EPiC Series in Computing
Volume 73, 2020, Pages 465–482
LPAR23. LPAR-23: 23rd International Conference on Logic for Programming, Artificial Intelligence and Reasoning

Finding Periodic Apartments via Boolean Satisfiability and Orderly Generation

Jarkko Savela, Emilia Oikarinen, and Matti Järvisalo

HIIT, Department of Computer Science, University of Helsinki, Finland
{jarkko.savela, emilia.oikarinen, matti.jarvisalo}@helsinki.fi

Abstract

Motivated by Gromov's subgroup conjecture (GSC), a fundamental open conjecture in the area of geometric group theory, we tackle the problem of the existence of particular types of subgroups—arising from so-called periodic apartments—for a specific set of hyperbolic groups with respect to which GSC is currently open. This problem is equivalent to determining whether specific types of graphs with a non-trivial combination of properties exist. The existence of periodic apartments allows for ruling the groups out as some of the remaining potential counterexamples to GSC. Our approach combines both automated reasoning techniques—in particular, Boolean satisfiability (SAT) solving—with problem-specific orderly generation. Compared to earlier attempts to tackle the problem through computational means, our approach scales noticeably better, and allows for both confirming results from a previous computational treatment for smaller parameter values as well as ruling out further groups out as potential counterexamples to GSC.

1 Introduction

Automated reasoning has proven effective as a means of providing further understanding of fundamental open mathematical conjectures. Motivated by recent successes in applying Boolean satisfiability (SAT) based techniques in settling various forms of conjectures [20, 28, 17, 27, 22, 4, 5, 21, 3, 2], in this paper we tackle a fundamental open conjecture in the area of geometric group theory [12, 13] via a combination of SAT solving [1] and orderly generation [30].

In particular, we focus on the Gromov subgroup conjecture (GSC) which states—in group-theoretic terms—that *every one-ended hyperbolic group contains a subgroup isomorphic to the fundamental group of a closed surface of genus at least 2*. Adding to the interest in hyperbolic groups, isomorphism, word and conjugacy problems—while undecidable for groups in general—are decidable for hyperbolic groups [11, 10]. Despite its seemingly involved statement, GSC has received a fair amount of attention in terms of classical mathematical treatment as well as recently from a computational angle.

GSC has been established to hold for various hyperbolic groups [16, 8, 18, 26, 19], and it is even known that a randomly chosen (one-ended hyperbolic) group almost always contains a surface subgroup [9]. A subclass for which GSC remains open are *non-right-angled* hyperbolic groups, which hence may still provide counterexamples disproving GSC. Within this class of

hyperbolic groups, a set of 23 counterexample candidates were recently identified [24]. As shown in [25], the question of whether a particular one of these groups contains a surface subgroup arising from so-called *periodic apartments* is equivalent to determining whether a specific type of graph (with a non-trivial combination of properties) exists. Of interest from the view point of computational mathematics is the fact that ruling each individual group of the 23 out as counterexamples to GSC boils down to determining whether there exist bipartite, 3-regular, connected multigraphs that both decompose exactly into sets of cycles of specific lengths and admit a type of a labeling using a representation of the group in question [25].

In the rest of this paper, we will refrain from requiring in-depth background on geometric group theory from the reader, and will focus on the task of determining whether small graphs with the just-mentioned properties exist. In particular, as first recently considered in [25], this graph search problem admits computational approaches towards ruling out counterexamples candidates to GSC. The size of the graphs of interest is controlled by a parameter called the *genus*, arising from the classification of surfaces in topology. Towards ruling out the 23 groups as potential counterexamples to GSC, a specialized computational approach was developed in [25] for exhaustively searching over all graphs under the smallest genus value $g = 2$. The approach consisted of generating all of the bipartite, 3-regular multigraphs for $g = 2$, with $16(g-1) = 16$ nodes and $24(g-1) = 24$ edges, and performing a specialized depth-first search for each of the graphs. As a results, 5 of the 23 groups of interest were ruled out from the set of remaining counterexample candidates, leaving 18 groups for further inspection. However, the approach was reported not to scale beyond $g = 2$.

In this work, we harness an intricate combination of automated reasoning and orderly generation for pushing towards larger genus values, with the aim of ruling out further groups as potential counterexamples to GSC. It should be noted that the sheer number of graphs to be considered increases drastically as the value of genus increases: while for $g = 2$ there are 773 bipartite, 3-regular, connected multigraphs, already for $g = 3$ the number is $\approx 13 \cdot 10^9$. Furthermore, the number of cycles the graphs need to compose of also increases with g; and an individual graph may allow for decomposing into several different cyclesets, each of which need to be considered. We will show that our approach scales further to $g = 3$ and, with the help of massive parallelization, even to $g = 4$. We study the applicability of SAT solving for both the task of decomposing the graphs of interest into cyclesets as well as the final task of labeling the graphs oriented according to their cyclesets. For scaling to $g = 4$, we apply a combination of orderly generation of the graphs and their cyclesets (obtaining strong symmetry breaking), and employ SAT solving for the final labeling task. As whole, this combination of techniques provides massive scalability improvements over the specialized approach of [25]. In terms of GSC, we are able to rule out further 4 groups from the remaining set of possible counterexamples to GSC. Our results also serve as an independent validation of the results in [25] for $g = 2$.

The rest of this paper is organized as follows. We start by explaining the graph-theoretic characterization in Section 2. We then detail SAT encodings for the subproblems of finding cyclesets and labelings in Sections 3 and 4, respectively. We develop an orderly generation algorithm for pruning the search space of graphs of interest in Section 5. Finally, we provide an overview of the new empirical findings obtained using combinations of the SAT encodings and orderly generation in Section 6.

2 Combinatorial Characterization

With the aforementioned motivations, we focus on the 23 groups identified and studied by Kangaslampi and Vdovina [24]. Recall that the question of whether a particular one of these

T_{15}	T_{16}	T_{17}	T_{18}
(x_1, x_{15}, x_1)	(x_1, x_{15}, x_1)	(x_1, x_{15}, x_1)	(x_1, x_{15}, x_1)
(x_{10}, x_2, x_1)	(x_{10}, x_2, x_1)	(x_{10}, x_2, x_1)	(x_{10}, x_2, x_1)
(x_{11}, x_5, x_2)	(x_{11}, x_5, x_2)	(x_{11}, x_4, x_2)	(x_{11}, x_4, x_2)
(x_{14}, x_4, x_2)	(x_{14}, x_3, x_2)	(x_{14}, x_6, x_2)	(x_{14}, x_3, x_2)
(x_3, x_6, x_3)	(x_8, x_4, x_3)	(x_3, x_{12}, x_3)	(x_9, x_5, x_3)
(x_{15}, x_{12}, x_3)	(x_{14}, x_9, x_3)	(x_8, x_5, x_3)	(x_{13}, x_7, x_3)
(x_7, x_8, x_4)	(x_6, x_6, x_4)	(x_8, x_{13}, x_4)	(x_8, x_6, x_4)
(x_{15}, x_{13}, x_4)	(x_{15}, x_{13}, x_4)	(x_{14}, x_{14}, x_4)	(x_{14}, x_8, x_4)
(x_8, x_7, x_5)	(x_7, x_7, x_5)	(x_9, x_7, x_5)	(x_6, x_{12}, x_5)
(x_{14}, x_9, x_5)	(x_{15}, x_{12}, x_5)	(x_{11}, x_9, x_5)	(x_{15}, x_{13}, x_5)
(x_9, x_{11}, x_6)	(x_{14}, x_{11}, x_6)	(x_7, x_8, x_6)	(x_7, x_9, x_6)
(x_{11}, x_{13}, x_6)	(x_{11}, x_{13}, x_7)	(x_{15}, x_{12}, x_6)	(x_{11}, x_{10}, x_7)
(x_{10}, x_9, x_7)	(x_9, x_{12}, x_8)	(x_{10}, x_{13}, x_7)	(x_{14}, x_{12}, x_8)
(x_{12}, x_{12}, x_8)	(x_{10}, x_9, x_8)	(x_{11}, x_{10}, x_9)	(x_{13}, x_{11}, x_9)
(x_{13}, x_{14}, x_{10})	(x_{13}, x_{12}, x_{10})	(x_{15}, x_{13}, x_{12})	(x_{15}, x_{12}, x_{10})

