathbf{C}(16,3)$	2	58,492,955,941	116,985,744,91	887,659,233,813	17:16:57
*$\mathbf{C}(14,4)$	28	244,771,183	6,853,476,616	8,343,040,544	0:12:59
*$\mathbf{C}(15,4)$	30	18,845,509,142	565,365,033,880	650,520,069,380	18:59:44
*$\mathbf{C}(14,5)$	2	328,152,636,588	656,305,030,644	11,575,302,270,565	320:42:24
*$\mathbf{C}(14,6)$	28	8,314,337,199	232,797,963,456	535,970,897,965	18:10:42
*$\mathbf{C}(14,7)$	2	15,813,939,113	31,627,843,174	937,148,113,630	23:36:37
*$\mathbf{C}(15,9)$	2	1,397,895,884	2,795,741,709	117,124,014,923	3:21:53
*$\mathbf{C}(16,10)$	32	1,902,605,255	60,881,310,552	213,053,994,784	9:52:13
*icosahedron	120	95	8598	3813	0:00:00
*pseudoicosahedron	24	7701	182,670	147,775	0:00:00
*dodecahedron	120	12,775,757,027	1,533,079,037,570	125,333,463,449	4:58:15
*pyritohedron	24	1,363,918,758,719	32,734,029,351,118	13,786,801,148,394	381:32:27

Table 8: Computational results for the enumeration of triangulations using 16 threads (numbers with a "*" are new)

Table 8 presents the results obtained by applying `SymLexSubsetRSFeas_withData` with lex-breaking and lex-pruning to the enumeration of all triangulations of point configurations (see the end of Section 5 for explanations concerning the point configurations). Table 9 shows for three selected examples the CPU times compared to `mptopcom 1.4` (see [14, 15]). Note that, by design, in general `mptopcom` and `TOPCOM` do *not* compute the same thing: `mptopcom` computes *all subregular* triangulations, whereas `TOPCOM` computes *all* triangulations, for which there was *no* competitive software *at all* so far. In the listed examples, though, the numbers are known to coincide. The hypercube $\mathbf{C}_4$ was selected as an example with symmetry group of moderate order, the product of simplices $\mathbf{\Delta}_6 \times \mathbf{\Delta}_2$ was chosen as an example with a rather large symmetry group, and $\mathbf{C}(12,4)$ represents cyclic polytopes, for

A	CPU times [hh:mm:ss]			
	`mptopcom 1.4`		`TOPCOM 1.2.0b`	
	subregular	regular	all	regular
$\mathbf{C}_4$	0:01:46	0:02:41	0:00:01	0:00:54
$\mathbf{\Delta}_6 \times \mathbf{\Delta}_2$	0:09:25	0:13:10	0:01:46	0:05:36
$\mathbf{C}(12,4)$	0:00:21	-	0:00:01	-

Table 9: Comparison of CPU times with the so far fasted software `mptopcom 1.4` in [14] on small examples using 16 threads (the triangulations output was activated in both and then piped to `/dev/null` for a fair comparison); for these examples, all triangulations are subregular; since `mptopcom 1.4` does not support regularity checks for cyclic polytopes, the regular-triangulations timing was skipped for $\mathbf{C}(12,4)$

A	# sym's	# reg. triang's up to symmetry	# reg. triang's in total	CPU time all	CPU time regular
$\mathbf{C}_4$	384	235,277	87,959,448	0:00:01	0:00:54
$\mathbf{\Delta}(7,2)$	5040	30,485,496	153,209,697,210	0:47:15	8:42:33
$\mathbf{\Delta}(6,3)$	1440	42,489,025	61,035,863,100	0:27:08	18:10:37

Table 10: Comparison of CPU times for the enumeration of all versus regular triangulations using 16 threads for selected examples

which, first, the symmetry group is tiny, second, triangulations have many simplices (which is a stress-test for extension-based enumeration), and, finally, for which `mptopcom 1.4` uses a specialized (faster) implementation.

It can be seen that `TOPCOM` is particularly fast for the computation of all triangulations of $\mathbf{C}_4$, where exploring the enumeration tree is the dominant operation. For $\mathbf{\Delta}_6 \times \mathbf{\Delta}_2$ the symmetry handling is dominant, where `TOPCOM` is still several times faster, but not by the same margin. For the regularity checks, `cddlib` was used as an LP solver [11]. Since these regularity checks (besides being time-consuming) are advantageous for `mptopcom`'s flip-based enumeration and symmetry handling, the speed-up is less pronounced for the enumeration of regular triangulations.

Some numbers could be computed for the first time, to the best of my knowledge. The largest new examples are the number of triangulations of the pyritohedron (largest number up to symmetry) and the number of triangulations of $\mathbf{\Delta}_5 \times \mathbf{\Delta}_3$ (largest total number). Meanwhile, the numbers for three-dimensional cyclic polytopes can be computed much faster via the enumeration of persistent graphs [10].

Table 10 one can see how much time the regularity check takes compared to the mere enumeration of triangulations. For these results, again `cddlib` was used as an LP solver inside the regularity checks [11]. The numbers for the hypersimplices confirm the recently computed numbers from [5].

Remark 3. For all examples computed in this paper, it is not necessary to use a high-performance computing device; all numbers could be computed even faster by the eight performance cores of an M1Max laptop (see Section 6 for details on the computational environment).

9.4 Enhancements

For the enumeration of triangulations, extension-based algorithms can be equipped with some addititional functionality that is not easily incorporated in flip-based algorithms. A large part of this is based on the following observation.