Table 1: The relations $x_i x_j x_k = 1$ of groups T_{15}, T_{16}, T_{17}, and T_{18}, represented as triplets (x_i, x_j, x_k) where the generators are $x_1, \ldots, x_{15}$.

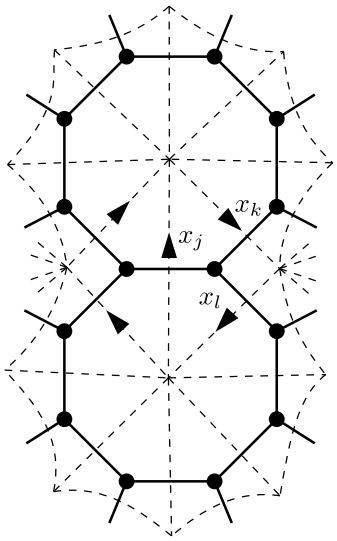


Figure 1: Tesselation with hyperbolic triangles whose angles are $\frac{\pi}{4}$.

groups contains a surface subgroup arising from so-called *periodic apartments* is equivalent to determining whether a specific type of a graph (with a non-trivial combination of properties) exists. More specifically, the existence of a periodic apartment implies the existence of a surface subgroup, which implies that the particular group in question is provably not a counterexample to Gromov's subgroup conjecture.

Towards determining whether periodic apartments exist for one of the 23 groups, we modularize the search for the witnessing graph structures into three consecutive steps: (i) generating connected, bipartite, and 3-regular multigraphs of specific size; (ii) for each of the graphs from (i), determining whether the graph admits a directed decomposition into a set of cycles of length 8, and if it does, enumerating all of such cyclesets; and (iii) for each of the graphs admitting a cycleset decomposition from (ii), checking whether the graph oriented by its cyclesets admits a specific type of a labeling. For (i), one can apply available orderly generation tools directly. For (ii) and (iii), we present SAT encodings using which the enumeration/checking can be delegated to a SAT solver (as described in Sections 3 and 4, respectively). As an alternative to separately computing (i) and (ii), we also present an alternative approach which allows for orderly generation of all graphs admitting a cycleset decomposition and all of the cyclesets (as described in Section 5). In the remainder of this section, we will more precisely define each of these tasks.

We denote the 23 groups from [24] by $T_1, \ldots, T_{23}$. Each of the 23 groups is finitely represented using 15 generators $x_1, \ldots, x_{15}$ and 15 relations of length 3 represented using *triplets* of the form (x_i, x_j, x_k), with the meaning $x_i x_j x_k = 1$. As examples, the relations of groups T_{15}, T_{16}, T_{17} and T_{18} are listed in in Table 1. A complete listing of the representations of each of the 23 groups is provided in [24]. Observe that the relations may contain multiple instances of the same generator; the group T_{15} serves as an example.

Let $g > 1$ be a fixed natural number and T_i one of the 23 groups. Each relation (x_j, x_k, x_l) of T_i determines an oriented triangle whose edges are labeled with x_j, x_k and x_l. A periodic apartment of genus g is then a surface of genus g ("donut with g holes") constructed from these

triangles in a way that the labels and orientations match. The graph we wish to find is the dual graph of this triangulation, i.e., a graph whose nodes represent the triangles with an edge between two nodes if the respective triangles share an edge, see Figure 1.

We next explain briefly how the relevant properties of these graphs arise and define families of (multi)graphs that will be useful to characterize and modularize the problem. We refer the reader to [25] for a complete description. Observe that the graphs must be 3-regular since they represent a triangulation, and bipartite because every other triangle must have "opposite" orientation. Here we consider the triangles to be hyperbolic with all angles $\frac{\pi}{4}$ from which it follows that exactly 8 triangles intersect at every corner. The number of vertices and edges can be deduced to be $16(g-1)$ and $24(g-1)$ using Euler's formula $V - E + F = 2 - 2g$ [25].

Definition 1. *Let $G = (V, E)$ be an undirected graph with the set V of nodes and the multiset E of edges $\{u, v\}$ with $u, v \in V$. Denote by* $\mathsf{edges}(v)$ *the set of edges that are incident to a node $v \in V$. Furthermore, let $g > 1$ be a natural number. Then $G \in \mathsf{base}(g)$ if $|V| = 16(g-1)$, $|E| = 24(g-1)$, and G is connected, bipartite, and 3-regular, i.e., $|\mathsf{edges}(v)| = 3$ for all $v \in V$.*

Hence step (i) reduces to enumerating $\mathsf{base}(g)$ for a given genus g. Observe that, due to 3-regularity and connectedness of graphs in $\mathsf{base}(g)$, two nodes can have at most two edges between them. These pairs of parallel edges are called *double edges*. We assume that the edges have unique names (in addition to their set of nodes), so that they are identifiable.

Next, for step (ii), we consider graphs in $G = (V, E) \in \mathsf{base}(g)$ which admit a *cycleset*. Let $\mathsf{directed}(E) = \{(v_1, v_2), (v_2, v_1) \mid \{v_1, v_2\} \in E\}$ denote the decomposition of E into directed edges. A cycleset for $G = (V, E) \in \mathsf{base}(g)$ is a set of $6(g-1)$ cycles of length 8 in $\mathsf{directed}(E)$ covering the directed decomposition. Formally, an 8-cycle in G is a sequence $(e_0, \ldots, e_7)$, where for each $i \in \{0, \ldots, 7\}$, $e_i \in \mathsf{directed}(E)$ and the end-points of the consecutive edges $e_i = (v', v)$ and $e_{(i+1) \bmod 8} = (v, v'')$ are distinct nodes $v, v', v'' \in V$. Note that each undirected edge is traversed twice (once in each direction) since a cycleset covers the entire directed decomposition.

Definition 2. *Let $G = (V, E)$ be an undirected graph such that $G \in \mathsf{base}(g)$ for some $g > 1$. Then $G \in \mathsf{cycles}(g)$ if G contains a cycleset, i.e., a set of $6(g-1)$ 8-cycles, where each edge in* $\mathsf{directed}(E)$ *is traversed exactly once.*

There may be several cyclesets for $G \in \mathsf{cycles}(g)$; we denote by $\mathsf{cyclesets}(G)$ the set of all cyclesets of G. Hence $\mathsf{cyclesets}(G) = \emptyset$ implies $G \notin \mathsf{cycles}(g)$ and vice versa. Observe that a cycleset covers all the edges in G, and this allows one to uniquely order the incident edges of

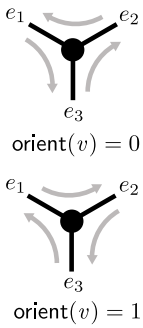


Figure 2: The 2 possible orientations of a node.

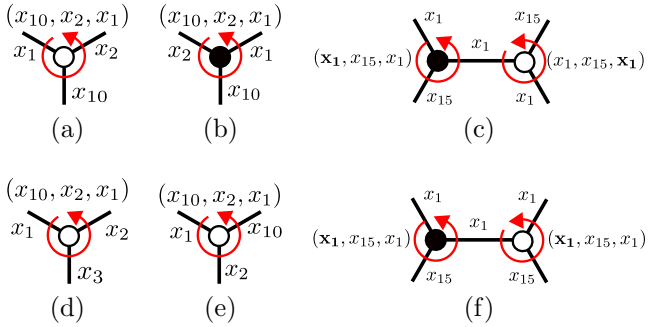


Figure 3: Examples of valid labelings in (a)–(c) and invalid labelings (d)–(f), see Example 2.

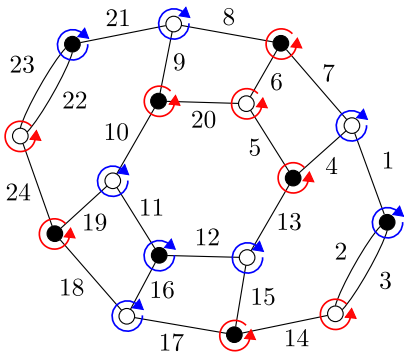


Figure 4: Graph G_7^2

Table 2: A cycleset of graph G_7^2.

1	3	2	1	4	5	6	7
2	14	15	12	16	17	14	3
4	7	8	9	10	11	12	13
5	13	15	17	18	19	10	20
6	20	9	21	22	23	21	8
11	19	24	22	23	24	18	16

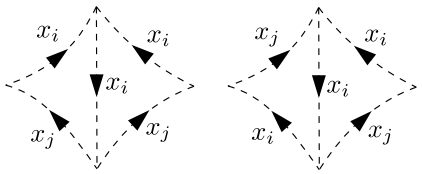


Figure 5: Invalid (left) and valid (right) adjacent triangles.

each node. There are exactly two ways in which the cycles can pass through a node (see Figure 2), and hence each node has an *orientation* determined by the cycles passing through it. The orientation of a node v is represented using an *order function* O_v mapping each $e \in \mathsf{edges}(v)$ to its successor. For instance, in the topmost case in Figure 2 the order function O_v is defined as $O_v(e_1) = e_3$, $O_v(e_2) = e_1$, and $O_v(e_3) = e_2$.

Example 1. *Consider the graph G_7^2 shown in Figure 4. Observe that G_7^2 is a non-simple graph with two double edges. Now, $G_7^2 \in \mathsf{base}(2)$, i.e., G_7^2 is bipartite, connected, 3-regular and has 16 nodes and 24 edges. The nodes of the graph are colored black/white to indicate bipartiteness.*

Furthermore, $G_7^2 \in \mathsf{cycles}(2)$, i.e., $\mathsf{cyclesets}(G_7^2) \neq \emptyset$. One of the cyclesets of G_7^2 is listed in Table 2 (using the names of undirected edges to avoid obscuring the figure too much). Each node is passed through exactly three times by the cycles (in the case of double edges the node is passed twice by the same cycle and once by another). Each undirected edge is traversed twice (once in each direction). The orientations arising from the cycleset in Table 2 are denoted using arrows around the nodes in Figure 4.

We are now ready to define the graphs that represent periodic apartments of the hyperbolic building corresponding to some group T_i. Recall that the nodes of any such graph represent triangles whose sides have labels x_i and edges between nodes the adjacency of the triangles. To represent the action of group T_i on the apartment, we need to define conditions for a *valid labeling* for $G \in \mathsf{cycles}(g)$, which corresponds to a labeling of the sides of the triangles.

Definition 3. *Let $G = (V, E) \in \mathsf{cycles}(g)$ for some $g > 1$. A labeling of G using a group T_i, denoted by (L_v, L_e), consists of two functions:*

- *L_v mapping $v \in V$ to relations (x_i, x_j, x_k) of T_i, and*
- *L_e mapping $e \in E$ to generators x_i of T_i,*

in a way that the label of node v matches the labels of $e \in \mathsf{edges}(v)$, i.e., for all $v \in V$ it holds that if $L_v(v) = (x_i, x_j, x_k)$, then $\{L_e(e) \mid e \in \mathsf{edges}(v)\} = \{x_i, x_j, x_k\}$.

The acceptability of a labeling depends on the orientations of the nodes of G. The orientations are determined by the cyclesets of $G \in \mathsf{cycles}(G)$ together with a chosen 2-coloring (due to

bipartiteness). We assume here a *fixed* 2-coloring of G using colors black and white. Intuitively, the labeling of a *white node* has to *match the orientation* of the node, and the labeling of a *black node* has to *match the inverted orientation* of the node. Furthermore, two adjacent nodes in G cannot be labeled with the same triplet unless (1) the triplet contains two occurrences of the same generator, (2) the connecting edge is labeled with the duplicated generator, and (3) the index of the generator in the triplet is different for both nodes. For a detailed discussion on how exactly these constraints arise, we refer the reader to [25]. For an intuition of the conditions (1)–(3), recall that the triangles tesselating the "surface with g holes" are oriented and have their sides labeled with elements x_i (generators of T_i), and the sides of two adjacent triangles that overlap must have the same label. Additionally, due to the nature of the group action, two triangles of the *same type* (i.e., labelled using the same triplet of T_i) cannot share the same side. This means that two triangles of the same type with three *different* labels x_i, x_j and x_k cannot be adjacent. Two triangles of the same type can, however, be adjacent if their labels are x_i, x_i and x_j, since then they may be attached from different sides which have the same label, see Figure 5.

Definition 4. *Let $G = (V, E) \in \mathsf{cycles}(g)$ for some $g > 1$ and $W \in \mathsf{cyclesets}(G)$. A labeling (L_v, L_e) of G using T_i respects the orientation induced by W if the following conditions hold for each $v \in V$ with $\mathsf{edges}(v) = \{e_1, e_2, e_3\}$ and orientation $O_v(e_1) = e_2$, $O_v(e_2) = e_3$, $O_v(e_3) = e_1$ in W.*

(i) $L_v(v) = (L_e(e_1), L_e(e_2), L_e(e_3))$ *if v is white.*

(ii) $L_v(v) = (L_e(e_3), L_e(e_2), L_e(e_1))$ *if v is black.*

(iii) For each $e = \{v, w\} \in E$, if $L_v(v) = L_v(w)$, then this label (triplet) has two occurrences of the same generator x_i, $L_e(e) = x_i$, and the orientations of v and w are such that e has different position (index) in $L_v(v)$ and $L_v(w)$.