Observation 1. *For any restriction on the triangulations to be enumerated that is a restriction on all simplices in the triangulation,* `SymLexSubsetRSFeas_withData` *can be run without modifications except that*

- *the set of considered simplex indices has to be restricted to the indices of the feasible simplices;*
- *the set of considered symmetries has to be restricted to the set-wise stabilizer subgroup of the set of feasible simplices.* □

This can be applied to full triangulations, i.e., triangulations that use all the points (exclude any simplex with points beyond its vertex set in its convex hull), unimodular triangulations (exclude any simplex with non-minimal volume), triangulations requiring a given face in each simplex (exclude any simplex not containing the face), triangulations avoiding some face alltogether (exclude any simplex containing the face), etc. In particular, this allows to compute all triangulations of the boundary of a point configuration, allthough such triangulations (topological spheres) do not even belong to the class of triangulations of point configurations (topological balls). A triangulation where all simplices contain a given point is called a *conical triangulation* with the given point as its apex. A conical triangulation where the apex is a relative interior point and all other used points are in the boundary is called *central.* The link of a central triangulation at its apex is then a boundary triangulation. Coning any boundary triangulation to a relative interior point yields a central triangulation. Thus, there is a bijection between central triangulations and boundary triangulations. In particular, central triangulations do not depend on the relative interior point that was used to construct them.

Observation 2. *Let* $\mathbf{A}$ *be a point configuration with symmetry group* $\mathfrak{G}$*. Then, all boundary triangulations of* $\mathbf{A}$ *can be computed up to symmetry as follows:*

(1) *Remove all points in the relative interior of* $\mathbf{A}$ *to obtain* $\mathbf{A}'$ *and let* $\mathfrak{G}'$ *be the restriction of* $\mathfrak{G}$ *to the remaining elements.*

(2) *Let* $\mathbf{b} := \frac{1}{|\mathbf{A}'|}\sum_{\mathbf{p}\in\mathbf{A}'}\mathbf{p}$ *be the barycenter of* $\mathbf{A}'$ *(or any other point in its relative interior) and add it with new label* 0 *to* $\mathbf{A}'$ *to obtain* $\mathbf{A}^{\text{central}}$.

(3) *Let* $\mathscr{S}^{\text{central}}$ *be the set of all simplices in* $\mathbf{A}^{\text{central}}$ *that contain point* 0*, and let* $\mathfrak{G}^{\text{central}}$ *be the set-wise stabilizer of* $\mathscr{S}^{\text{central}}$ *in* $\mathfrak{G}'$.

(4) *Apply* `SymLexSubsetRSFeas_withData` *to* $\mathbf{A}^{\text{central}}$ *with simplex set* $\mathscr{S}^{\text{central}}$ *and symmetries* $\mathfrak{S}^{\text{central}}$ *in order to compute all central triangulations of* $\mathbf{A}^{\text{central}}$ *up to symmetry.*

(5) *For each central triangulation of* $\mathbf{A}^{\text{central}}$ *compute the link of* 0 *by deleting the point* 0 *in all its simplices to obtain the corresponding boundary triangulation.* □

Before this will be applied to full root polytopes below, another type of restrictions has to be considered: prescribed symmetries. This is slightly more involved. Note that a triangulation can have symmetries that are not in the automorphism group of the point configuration, since, a-priori, a triangulation need not use all the points. All symmetries are in the automorphism group $\operatorname{Aut}\bigl(\operatorname{vert}(\mathbf{A})\bigr) \le \mathfrak{S}_n$ of the vertices $\operatorname{vert}(\mathbf{A})$ of $\mathbf{A}$, though.

Definition 18. Let $\mathbf{A}$ be a point configuration with n points and $\mathfrak{H}$ be a subgroup of $\operatorname{Aut}\bigl(\operatorname{vert}(\mathbf{A})\bigr)$. An element s indexes an $\mathfrak{H}$*-feasible simplex* if any two elements from the $\mathfrak{H}$-orbit of s index identical or properly intersecting simplices. Two elements s and s' *intersect* $\mathfrak{H}$*-properly* if any two elements from their respective $\mathfrak{H}$-orbits index identical or properly intersecting simplices. A triangulation of $\mathbf{A}$ indexed by T is $\mathfrak{H}$*-invariant*, if $\sigma(T) = T$ for all $\sigma \in \mathfrak{H}$. A symmetry π of $\mathbf{A}$ is $\mathfrak{H}$*-feasible* whenever for all $s \in [n]$ one has that s indexes an $\mathfrak{H}$-feasible simplex if and only if $\pi(s)$ indexes an $\mathfrak{H}$-feasible simplex.

With this notion, the following observation is the basis for the enumeration of triangulations with prescribed symmetries. It will be called a theorem because of its impact, although the proof is immediate.

Theorem 14. *Let* $\mathbf{A}$ *be a point configuration with* n *points, and let* $\mathfrak{H}$ *be a subgroup of* $\operatorname{Aut}\bigl(\operatorname{vert}(\mathbf{A})\bigr)$*. Let* T *be a non-empty subset of pairwise* $\mathfrak{H}$*-properly intersecting* $\mathfrak{H}$*-feasible simplices whose set of free interior facets is empty. Then* T *indexes an* $\mathfrak{H}$*-invariant triangulation of* $\mathbf{A}$.

Proof. Let T be as in the assumption. Since proper $\mathfrak{H}$-intersection implies in particular proper intersection, T indexes a partial triangulation. Since it has no interior free interior facets, it indexes a triangulation. For $\sigma \in \mathfrak{H}$ and $s \in T$, by $\mathfrak{H}$-feasibility and $\mathfrak{H}$-proper intersection, the image $\sigma(s)$ indexes a simplex that is identical to or intersects properly with all simplices in T, in particular with the simplex indexed by s. Since any triangulation is an inclusion-maximal set of properly intersecting simplices, $\sigma(s)$ is contained in T. Applied to all $s \in T$ this implies that $\sigma(T) \subseteq T$. Since any $\sigma \in \mathfrak{H}$, as a permutation, is a bijection, $\sigma(T) = T$ must hold. Since σ was chosen arbitrarily in $\mathfrak{H}$, the claim is proved. □

That means: Once the bijection indexing simplices has been restricted to $\mathfrak{H}$-feasible elements, the symmetries have been restricted to $\mathfrak{H}$-feasible symmetries, and the admissibles table of the point configuration has been modified to represent $\mathfrak{H}$-proper intersection, then `SymLexSubsetRSFeas_withData` will automatically enumerate only $\mathfrak{H}$-invariant triangulations up to $\mathfrak{H}$-feasible symmetries.