Given $W \in \mathsf{cyclesets}(G)$, a labeling using T_i is *valid* with respect to W if it satisfies conditions (i)–(iii) of Definition 4, and otherwise *invalid*.

Example 2. *Figure 3 illustrates examples of valid and invalid labelings. In (a)–(c), valid labelings in the neighborhood of a white node, a black node, and for two adjacent nodes which are assigned the same triple, respectively, are illustrated. Invalid labels are illustrated in (d)–(f). In (d) the labels of the edges do not match the triple; in (e) the labels match the triplet but in the wrong order; and (f) illustrates how a labeling of two adjacent nodes which are assigned the same triplet may fail. Here the generators x_1 in bold in (c) and (f) denote the elements of the triplets corresponding to the label of the connecting edge.*

Finally, we define the set of graphs that admit a valid labeling.

Definition 5. *Let $G = (V, E)$ be an undirected graph such that $G \in \mathsf{cycles}(g)$ for some $g > 1$, and let T_i be one of the 23 groups constructed in [24]. We say that $G \in \mathsf{labels}(T_i, g)$, if for some set $W \in \mathsf{cyclesets}(G)$ there exists a valid labeling (L_v, L_e) of G using T_i with respect to W.*

Observe that $\mathsf{labels}(T_i, g) \subseteq \mathsf{cycles}(g) \subseteq \mathsf{base}(g)$ for all $g > 1$ and T_i such that $i \in \{1, \ldots, 23\}$. The existence of a graph $G \in \mathsf{labels}(T_i, g)$ is connected to the existence of surface subgroups in T_i as follows.

Theorem 1. *[25] Let T_i be one of the 23 groups constructed in [24] and $g > 1$ a natural number. If $\mathsf{labels}(T_i, g) \neq \emptyset$, then there exists a periodic apartment in the hyperbolic building corresponding to T_i that is invariant under the action of a genus g surface.*

The existence of a periodic apartment in the hyperbolic building implies the existence of a surface subgroup of genus g. Hence, if $\mathsf{labels}(T_i, g) \neq \emptyset$, then Gromov's subgroup conjecture holds for T_i.[1]

Corollary 1. *Let T_i and g be as defined in Theorem 1. If $\mathsf{labels}(T_i, g) \neq \emptyset$, then there exists a subgroup in T_i isomorphic to the fundamental group of a genus g surface.*

3 Enumerating Cyclesets via SAT

We develop a SAT encoding $\phi_{\mathsf{cycles}}(G, g)$ for checking whether a given graph $G = (V, E)$ in $\mathsf{base}(g)$ (for an arbitrary genus $g > 1$) is in $\mathsf{cycles}(g)$. The encoding allows for the enumeration of cyclesets as it captures all $W \in \mathsf{cyclesets}(G)$. Note that for a fixed genus g, all graphs in $\mathsf{base}(g)$ can be generated using off-the-shelf tools, in particular Multigraph [6, 7] implementing an approach to generating isomorph-free lists of multigraphs of specified size and degree sequence.

For the following, let $G = (V, E)$ be an undirected graph in $\mathsf{base}(g)$ for some $g > 1$. Recall that $\mathsf{directed}(E)$ denotes the decomposition of undirected edges of E into directed ones. Finding a cycleset of G boils down to partitioning $\mathsf{directed}(E)$ into $N = 6(g-1)$ mutually disjoint sets of size 8 while ensuring that each partition forms a cycle (recall Definition 2).

We use the following variables in $\phi_{\mathsf{cycles}}(G, g)$: $\mathsf{orient}(v)$ indicates which one of the two possible orientations node v is assigned (recall Figure 2), and $\mathsf{inCycle}(e, p)$ and $\mathsf{index}(e, i)$, respectively, indicate into which cycle p and index i in that cycle, respectively, the directed edge $e \in \mathsf{directed}(E)$ is assigned. The SAT encoding $\phi_{\mathsf{cycles}}(G, g)$ is detailed in Figure 6. The encoding is composed of two parts: one ensuring unique assignment of edges into cycles, and the other checking that the orientations of nodes are compatible with the partitioning of $\mathsf{directed}(E)$. The constraints (1)–(4) ensure that $\mathsf{directed}(E)$ is partitioned into N subsets of size 8. The cardinality constraints (1) and (2) encode that each edge is assigned to exactly one cycle and one index, respectively.[2] The constraint (3) ensures that each partition is non-empty, whereas the constraint (4) blocks two distinct edges $e_1, e_2 \in \mathsf{directed}(E)$ from being assigned to the same partition at the same index. Note that our encoding works with both simple and non-simple graphs as long as edges between the same nodes are given distinct names.

The remaining constraints (5)–(8) encode that the partitions of $\mathsf{directed}(E)$ are cycles, i.e., an edge $e = (x, y)$ at index i in partition p implies the edge at index $i + 1 \mod 8$ starts at y. Here we need to take into account that there are two possible orientations for a node. Let us consider the edges incident to a node v, i.e., $\mathsf{edges}(v) = \{e_1, e_2, e_3\}$. We denote the corresponding directed edges by e_i^{out} and e_i^{in} where e_i^{out} refers to the directed edge going outwards from v and vice versa for e_i^{in}. Furthermore, let $O_v(e_1) = e_2$, $O_v(e_2) = e_3$, and $O_v(e_3) = e_1$ correspond to orientation 1, and $O_v(e_1) = e_3$, $O_v(e_2) = e_1$, and $O_v(e_3) = e_2$ correspond to orientation 0 (recall Figure 2). We then denote by $\mathsf{pairs}_1(v)$ the successor pairs of the directed edges corresponding to orientation 1, i.e., $\mathsf{pairs}_1(v) = \{(e_1^{\text{in}}, e_2^{\text{out}}), (e_2^{\text{in}}, e_3^{\text{out}}), (e_3^{\text{in}}, e_1^{\text{out}})\}$ and by $\mathsf{pairs}_0(v)$ the successor pairs corresponding to orientation 0, i.e., $\mathsf{pairs}_0(v) = \{(e_1^{\text{in}}, e_3^{\text{out}}), (e_3^{\text{in}}, e_2^{\text{out}}), (e_2^{\text{in}}, e_1^{\text{out}})\}$. The constraint (5) then encodes, for each $v \in V$, that each pair of directed edges appearing in $\mathsf{pairs}_1(v)$ is assigned to the same partition if v has orientation 1; the constraint (6) encodes the same for $\mathsf{pairs}_0(v)$ and orientation 0. The constraint (7) ensures that for each $v \in V$ the edges in the pairs $\mathsf{pairs}_1(v)$ have subsequent indices $(i, i + 1 \mod 8)$ if v has orientation 1, while the constraint (8) encodes the same for $\mathsf{pairs}_0(v)$ and orientation 0.

[1] It is not known if the existence of a surface subgroup of genus g implies the existence of a periodic apartment.

[2] We use the standard pairwise SAT encoding for the exactly-one constraints, which is suitable here as the number of variables in the constraints is small.

$$\text{for each } e \in \mathsf{directed}(E): \sum_{p \in \{0,\ldots,N-1\}} \mathsf{inCycle}(e,p) = 1 \tag{1}$$

$$\sum_{i \in \{0,\ldots,7\}} \mathsf{index}(e,i) = 1 \tag{2}$$

$$\text{for each } p \in \{0,\ldots,N-1\}: \bigvee_{e \in \mathsf{directed}(E)} \mathsf{inCycle}(e,p) \tag{3}$$

$$\text{for each } p \in \{0,\ldots,N-1\}, i \in \{0,\ldots,7\}, e_1, e_2 \in \mathsf{directed}(E) \text{ such that } e_1 \neq e_2: \\ \neg\mathsf{inCycle}(e_1,p) \vee \neg\mathsf{index}(e_1,i) \vee \neg\mathsf{inCycle}(e_2,p) \vee \neg\mathsf{index}(e_2,i) \tag{4}$$

$$\text{for each } v \in V, p \in \{0,\ldots,N-1\}, (e,e') \in \mathsf{pairs}_1(v): \\ \neg\mathsf{orient}(v) \rightarrow (\mathsf{inCycle}(e,p) \leftrightarrow \mathsf{inCycle}(e',p)) \tag{5}$$

$$\text{for each } v \in V, p \in \{0,\ldots,N-1\}, (e,e') \in \mathsf{pairs}_0(v): \\ \mathsf{orient}(v) \rightarrow (\mathsf{inCycle}(e,p) \leftrightarrow \mathsf{inCycle}(e',p)) \tag{6}$$

$$\text{for each } v \in V, i \in \{0,\ldots,7\}, i' = (i+1) \bmod 8, (e,e') \in \mathsf{pairs}_1(v): \\ \neg\mathsf{orient}(v) \rightarrow (\mathsf{index}(e,i) \leftrightarrow \mathsf{index}(e',i')) \tag{7}$$

$$\text{for each } v \in V, i \in \{0,\ldots,7\}, i' = (i+1) \bmod 8, (e,e') \in \mathsf{pairs}_0(v): \\ \mathsf{orient}(v) \rightarrow (\mathsf{index}(e,i) \leftrightarrow \mathsf{index}(e',i')) \tag{8}$$

$$\text{for each } p \in \{0,\ldots,N-1\}, e, e' \in \mathsf{directed}(E) \text{ such that } e' < e: \\ \mathsf{inCycle}(e,p) \wedge \mathsf{index}(e,0) \rightarrow \neg\mathsf{inCycle}(e',p) \tag{9}$$

$$\text{for each } p \in \{1,\ldots,N-1\}, e, e' \in \mathsf{directed}(E) \text{ such that } e' < e: \\ \mathsf{inCycle}(e',p) \wedge \mathsf{index}(e',0) \rightarrow \bigwedge_{p'<p} \neg(\mathsf{inCycle}(e,p') \wedge \mathsf{index}(e,0)) \tag{10}$$

Figure 6: SAT encoding of cyclesets.

We additionally break the symmetries underlying this cycleset representation via enfocing ordering constraints to rule out all but one assignment corresponding to each distinct cycleset. Specifically, under a fixed linear ordering of $\mathsf{directed}(E)$, constraints (9) enforce that each cycle $p \in \{0,\ldots,N-1\}$ must have its least edge at index 0 (breaking rotation symmetry in each cycle), and constraints (10) enforce that the cycles $p \in \{0,\ldots,N-1\}$ are in increasing order according to their edges at index 0.

All in all, the encoding provided in Figure 6 captures $\mathsf{cyclesets}(G)$ for any given G, and in particular, any satisfying assignment of $\phi_{\mathsf{cycles}}(G,g)$ can be projected into $W \in \mathsf{cyclesets}(G)$.

Proposition 1 (Correctness of the cycleset encoding.)**.** *Let $g > 1$ be a natural number and $G \in \mathsf{base}(g)$. There exists a bijective mapping between the satisfying assignments of $\phi_{\mathsf{cycles}}(G,g)$ and $W \in \mathsf{cyclesets}(G)$.*

Hence, if $\phi_{\mathsf{cycles}}(G,g)$ is unsatisfiable, then $G \notin \mathsf{cycles}(g)$, and if $\phi_{\mathsf{cycles}}(G,g)$ is satisfiable, then $G \in \mathsf{cycles}(g)$ and the satisfying assignments, when projected, yield exactly $\mathsf{cyclesets}(G)$.

4 Labeling Oriented Graphs via SAT

We next consider the problem of determining whether a given graph $G = (V,E) \in \mathsf{cycles}(g)$ can be labeled using some $T_i \in \{T_1,\ldots,T_{23}\}$, i.e., whether $G \in \mathsf{labels}(T_i,g)$. To solve this problem,

$$\text{for each } e \in E : \sum_{l \in L} \mathsf{edgeLabel}(e, l) = 1 \tag{11}$$

$$\text{for each } v \in V : \sum_{t \in T} \mathsf{nodeLabel}(v, t) = 1 \tag{12}$$

$$\sum_{o \in \{0,1,2\}} \mathsf{offset}(v, o) = 1 \tag{13}$$

$$\text{for each } v \in V,\ o \in \{0,1,2\},\ t \in T : \mathsf{nodeLabel}(v, t) \wedge \mathsf{offset}(v, o) \rightarrow \bigwedge_{i \in \{0,1,2\}} \mathsf{edgeLabel}(e_i, l_i) \tag{14}$$

$$\text{where } e_i = O'(v)[i],\ l_i = t[(i + o) \mod 3]$$

$$\text{for each } \{v_1, v_2\} \in E,\ t = (l_1, l_2, l_3) \in T \text{ such that } l_1 \neq l_2 \neq l_3 \neq l_1 : \neg\mathsf{nodeLabel}(v_1, t) \vee \neg\mathsf{nodeLabel}(v_2, t) \tag{15}$$

$$\text{for each } e = \{v_1, v_2\} \in E,\ t = (l, l, l') \in T \text{ such that } l \neq l' : \mathsf{nodeLabel}(v_1, t) \wedge \mathsf{nodeLabel}(v_2, t) \rightarrow \mathsf{edgeLabel}(e, l) \tag{16}$$

$$\text{for each } e = \{v_1, v_2\} \in E, (o_1, o_2) \in \mathsf{bad\_offsets}_W^G(e), t = (l, l, l') \in T \text{ such that } l \neq l', \mathsf{nodeLabel}(v_1, t) \wedge \mathsf{nodeLabel}(v_2, t) \rightarrow \neg\mathsf{offset}(v_1, o_1) \vee \neg\mathsf{offset}(v_2, o_2) \tag{17}$$

Figure 7: SAT encoding of labeling.

we formulate a SAT encoding which can be used to decide for any suited graph G, group T_i and cycleset $W \in \mathsf{cyclesets}(G)$ whether there exists a valid labeling of G using T_i with respect to W. We refer to the encoding as $\phi_{\mathsf{labels}}(G, T_i, W)$. If there exists $W \in \mathsf{cyclesets}(G)$ such that $\phi_{\mathsf{labels}}(G, T_i, W)$ is satisfiable, then $G \in \mathsf{labels}(T_i, g)$, whereas if $\phi_{\mathsf{labels}}(G, T_i, W)$ is unsatisfiable for all $W \in \mathsf{cyclesets}(G)$, then $G \notin \mathsf{labels}(T_i, g)$.