Table 11 compares the numbers of nodes and triangulations for various point configurations with and without certain restrictions. Prescribing symmetries is particularly effective.

$\mathbf{A}$	restriction	# feas. symm's	# triang's up to symm.	# triang's in total	# nodes
$\mathbf{C}_4$	none	384	247,451	92,487,256	3,446,659
$\mathbf{C}_4$	unimodular	384	159,037	59,546,240	2,375,773
*$\mathbf{C}_4$	$\mathbb{Z}_2$-inv.	384	181	22,280	12,884
*$\mathbf{C}_4$	$\mathbb{Z}_2$-inv./unim.	384	154	19,520	11,414
*pyrit.	none	24	1,363,918,758,719	32,734,029,351,118	13,786,801,148,394
*pyrit.	$\mathbb{Z}_2$-inv.	24	1,313,581	15,761,630	80,245,911
*pyrit.	$\mathbb{Z}_3$-inv.	6	7968	15,889	546,939
$3\mathbf{\Delta}_3$	none	24	925,148,763	22,201,684,367	7,154,329,211
$3\mathbf{\Delta}_3$	full	24	21,302,400	511,052,427	316,591,002
$3\mathbf{\Delta}_3$	unimodular	24	14,459,488	346,903,379	207,932,285
*$3\mathbf{\Delta}_3$	$\mathbb{Z}_4$-inv.	8	98	181	4795
*$3\mathbf{\Delta}_3$	$\mathbb{Z}_4$-inv./full	8	36	65	2365
*$3\mathbf{\Delta}_3$	$\mathbb{Z}_4$-inv./unim.	8	18	33	1762
*$3\mathbf{\Delta}_3$	$\mathfrak{S}_4$-inv.	24	3	3	100
*$3\mathbf{\Delta}_3$	$\mathfrak{S}_4$-inv./full	24	1	1	74
*$3\mathbf{\Delta}_3$	$\mathfrak{S}_4$-inv./unim.	24	0	0	29
$4\mathbf{\Delta}_3$	none	24	?	?	?
$4\mathbf{\Delta}_3$	full	24	?	?	?
$4\mathbf{\Delta}_3$	unimodular	24	?	?	?
*$4\mathbf{\Delta}_3$	$\mathbb{Z}_4$-inv.	8	836,982	1,670,895	73,894,796
*$4\mathbf{\Delta}_3$	$\mathbb{Z}_4$-inv./full	8	108,103	215,479	11,807,847
*$4\mathbf{\Delta}_3$	$\mathbb{Z}_4$-inv./unim.	8	79,147	157,724	8,862,008
*$4\mathbf{\Delta}_3$	$\mathfrak{S}_4$-inv.	24	12	12	1182
*$4\mathbf{\Delta}_3$	$\mathfrak{S}_4$-inv./full	24	5	5	785
*$4\mathbf{\Delta}_3$	$\mathfrak{S}_4$-inv./unim.	24	3	3	543
$3\mathbf{\Delta}_4$	none	120	?	?	?
$3\mathbf{\Delta}_4$	full	120	?	?	?
$3\mathbf{\Delta}_4$	unimodular	120	?	?	?
*$3\mathbf{\Delta}_4$	$\mathbb{Z}_5$-inv.	20	43,882	175,441	6,071,031
*$3\mathbf{\Delta}_4$	$\mathbb{Z}_5$-inv./full	20	15,841	63,306	2,436,943
*$3\mathbf{\Delta}_4$	$\mathbb{Z}_5$-inv./unim.	20	7720	30,832	1,345,947
*$3\mathbf{\Delta}_4$	$\mathfrak{S}_5$-inv.	120	1	1	56
*$3\mathbf{\Delta}_4$	$\mathfrak{S}_5$-inv./full	120	0	0	45
*$3\mathbf{\Delta}_4$	$\mathfrak{S}_5$-inv./unim.	120	0	0	45
$\mathbf{\Delta}_4\times\mathbf{\Delta}_4$	none	28,800	?	?	?
*$\mathbf{\Delta}_4\times\mathbf{\Delta}_4$	$\mathbb{Z}_5$-inv.	200	317	9630	61,039
*$\mathbf{\Delta}_4\times\mathbf{\Delta}_4$	$\mathbb{Z}_2$-inv.	240	30,327,170	3,638,732,520	2,016,686,741

Table 11: Computational results for the enumeration of triangulations using 16 threads without and with restrictions on the feasibility of a triangulation (numbers with a "*" are new) to demonstrate the reduction of effort as indicated by the no. of nodes; since $\mathbf{C}_4$, the pyritohedron, and $\mathbf{\Delta}_4\times\mathbf{\Delta}_4$ are in strictly convex position, all triangulations are full; for $\mathbf{\Delta}_4\times\mathbf{\Delta}_4$ as a product of simplices all triangulations are unimodular