Assume that $G = (V, E) \in \mathsf{cycles}(g)$ for fixed $g > 1$. Recall that $\mathsf{cyclesets}(G)$ can be obtained, e.g., via the cycleset encoding presented in the previous section, or alternatively through orderly generation, as outlined later in Section 5. Let $W \in \mathsf{cyclesets}(G)$ be one of the cyclesets. We denote the generators and relations of a fixed group T_i by L and T, respectively. Let (V_B, V_W) be a 2-coloring of G, i.e., $V_B \cup V_W = V$ and $V_B \cap V_W = \emptyset$ such that there is no edge in E consisting of nodes only in V_B or only in V_W. We use here a compact triplet representation for the order functions arising from W. Let $O \colon V \to E^3$ denote the orientations of nodes as triplets such that $O(v) = (e_1, e_2, e_3)$, if $O_v(e_1) = e_2$, $O_v(e_2) = e_3$, and $O_v(e_3) = e_1$. To make the encoding more uniform we define the adjusted orientations $O' \colon V \to E^3$ by flipping the orientations of black nodes, i.e., $O'(v) = O(v)$ for all $v \in V_W$ and $O'(v) = (O(v)[2], O(v)[1], O(v)[0])$ for all $v \in V_B$. In particular, using the adjusted orientations, items (i) and (ii) in Definition 4 coincide.

From the adjusted orientation $O'(v) = (e_1, e_2, e_3)$ representing a cyclic ordering of $\mathsf{edges}(v)$, we derive an ordering of the labels of the edges as (l_1, l_2, l_3), where $l_i \in L$ is the label of e_i for $i \in \{1, 2, 3\}$. The label of node v, on the other hand, is also a triplet $t = (x, y, z) \in T$ of elements in L. Since also the triplets (y, z, x) and (z, x, y) represent the same cyclic ordering of the labels, we use a Boolean variable $\mathsf{offset}(v, o)$, where $o \in \{0, 1, 2\}$, to indicate which representative of triplet t the node v takes. The other variables used in the encoding are $\mathsf{edgeLabel}(e, l)$ and $\mathsf{nodeLabel}(v, t)$ indicating that edge $e \in E$ has label (generator) $l \in L$ and node $v \in V$ has label (triplet) $t \in T$, respectively.

The SAT encoding $\phi_{\mathsf{labels}}(G, T_i, W)$ is detailed in Figure 7. The constraints (11)–(13) ensure

that each edge $e \in E$ has a unique label (generator) and that each node $v \in V$ a unique triplet (relation) and an offset. The constraints (14) ensure that the chosen representative of triplet t of node v matches the triplet (l_1, l_2, l_3) of the labels of $\mathsf{edges}(v)$. Constraints (14) together with cardinality constraints in (11)–(13) rule out cases (d) and (e) in Figure 3. Constraints (15)–(17) together with the cardinality constraints (11)–(13) rule out cases where adjacent nodes are assigned triples inconsistently, such as case (f) in Figure 3. Intuitively, constraint (15) blocks any pair $\{v_1, v_2\}$ of adjacent nodes from taking the same triplet t with three distinct labels. Constraint (16) ensures that any edge $e = \{v_1, v_2\}$ such that the nodes v_1 and v_2 take the same triplet $t = (l, l, l')$ takes the duplicated label l.

As stated in Definition 4, the position of the label of an edge between two nodes must be different in the triplets of the nodes, see cases (c) and (f) in Figure 2. The offsets of the nodes along with the chosen triplet determine the positions of the label. We thus define the set of illegal pairs of offsets for $e \in E$ as

$$\mathsf{bad\_offsets}_W^G(e) = \{(o_1, o_2) \mid e = \{v_1, v_2\}, o_1, o_2 \in \{0, 1, 2\}, i_1 + o_1 = i_2 + o_2 \mod 3 \\ \text{where } O'(v_1)[i_1] = e \text{ and } O'(v_2)[i_2] = e\},$$

and rule them out via constraint (17). Here i_1 and i_2 are the indices of the connecting edge e in the orientations of v_1 and v_2, respectively. Further, the sums $i_1 + o_1$ and $i_2 + o_2$ (mod 3) represent the indices of the label of e in the triplets of v_1 and v_2, respectively. The set $\mathsf{bad\_offsets}_G(e)$ thus contains the pairs of offsets that would give the label of $e = \{v_1, v_2\}$ the same index in both v_1 and v_2.

Proposition 2 (Correctness of the labeling encoding)**.** *Let $G \in \mathsf{cycles}(g)$ for $g > 1$, and T_i be one of the 23 groups constructed in [24] . It holds that $G \in \mathsf{labels}(T_i, g)$ iff there exists $W \in \mathsf{cyclesets}(G)$ such that $\phi_{\mathsf{labels}}(G, T_i, W)$ is satisfiable.*

A valid labeling for G with respect to W using T_i can be directly extracted from a satisfying assignment to $\phi_{\mathsf{labels}}(G, T_i, W)$. This allows for constructing a periodic apartment in the hyperbolic building corresponding to T_i that is invariant under the action of a genus g surface.

5 Orderly Generation of Graphs having Cyclesets

In this section we develop an algorithm for the exhaustive generation of graphs in $\mathsf{cycles}(g)$ and $\mathsf{cyclesets}(G)$ for each $G \in \mathsf{cycles}(g)$. We achieve this by generating *configurations* which can be mapped to pairs (G, W) where $G \in \mathsf{cycles}(g)$ and $W \in \mathsf{cyclesets}(G)$. Our approach builds on Read's *orderly generation* method [30] which we adapt to our problem setting and into which we integrate optimizations specific to our configurations for empirical efficiency. In orderly generation the explicit removal of duplicates in the isomorph-free collection is avoided by generating configurations in a specific *order* and outputting only canonical configurations lexicographically greater than the previous ones.

To implement an orderly generation algorithm we need a formal definition of a *configuration* and *linear ordering* of configurations, a notion of *canonicity*, as well as a depth parameter and a method of augmentation. The idea of the construction is to take $N = 6(g-1)$ directed cycles of length 8 and glue the directed edges together (into undirected edges) while preserving connectedness and bipartiteness of the induced graph, in the end resulting in a 3-regular graph. We number the cycles with elements $c \in C = \{0, 1 \dots N-1\}$, and for each cycle, its directed edges $i \in I = \{0, \dots, 7\}$. Hence each directed edge is identified by a pair $(c, i) \in C \times I$. We set the source node of each directed edge at even (odd) index black (white).