Consider, for example, the dilated tetrahedron $3\mathbf{\Delta}_3$ with $\mathfrak{S}_4$-symmetry when the enumeration is restricted to $\mathfrak{S}_4$-invariant triangulations. Prior to the new method in this paper, the only way to deal with prescribed symmetries was to inspect all triangulations and filter by symmetry in post-processing. In other words, in order to count all $\mathfrak{S}_4$-invariant triangulations of $3\mathbf{\Delta}_3$ up to $\mathfrak{S}_4$-feasible symmetry, it was necessary to generate all triangulations up to symmetry at some point in the process. According to Table 8, for the enumeration method in this paper (which is the fastest known so far) this takes 7,154,329,212 enumeration nodes. Now, with the new method based on $\mathfrak{S}_4$-feasible simplices that intersect $\mathfrak{S}_4$-properly, computing the three $\mathfrak{S}_4$-invariant triangulations up to $\mathfrak{S}_4$-feasible symmetry (in this case, all 24 elements of $\mathfrak{S}_4$ are $\mathfrak{S}_4$-feasible) takes only 100 enumeration nodes. And with 79 nodes the unique $\mathfrak{S}_4$-invariant triangulation using all the lattice points in $3\mathbf{\Delta}_3$ is found. Finally, it takes only 29 enumeration nodes to prove computationally that there is no unimodular $\mathfrak{S}_4$-invariant triangulation of $3\mathbf{\Delta}_3$. The enumeration effort is, therefore, completely dominated by preprocessing. While the case of $3\mathbf{\Delta}_3$ the post-processing approach would still be viable (though inefficient), things are different, e.g., for $4\mathbf{\Delta}_3$: an enumeration of all triangulations seems currently out of reach. Still, the new method can compute in seconds the triangulations with full symmetry. This time, three unimodular $\mathfrak{S}_4$-invariant triangulations exist. As an example for a less restricting prescribed symmetry group, the $\mathbb{Z}_4$-invariant triangulations (induced by cyclic permutations of coordinates) have been computed as well in essentially a minute.

A	restriction	# feas. symm's	# triang's up to symm.	# triang's in total	# nodes
$\mathbf{P}(A_2)$	none	12	8	32	100
$\mathbf{P}(A_2)$	central	12	1	1	29
$\mathbf{P}(A_2)$	central/$\mathbb{Z}_2$-inv.	12	1	1	26
$\mathbf{P}(A_3)$	none	48	1843	79,884	51,039
$\mathbf{P}(A_3)$	central	48	7	64	557
$\mathbf{P}(A_3)$	central/$\mathbb{Z}_2$-inv.	48	2	8	258
$\mathbf{P}(A_4)$	none	240	32,483,441,808	7,795,598,797,008	1,438,773,642,274
$\mathbf{P}(A_4)$	central	240	15,264	3,523,506	2,776,349
$\mathbf{P}(A_4)$	central/$\mathbb{Z}_2$-inv.	240	20	1782	12,950
$\mathbf{P}(A_5)$	none	1440	?	?	?
$\mathbf{P}(A_5)$	central	1440	?	?	?
*$\mathbf{P}(A_5)$	central/$\mathbb{Z}_2$-inv.	1440	112,234	79,216,008	183,816,883

Table 12: Computational results for the enumeration of all/central/$\mathbb{Z}_2$-invariant triangulations of full root polytopes using 16 threads (the number with a "*" is new, the numbers of $\mathbb{Z}_2$-invariant central triangulations up to $\mathbf{P}(A_4)$ have been computed in [9] by a completely different method before and have been confirmed by this computation)

My original motivation to compute triangulations with prescribed symmetries came from the n-dimensional full root polytopes $\mathbf{P}(A_n)$ in ambient $n+1$-space, where the centrally symmetric central triangulations have a special meaning [9]. Up to now, prior to the result in this paper no numbers for $\mathbf{P}(A_5)$ have been published, restricted or not. *Nota bene:* In [9] it was already reported that $\mathbf{P}(A_5)$ has 25,224 regular central and centrally symmetric triangulations. This was based on preliminary results from this paper, though.

Table 12 shows the all the results up to $n = 5$. Using the new method in this paper the number 112,234 of central and centrally symmetric ($\mathbb{Z}_2$-invariant) triangulations of $\mathbf{P}(A_5)$ could be computed up to $\mathbb{Z}_2$-feasible symmetries for the first time (in about one hour). Again, all symmetries of $\mathbf{P}(A_n)$ are $\mathbb{Z}_2$-feasible. Here, the post-processing approach would have been way more time-consuming: A several-weeks-long partial enumeration of the central triangulations of $\mathbf{P}(A_5)$ showed that there are more than 1,799,917,616 symmetry classes of central triangulations (2,591,706,000,744 in total using 1,130,442,682,392 enumeration nodes).

The smallest instance for products of simplices with unknown number of all triangulations is $\mathbf{\Delta}_4 \times \mathbf{\Delta}_4$. A several-months-long partial enumeration with several interruptions with checkpointing on up to 192 threads showed that there are more than 365,127,837,881 symmetry classes of triangulations (10,515,476,512,745,280 in total using 7,431,758,631,675 enumeration nodes). In contrast to this, it can now be computed for the first time in less than a minute that there are, e.g., 317 triangulations with $\mathbb{Z}_5$-symmetry (i.e., cyclic symmetry of order 5) up to the $\mathbb{Z}_5$-feasible symmetries of order 200. For the vertices v_i, v_j of $\mathbf{\Delta}_4$, these symmetries are generated by the permutation $(v_i, v_j) \mapsto (v_{i+1}, v_{j+1})$ for $i, j = 0, 1, \ldots, 4$ considered modulo 5. This took no more than 61,039 enumeration nodes. The total number of such $\mathbb{Z}_5$-invariant triangulations is 9630. The results for $\mathbb{Z}_2$-symmetries induced by $(v_i, v_j) \mapsto (v_j, v_i)$ in $\mathbf{\Delta}_4 \times \mathbf{\Delta}_4$: There are 30,327,170 $\mathbb{Z}_2$-invariant triangulations up to the $\mathbb{Z}_2$-feasible symmetries of order 240. The total number of these is 3,638,732,520. This took 2,016,686,741 enumeration nodes (which is perfectly tractable).

Another enhancement stems from the combination of extension-based enumeration and flip-graph exploration in one go. For the following, recall that a triangulation is *subregular,* if it can be flipped to a regular triangulation by a sequence of GKZ-increasing flips (*upflips*). Call it *non-subregular* otherwise. Call a triangulation a *regular-component triangulation (rc-triangulation)* if can be flipped to a regular triangulation by a sequence of arbitrary flips. Call it a *non-regular-component triagulation (nrc-triangulation)* otherwise.

In order to count all non-subregular triangulations, proceed as follows:

- Enumerate all triangulations by extension.
- For each non-regular triangulation found, search for a regular triangulation in the flip-graph by a depth-first search along upflips dropping all restrictions and ignoring symmetry.
- Count all non-regular triangulations that cannot be upflipped to a regular triangulation this way.

Similarly, in order to count all nrc-triangulations, proceed as follows:

- Enumerate all triangulations by extension.

- For each non-regular triangulation found, search for a regular triangulation in its flip-graph component dropping all restrictions. In order to heuristically find a regular triangulation more quickly, e.g., use the following heuristic, where symmetries are ignored except in the final step:
  - dive along flips that lexicographically increase the triangulation;
  - dive along flips that lexicographically decrease the triangulation;
  - depth-first-search along GKZ-increasing flips;
  - depth-first-search along GKZ-decreasing flips;
  - breadth-first-search along all flips to obtain a conclusive answer.
- Count all non-regular triangulations that could not be flipped to a regular triangulation at all.

This has been applied to the following configurations: $3\mathbf{\Delta}_3$ (where it is known that all triangulations are subregular from combining the results in [14] and this paper), $4\mathbf{\Delta}_3$, $3\mathbf{\Delta}_4$, the pyritohedron, and $\mathbf{\Delta}_4 \times \mathbf{\Delta}_4$. The latter example was of special interest because of the following: It is known that $\mathbf{\Delta}_k \times \mathbf{\Delta}_\ell$ has a connected flip-graph for all $k \leq 3$ [21] and that $\mathbf{\Delta}_4 \times \mathbf{\Delta}_\ell$ has a non-connected flip-graph for ℓ sufficiently large [20] (by a non-constructive probabilistic proof for $\ell \approx 4 \cdot 10^4$). The result does not exclude that $\mathbf{\Delta}_4 \times \mathbf{\Delta}_4$ has a non-connected flip-graph. In order to make computations tractable, cyclic symmetries have been prescribed throughout. Cyclic symmetries were chosen, since many famous non-regular triangulations exhibit cyclic symmetries in one way or another.

Table 13 show the results with some details concerning the effort in terms of the number of regularity checks (which is the dominating factor as soon as it is applied). The most striking result: For the first time, to the best of my knowledge, non-subregular triangulations have been found, namely in $4\mathbf{\Delta}_3$ and in $3\mathbf{\Delta}_4$. On the other hand, all of the non-subregular triangulations are rc (they can actually all be flipped to a regular triangulation by only GKZ-increasing or only GKZ-decreasing flips). Together this means: `mptopcom 1.4` on this input with the given order of points (see the end of Section 6) would inevitably miss out on some rc-triangulations of $4\mathbf{\Delta}_3$ and $3\mathbf{\Delta}_4$. Note that without prescribed symmetries the computational approach above would have been intractable for $4\mathbf{\Delta}_3$, $3\mathbf{\Delta}_4$ and $\mathbf{\Delta}_4 \times \mathbf{\Delta}_4$ given today's computational power.

The following prototypical theorem summarizes the finding that demonstrate the potential of the method, in the spirit of [19].

Theorem 15. *There are* 317 $\mathbb{Z}_5$*-invariant triangulations of* $\mathbf{\Delta}_4 \times \mathbf{\Delta}_4$ *up to* $\mathbb{Z}_5$*-feasible symmetry (*9630 *in total);* 247 *of them are non-regular (*8260 *in total). All these triangulations are connected to the flip-graph component of the regular triangulations by GKZ-increasing flips. In particular, they are all subregular.*

There are 7968 $\mathbb{Z}_3$*-invariant triangulations of the pyritohedron up to* $\mathbb{Z}_3$*-feasible symmetry (*15,889 *in total);* 2222 *of them are non-regular (*4430 *in total). All these triangulations are connected to the flip-graph component of the regular triangulations by GKZ-increasing flips. In particular, they are all subregular.*

A	restriction	# feas. symm's	# triang's up to symm.	# regularity checks	cpu time [hh:mm:ss]
*pyrit.	$\mathbb{Z}_2$-inv./non-regular	24	157,909	1,313,581	0:14:08
*pyrit.	$\mathbb{Z}_2$-inv./non-subreg.	24	0	3,960,419	0:40:38
*pyrit.	$\mathbb{Z}_2$-inv./nrc	24	0	4,308,087	0:46:29
*pyrit.	$\mathbb{Z}_3$-inv./non-regular	6	2222	7968	0:00:05
*pyrit.	$\mathbb{Z}_3$-inv./non-subreg.	6	0	34,577	0:00:19
*pyrit.	$\mathbb{Z}_3$-inv./nrc	6	0	33,805	0:00:19
$3\mathbf{\Delta}_3$	$\mathbb{Z}_4$-inv./non-regular	8	24	98	0:00:29
$3\mathbf{\Delta}_3$	$\mathbb{Z}_4$-inv./non-subreg.	8	0	259	0:00:41
$3\mathbf{\Delta}_3$	$\mathbb{Z}_4$-inv./nrc	8	0	274	0:00:47
*$4\mathbf{\Delta}_3$	$\mathbb{Z}_4$-inv./non-regular	8	580,117	836,982	0:48:40
*$4\mathbf{\Delta}_3$	$\mathbb{Z}_4$-inv./non-subreg.	8	2507	11,722,197	8:58:14
*$4\mathbf{\Delta}_3$	$\mathbb{Z}_4$-inv./nrc	8	0	16,223,896	14:40:39
*$3\mathbf{\Delta}_4$	$\mathbb{Z}_5$-inv./non-regular	20	35,367	43,882	0:17:58
*$3\mathbf{\Delta}_4$	$\mathbb{Z}_5$-inv./non-subreg.	20	58	1,211,612	1:28:56
*$3\mathbf{\Delta}_4$	$\mathbb{Z}_5$-inv./nrc	20	0	1,467,289	1:54:42
*$\mathbf{\Delta}_4 \times \mathbf{\Delta}_4$	$\mathbb{Z}_5$-inv./non-regular	200	247	317	0:03:50
*$\mathbf{\Delta}_4 \times \mathbf{\Delta}_4$	$\mathbb{Z}_5$-inv./non-subreg.	200	0	5817	0:04:39
*$\mathbf{\Delta}_4 \times \mathbf{\Delta}_4$	$\mathbb{Z}_5$-inv./nrc	200	0	6369	0:04:40