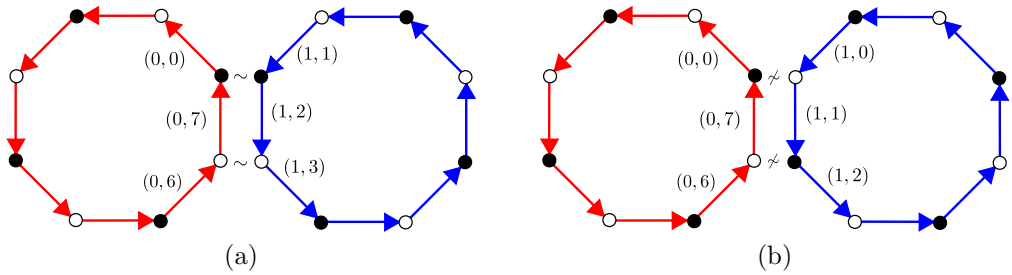


Figure 8: (a) Glueing edges $(0, 7)$ and $(1, 2)$ is allowed since one index is odd while the other is even. (b) Glueing $(0, 7)$ to $(1, 1)$ is not allowed since both indices are odd.

Formally a configuration $\mathcal{X}$ is a list of ordered pairs $x_i = \langle e_1, e_2 \rangle$ of edge identifiers $e_1 = (c_1, i_1)$ and $e_2 = (c_2, i_2)$, representing the directed edges that have been glued together to form undirected edges. We take the depth parameter to be the length of a configuration. There are $8N$ distinct edge identifiers since there are N 8-cycles, and therefore the maximal length of a configuration is $4N$. Let $\mathcal{C}(N)$ be the set of configurations of length $4N$ built from N 8-cycles.

We define an ordering of $\mathcal{C}(N)$ by lifting the natural lexicographic ordering of edge identifiers $e = (c, i)$ to lists of pairs of edge identifiers, i.e., configurations. Thus $e_1 = (c_1, i_1) \leq e_2 = (c_2, i_2)$ iff $c_1 < c_2$ or $c_1 = c_2$ and $i_1 < i_2$. Now $x = \langle e_1, e_2 \rangle \leq x' = \langle e'_1, e'_2 \rangle$ iff $e_1 < e'_1$ or $e_1 = e'_1$ and $e_2 < e'_2$. The ordering of configurations takes into account their varying length, i.e., $\mathcal{X} = (x_0, \ldots, x_l) \leq \mathcal{Y} = (y_0, \ldots, y_k)$ iff (i) $\mathcal{X} = \mathcal{Y}$, (ii) $l < k$, or (iii) $l = k$ and $\exists i$ such that $x_i < y_i$ and $x_j = y_j$ for all $j < i$.

Observe that configurations $\mathcal{X}, \mathcal{Y} \in \mathcal{C}(N)$ may be syntactically different representations of the same graph-cycleset pair. Specifically, the names of cycles and their edges bear no relevance to the induced graph-cycleset pair. We may thus permute the cycle names arbitrarily and the indices of edges in steps of two (due to bipartiteness). Stated group-theoretically, the symmetry group of $\mathcal{C}(N)$ is the direct product of the symmetric group on N elements S_N (corresponding to permuting the names of cycles) and the N-wise product of the cyclic group of 4 elements (corresponding to permuting the indices). Thereby $\mathcal{X}$ and $\mathcal{Y}$ are equivalent iff there exists a permutation $\pi \in S_N \times C_4^N$ such that $\mathcal{X} = \pi(\mathcal{Y})$. We denote by $[\mathcal{X}]$ the equivalence class of $\mathcal{X} \in \mathcal{C}(N)$, i.e., $[\mathcal{X}] = \{\mathcal{Y} \in \mathcal{C}(N) \mid \mathcal{Y}$ and $\mathcal{X}$ are equivalent$\}$. The canonical representative of $[\mathcal{X}]$ is then the least $\mathcal{Y} \in [\mathcal{X}]$.

Our orderly generation algorithm is outlined as Algorithm 1. The recursive algorithm starts with an initial configuration $\mathcal{X}$ of length l and incrementally constructs a list of canonical configurations of maximal length $(4N)$ having $\mathcal{X}$ as their prefix. Hence, starting from the empty configuration, we obtain the list of canonical configurations built from N 8-cycles. The main parts of the algorithm are the *augmenting* and *canonicity checking* steps. Augmenting adds a new pair $\langle e_1, e_2 \rangle$ to the end of configuration $\mathcal{X}$ thereby increasing its length by 1. Canonicity checking consists of determining whether a given configuration is the canonical representative of its equivalence class, and is performed greedily after each augmentation step since this efficiently prunes the search space. It can be shown that our algorithm satisfies the following conditions, which are a stronger variant of Read's necessary and sufficient conditions [30] for the correctness of orderly generation.

(i) Each canonical configuration of length $l + 1$ can be produced from exactly one canonical configuration of length l.

(ii) If $\mathcal{X}$ and $\mathcal{Y}$ are configurations of length l and $\mathcal{X} < \mathcal{Y}$, the configurations produced by augmenting $\mathcal{X}$ must precede the ones produced by augmenting $\mathcal{Y}$.

Algorithm 1: Orderly generation algorithm $\mathsf{orderly}(\mathcal{X}, l, N)$

Input: configuration $\mathcal{X}$, level l, number of 8-cycles N
Output: the list of canonical representatives of $\mathcal{C}(N)$ having $\mathcal{X}$ as their prefix
$\mathcal{G} \leftarrow ()$ `/* empty list */`
if $l = 4N$ **then**
  if $\mathsf{canonical}(\mathcal{X})$ **then** $\mathcal{G} \leftarrow \mathsf{append}(\mathcal{G}, \mathcal{X})$
else
  $A \leftarrow \mathsf{augmentations}(\mathcal{X})$
  while $A \neq ()$ **do**
    $a \leftarrow \mathsf{pop}(A)$ `/* returns first element while removing from` A `*/`
    $\mathcal{X}' \leftarrow \mathsf{append}(\mathcal{X}, a)$
    if $\mathsf{canonical}(\mathcal{X}')$ **then** $\mathcal{G} \leftarrow \mathsf{append}(\mathcal{G}, \mathsf{orderly}(\mathcal{X}', l+1, N))$
return $\mathcal{G}$

(iii) The augmenting operation produces the new configurations in order.

Augmenting Let $\mathcal{X} = (x_0, \dots x_{l-1})$ be a configuration of length l. The augmenting subroutine produces a list of pairs $\langle e_1, e_2 \rangle$ used in the main algorithm to extend $\mathcal{X}$, denoted by $\mathsf{augmentations}(\mathcal{X})$. To ensure correctness the list $\mathsf{augmentations}(\mathcal{X})$ should contain every element $p = \langle e_1, e_2 \rangle$ for which $\mathsf{append}(\mathcal{X}, p)$ is canonical. Note that if $\mathsf{augmentations}(\mathcal{X})$ contains elements yielding non-canonical configurations, correctness is still maintained, but empirical performance may degrade. Also note that any canonical configuration not having $\mathcal{X}$ as a prefix will be produced by augmenting some other suitable configuration $\mathcal{X}'$.