Table 13: Computational results for triangulations with special symmetries that were checked for non-regularity, non-subregularity and flip-graph connectivity to the component of the regular triangulations ("nrc" = non-regular-component triangulation; numbers with a "*" are new)

There are 1,313,581 $\mathbb{Z}_2$*-invariant triangulations of the pyritohedron up to* $\mathbb{Z}_2$*-feasible symmetry (*15,761,630 *in total);* 157,909 *of them are non-regular (*1,894,796 *in total). All these triangulations are connected to the flip-graph component of the regular triangulations by GKZ-increasing flips. In particular, they are all subregular.*

There are 836,982 $\mathbb{Z}_4$*-invariant triangulations of* $4\mathbf{\Delta}_3$ *up to* $\mathbb{Z}_4$*-feasible symmetry (*1,670,895 *in total);* 580,117 *of them are non-regular (*1,159,626 *in total), and* 2507 *of them are non-subregular (*5014 *in total). All these triangulations are connected to the flip-graph component of the regular triangulations (some by GKZ-increasing and some by GKZ-decreasing flips).*

There are 43,882 $\mathbb{Z}_5$*-invariant triangulations of* $3\mathbf{\Delta}_4$ *up to* $\mathbb{Z}_5$*-feasible symmetry (*175,441 *in total);* 35,367 *of them are non-regular (*141,442 *in total), and* 58 *of them are non-subregular (*232 *in total). All these triangulations are connected to the flip-graph component of the regular triangulations (some by GKZ-increasing and some by GKZ-decreasing flips).* □

To the best of my knowledge, a computation of the numbers with prescribed symmetries has not even been considered with the methods known prior to this paper, most likely for the following reason: For all known flip-based algorithms prescribed symmetries have no chance to reduce the effort of enumeration, since the flip-graph of these triangulations is, in general, not connected, not even for the regular triangulations: just consider $2\mathbf{\Delta}_2$ and its two symmetry classes of regular triangulations with 120-degree rotational symmetry with one and four simplices, respectively. Since their cardinalities differ, they can not be equivalent w.r.t. to the symmetries of $2\mathbf{\Delta}_2$. Moreover, they are not connected by a flip. This marks a systematic advantage of extension-based enumeration algorithms like SymLexSubsetRSFeas_withData.

A	regularity checks ...					
	... in triangulations only			... in all nodes		
	# nodes	# reg. checks	CPU time [hh:mm:ss]	# nodes	# reg. checks	CPU time [hh:mm:ss]
$\mathbf{C}_4$	3,446,659	247,451	0:00:54	3,382,448	1,791,034	0:04:59
$(\mathbf{C}_4)^*$	18,316,313	75,756	0:01:38	192,448	79,266	0:01:11

Table 14: Regularity checks in each node versus regularity checks only for triangulations on the 4-cube (235,277 regular out of 247,451 triangulations) and its Gale dual (490 regular out of 75,756 triangulations)

Regularity checks can reduce the enumeration effort of extension-based algorithms if regularity is checked for each partial triangulation. Because regularity checks are quite expensive compared to the enumeration operations, the benefits are outweighed by the effort for the examples seen so far. However, there are examples where the early regularity checks pay off. One such example is the enumeration of regular triangulations of the (totally cyclic) vector configuration corresponding to a Gale dual of the 4-cube, as can be seen in Table 14. There, the number of necessary regularity checks is only ever-so-slightly increased by checking each node. Since for this example many triangulations are non-regular, the resulting reduced effort in the enumeration tree leads to shorter CPU times. Note that for these early regularity checks there is a warm-start opportunity if the LPs are stored withing the local auxiliary data of a node. This would speed-up the regularity checks in each node. Since TOPCOM's default LP solver from cddlib does not support warm-starts yet, TOPCOM does not yet support warm-starts either.

In contrast to most of the other mentioned restrictions, requiring regularity is known to reduce the effort for flip-based algorithms as well, since the exploration of the flip-graph can be restricted to the flip-graph of regular triangulations, which is connected, in contrast to the flip-graph of all triangulations [8, Chapter 5]. The flip-graph of all regular conical (and, thus, central) triangulations of a d-dimensional point configuration $\mathbf{A}$ with n points is connected, too: if the apex point is the first point, then the central triangulations form the face of the secondary polytope induced by the supporting hyperplane $\{z \in \mathbb{R}^n : z_1 = \mathrm{vol}_d(\mathrm{conv}\mathbf{A})\}$. Moreover, any face of the secondary polytope is a polytope itself. Thus, it has a connected edge graph, and each edge in this graph corresponds to a flip [8, Chapter 5].

Finally, one can use the methods in this paper to design optimization algorithms as follows. Any variant of algorithm SymLexSubsetRS – as any enumeration algorithm – can be turned into a branch-and-bound optimization algorithm by specifying an objective function and a dual-bound procedure. This was used to compute the minimal number of simplices in a triangulation for some examples. Details are omitted, since the method is straight-forward. Example results are: a minimal triangulation of the product of a square and a triangle has 10 simplices (a result originally proved in [30] with quite some effort, since in that paper more general dissections are allowed), which took less than a tenth of a second; a minimal triangulation of the 4-cube has 16 simplices, which took less than half a second; a minimal triangulation of the regular dodecahedron has 23 simplices, which took less than 10 minutes; a minimal triangulation of the pyritohedron has 23 simplices, too, which took less than 4 hours. And a minimal unrestricted triangulation of the full, centered root polytope in ambient 5-space has 46 simplices, which took less than 11 hours, whereas the enumeration of all its triangulations took around 95 hours. A minimal centrally symmetric central triangulation of it has 70 simplices, which took half a second. All computation times for optimization are significantly shorter than those for the complete enumeration. In branch-and-bound, the exploitation of symmetries is especially vital, since none of the equivalent branches leading to an optimal leaf can ever be pruned.

9.5 Notes on the Use of GKZ Vectors

The switch-table method to compute canonical representatives from [14] is particularly efficient if it can be based on a representation of triangulations by the so-called *GKZ-vector* (Gelfand-Kapranov-Zelevinsky vector, see [8, Chapter 5]). The GKZ-vector is a vector with a component for each point in the configuration. Such vectors can be compared lexicographically like subsets of simplex indices. Subsets of simplices (as in this paper) can be interpreted as characteristic vectors with as many components as there are simplices. While each triangulation can be represented by its characteristic vector, each regular triangulation can be uniquely represented by its GKZ-vector. Since, in general, the number of points is much smaller than the number of simplices spanned by the points, the representation by GKZ-vectors is significantly more compact than the representation as characteristic vectors. Moreover, when using switch tables the lexicographic comparison of two GKZ-vectors can be much faster than the lexicographic comparison of two characteristic vectors. Thus, one may ask whether the lexmin-check in `SymLexSubsetRSFeas_withData` could be accelerated by utilizing GKZ-vectors, at least for the enumeration of regular triangulations. The answer is no, at least not in any straight-forward way.

The reason is the basic justification of `SymLexSubsetRS`: a suitable equivalent of Lemma 2 does not hold in any straight-forward variant for canonicals based on the order of GKZ-vectors. Here is why. The inspection of the subset poset of partial triangulations of a square with the points ordered as

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \tag{138}$$

shows that picking the lexmax GKZ-vector as the canonical representative would result in $\{124, 134\}$ with GKZ-vector $(2, 1, 1, 2)$ being canonical. Note that the symmetries of the square act transitively on points, simplices, and triangulations. However, neither the subset $\{124\}$ with GKZ-vector $(1, 1, 0, 1)$ nor the subset $\{134\}$ with GKZ-vector $(1, 0, 1, 1)$ would be canonical, since only $\{1, 2, 3\}$ with GKZ-vector $(1, 1, 1, 0)$ is canonical. Thus, there is no way to sort the simplices such that an equivalent of Lemma 2 holds with a canonical defined by the lexmax GKZ-vector.

Picking instead the lexmin GKZ-vector as the canonical representative and sorting the simplices by increasing GKZ-vector is, therefore, the only way to make an equivalent of Lemma 2 true for this point configuration. Now consider the following six-point configuration whose symmetries are horizontal and vertical reflection as well as rotation by 180 degrees.

$$\mathbf{A}' = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 2 & 2 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \tag{139}$$

For the partial triangulation $\mathscr{T} = \{356, 345, 236\}$ the GKZ-vector is $(0, 1, 3, 1, 2, 2)$, which is lexmin (= canonical) in its orbit. Removing the simplex 236 with the lexmax GKZ-vector leads to the partial triangulation $\{356, 345\}$ with GKZ-vector $(0, 0, 2, 1, 2, 1)$, which is *not* lexmin in its orbit, since by horizontal reflection the partial triangulation $\{456, 346\}$ is equivalent and has the lexsmaller GKZ-vector $(0, 0, 1, 2, 1, 2)$.

Thus, no straight-forward equivalent of Lemma 2 holds true, and, hence, there is no correct `SymLexSubsetRS` available in general based on GKZ-vectors.

10 Conclusions

Variants of the generic algorithm `SymLexSubsetRS` have been introduced to enumerate maximal, co-minimal, and feasible subsets of a finite set up to symmetry. New versions of lex-minimality checks and new pruning methods for partial cocircuits and partial triangulations have been presented and analyzed. The new methods allowed for the computation of many new cardinalities, among them the number, up to symmetry, of cocircuits of the 9-cube, the number of circuits of the 8-cube, and the number of all triangulations of the pyritohedron, the dodecahedron, and $\mathbf{\Delta}_5 \times \mathbf{\Delta}_3$.

The algorithm `SymLexSubsetRS` can be enhanced. First, some types of restrictions imposed on the objects to be enumerated can reduce the enumeration effort by directly invalidating branches in the enumeration tree. This was shown for triangulations, most notably with prescribed symmetry, where restricted enumeration problems for, e.g., full root polytopes could be solved that would have been intractable without the restrictions, at the time being. Second, an objective function and a dual-bound procedure together with `SymLexSubsetRS` generate a straight-forward optimization algorithm. This was applied to find triangulations with a minimal number of simplices for the 4-cube and the regular dodecahedron in much less time than the enumeration takes. The efficiency of such an optimization algorithm will depend on the quality of the dual-bound procedure. It would be interesting to see where such an approach would be competitive to other special optimization algorithms (like the universal-polytope approach [7] for the minimal triangulation).

The new methods should have many more applications beyond the three applications in this paper like the enumeration of maximal cliques in a graph up to symmetry. This will be subject of further research.

References

[1] Oswin Aichholzer and Franz Aurenhammer. Classifying hyperplanes in hypercubes. *SIAM Journal on Discrete Mathematics*, 9(2):225–232, 1996.

[2] David Avis and Komei Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 65(1):21 – 46, 1996. First International Colloquium on Graphs and Optimization.

[3] David Avis and Charles Jordan. mts: a light framework for parallelizing tree search codes. *Optimization Methods and Software*, 36(2-3):279–300, May 2021.

[4] Anders Björner, Michel Las Vergnas, Bernd Sturmfels, Neil White, and Günter M. Ziegler. *Oriented Matroids*. Cambridge University Press, Cambridge, 2 edition, 1999.

[5] Laura Casabella, Michael Joswig, and Lars Kastner. Subdivisions of hypersimplices: with a view toward finite metric spaces. Preprint 2402.17665, arXiv, 2024.

[6] Jesús A. de Loera. *Triangulations of Polytopes and Computational Algebra*. PhD thesis, Cornell University, 1995.

[7] Jesús A. de Loera, Serkan Hoşten, Francisco Santos, and Bernd Sturmfels. The polytope of all triangulations of a point configuration. *Documenta Mathematika*, 1:103–119, 1996.

[8] Jesús A. de Loera, Jörg Rambau, and Francisco Santos. *Triangulations – Structures for Applications and Algorithms*, volume 25 of *Algorithms and Computation in Mathematics*. Springer, Berlin, Heidelberg, 2010.

[9] Emanuele Delucchi, Lukas Kühne, and Leonie Mühlherr. Combinatorial invariants of finite metric spaces and the Wasserstein arrangement. Preprint arXiv:2408.15584, arXiv, 2024.

[10] Vincent Froese and Malte Renken. Persistent graphs and cyclic polytope triangulations. *Combinatorica*, 41(3):407–423, 2021.

[11] Komei Fukuda. Cddlib reference manual 0.93a. Technical report, McGill University, Montréal, Quebec, Canada, 2003.

[12] Birkett Huber, Jörg Rambau, and Francisco Santos. The Cayley trick, lifting subdivisions, and the Bohne-Dress theorem on zonotopal tilings. *Journal of the European Mathematical Society*, 2:179–198, 2000.

[13] Hiroshi Imai, Tomonari Masada, Fumihiko Takeuchi, and Keiko Imai. Enumerating triangulations in general dimensions. *International Journal of Computational Geometry & Applications*, 12(06):455–480, 2002.

[14] Charles Jordan, Michael Joswig, and Lars Kastner. Parallel enumeration of triangulations. *The electronic journal of combinatorics*, 25(3):P3.6, 2018.

[15] Michael Joswig and Lars Kastner. New counts for the number of triangulations of cyclic polytopes. In James H. Davenport, Manuel Kauers, George Labahn, and Josef Urban, editors, *Mathematical Software – ICMS 2018*, pages 264–271, Berlin, Heidelberg, 2018. Springer.

[16] Michael Joswig and Benjamin Schröter. Parametric shortest-path algorithms via tropical geometry. *Mathematics of OR*, 47(3):2065–2081, February 2022.

[17] L. Khachiyan, E. Boros, K. Elbassioni, V. Gurvich, and K. Makino. On the complexity of some enumeration problems for matroids. *SIAM Journal on Discrete Mathematics*, 19(4):966–984, 2005.

[18] Miriam Kießling, Sascha Kurz, and Jörg Rambau. An exact column-generation approach for the lot-type design problem. *TOP*, 29(3):741–780, 2021.

[19] Earl S. Kramer and Dale M. Mesner. t-designs on hypergraphs. *Discrete Mathematics*, 15(3):263–296, 1976.

[20] Gaku Liu. A zonotope and a product of two simplices with disconnected flip graphs. *Discrete & Computational Geometry*, 59(4):810–842, 2018.

[21] Gaku Liu. Flip-connectivity of triangulations of the product of a tetrahedron and simplex. *Discrete & Computational Geometry*, 63(1):1–30, 2020.

[22] Arnaud Mary and Yann Strozecki. Efficient enumeration of solutions produced by closure operations. *CoRR*, abs/1712.03714, 2019.

[23] E. Minieka. Finding the circuits of a matroid. *Journal of Research of the National Bureau of Standards, Section B: Mathematical Sciences*, page 337, 1976.

[24] Christian Pech and Sven Reichard. Enumerating set orbits. In Mikhail Klin, Gareth A. Jones, Aleksandar Jurišić, Mikhail Muzychuk, and Ilia Ponomarenko, editors, *Algorithmic Algebraic Combinatorics and Gröbner Bases*, pages 137–150. Springer, Berlin, Heidelberg, 2009.

[25] Jörg Rambau. TOPCOM: Triangulations of point configurations and oriented matroids. In *Proceedings of the International Congress of Mathematical Software*, 2002.

[26] Jörg Rambau. On a generalization of Schönhardt's polyhedron. In Jacob E. Goodman, János Pach, and Emo Welzl, editors, *Combinatorial and Computational Geometry*, volume 52 of *MSRI Publications*, pages 501–516. Cambridge University Press, 2005.

[27] Jörg Rambau and Victor Reiner. A survey of the higher Stasheff-Tamari orders. In Folkert Müller-Hoissen, Jean Marcel Pallo, and Jim Stasheff, editors, *Associahedra, Tamari Lattices and Related Structures – Tamari Memorial Festschrift*, volume 299 of *Progess in Mathematics*, chapter 18, pages 351–390. Springer, Berlin, Heidelberg, 2012.

[28] Jim Ruppert and Raimund Seidel. On the difficulty of triangulating three-dimensional nonconvex polyhedra. *Discrete & Computational Geometry*, 7:227–253, 1992.

[29] Francisco Santos. A point set whose space of triangulations is disconnected. *Journal of the American Mathematical Society*, 13:611–637, 2000.

[30] Tyler Seacrest and Francis Edward Su. A lower bound technique for triangulations of simplotopes. *SIAM Journal on Discrete Mathematics*, 32(1):1–28, 2018.

[31] Ngoc Mai Tran. Enumerating polytropes. *Journal of Combinatorial Theory, Series A*, 151:1–22, October 2017.

Statements and Declarations

Funding

The author declares that no funds, grants, or other support were received for the preparation of this manuscript.

Competing Interests

The author has no conflicts of interest to declare that are relevant to the content of this article.

Availability of data and materials

The software package `TOPCOM` is available online via the professional web page of the author under `https://www.wm.uni-bayreuth.de/de/team/rambau_joerg/TOPCOM/`. It is further installable as a package in several major Linux distributions.