We denote the free edges of $\mathcal{X}$ by $E^{\mathsf{free}}_{\mathcal{X}} = (C \times I) \setminus \mathsf{edges}(\mathcal{X})$, where $\mathsf{edges}(\mathcal{X})$ consists of all the edge identifiers appearing in the ordered pairs $x_i \in \mathcal{X}$. The list $\mathsf{augmentations}(\mathcal{X})$ thus consists of pairs $\langle e_1, e_2 \rangle$ where $e_1, e_2 \in E^{\mathsf{free}}_{\mathcal{X}}$. For each pair, we set e_1 to the least element in $E^{\mathsf{free}}_{\mathcal{X}}$, denoted by e_{min}. For e_2 we consider a subset E' of the remaining elements $E^{\mathsf{free}}_{\mathcal{X}} \setminus \{e_{\mathsf{min}}\}$, i.e., $\mathsf{augmentations}(\mathcal{X}) = (\langle e_1, e_2 \rangle \mid e_1 = e_{\mathsf{min}}, e_2 \in E')$ with the pairs listed in order.

For $E' \subseteq E^{\mathsf{free}}_{\mathcal{X}} \setminus \{e_{\mathsf{min}}\}$ we observe the following. Denote $e_1 = (c_1, i_1)$, $e_2 = (c_2, i_2)$, and let $c_{\max}$ be the largest cycle number appearing in $\mathcal{X}$. To ensure bipartiteness we include in E' only edges e_2 such that i_2 has different parity to i_1, see Figure 8 for examples. If $\mathsf{edges}(\mathcal{X}) = \{0, \dots, c_{\max}\} \times I$, then all the edges currently in $\mathcal{X}$ have already been paired. In this case augmenting with a new pair would yield a configuration with a disconnected underlying graph, and hence if $\mathsf{edges}(\mathcal{X}) = \{0, \dots, c_{\max}\} \times I$, we set $E' = \emptyset$. Finally, E' is reduced by observing that we need to ensure that any partial configuration can be augmented to a full configuration so that the underlying graph is 3-regular. Two types of nodes can prevent this: (i) nodes with degree > 3 or (ii) nodes with degree 1 or 2 such that their degree cannot be increased by augmenting. Hence we exclude from E' all edges that would result in such nodes in the underlying graph, guaranteeing the 3-regularity of the induced graph in a configuration of maximal length $4N$. Observe that while these considerations are enough to guarantee that only augmentations yielding connected, bipartite, eventually 3-regular underlying graphs are produced, E' can be reduced even further by excluding pairs that would necessarily yield a non-canonical configuration (details omitted due to space constraints).

Canonization While the question of whether graph isomorphism is polynomial-time decidable is open in general, checking isomorphism of degree-bounded graphs is polynomial-time computable [29]. In our case, the underlying graphs are 3-regular, and hence it is not surprising that configurations can be canonized in polynomial-time.

Algorithm 2: Canonicity checking algorithm $\mathsf{canonical}(\mathcal{X})$

Input: configuration $\mathcal{X}$
Output: Boolean indicating whether $\mathcal{X}$ is canonical
$c_{\mathsf{max}} \leftarrow$ largest cycle number in $\mathcal{X}$
for $c \in \{0, \ldots, c_{\mathsf{max}}\}$ **do**
  for $a \in \{0, 2, 4, 6\}$ **do**
    $\pi \leftarrow \{\langle (c, i), (0, i + a \bmod 8)\rangle \mid i \in \{0, \ldots, 7\}\}$
    $\pi \leftarrow \mathsf{extend}(\pi, \mathcal{X})$ `/* extend π to a full permutation */`
    $\mathcal{Y} \leftarrow \pi(\mathcal{X})$ `/* compute Y */`
    if $\mathcal{Y} < \mathcal{X}$ **then return** *false*
return *true* `/* No equivalent configuration smaller than X was found */`

We outline our canonicity checking procedure as Algorithm 2. Given a configuration $\mathcal{X} = (x_0, \ldots x_{q-1})$ of length q, the algorithm iterates through $\mathcal{X}$ while constructing the permutation minimizing the configuration. If this permutation produces a configuration smaller than $\mathcal{X}$, we conclude that $\mathcal{X}$ is not canonical. The algorithm goes through permutations mapping each cycle to 0 with different offsets. Once it is decided which cycle maps to 0 and with which offset, there is only one way to extend the permutation in a way that minimizes $\mathcal{Y} = \pi(\mathcal{X})$. Hence it suffices to iterate through every value for c and a instead of trying every element of $S_N \times C_4^N$. Recall that any configuration $\mathcal{X}$ produced by the augmentation in our algorithm is connected.

The extending of a partial permutation π in $\mathsf{extend}(\pi, \mathcal{X})$ works as follows. Assume that permutation π maps cycle c to 0 with offset a. The pairs of $\mathcal{X}$ containing edges from cycle c will become the prefix of the permuted configuration $\mathcal{Y} = \pi(\mathcal{X})$ since 0 is the least cycle. Let $\mathsf{cindex}(\pi, \mathcal{X}) = \{c\}$ and $\mathsf{cindex}(\pi, \mathcal{Y}) = \{0\}$ be the sets of cycle indices currently mapped by π in $\mathcal{X}$ and $\mathcal{Y}$, respectively. Since the configuration $\mathcal{X}$ is connected, there is at least one pair of edges $\langle (c_1, i_1), (c_2, i_2)\rangle$ in $\mathcal{X}$ such that (i) $c_1 \in \mathsf{cindex}(\pi, \mathcal{X})$ and $c_2 \notin \mathsf{cindex}(\pi, \mathcal{X})$ or (ii) $c_1 \notin \mathsf{cindex}(\pi, \mathcal{X})$ and $c_2 \in \mathsf{cindex}(\pi, \mathcal{X})$. Out of these pairs we choose the one whose permuted element is the smallest. Let us assume that this pair is $\langle (c_1, i_1), (c_2, i_2)\rangle$ and $c_1 \in \mathsf{cindex}(\pi, \mathcal{X})$ (the other case where $c_2 \in \mathsf{cindex}(\pi, \mathcal{X})$ can be treated in a similar fashion). We extend π to map c_2 to c', where $c' = \max(\mathsf{cindex}(\pi, \mathcal{Y})) + 1$, in order to produce the smallest possible $\mathcal{Y}$. The corresponding offset is chosen in a way that makes the index i' in $\pi(c_2, i_2) = (c', i')$ as small as possible. After updating $\mathsf{cindex}(\pi, \mathcal{X}) = \{c_2\} \cup \mathsf{cindex}(\pi, \mathcal{X})$ and $\mathsf{cindex}(\pi, \mathcal{Y}) = \{c'\} \cup \mathsf{cindex}(\pi, \mathcal{Y})$, the permutation π can be extended in this way as long as $\mathcal{X}$ contains pairs in which one of the cycle identifiers has already been permuted and the other has not. If at any point $\mathcal{X}$ contains only pairs where both cycle indices are either permuted or not permuted, and $|\mathsf{cindex}(\pi, \mathcal{Y})| \neq c_{\mathsf{max}} + 1$, where c_{max} is the largest cycle number appearing in $\mathcal{X}$, then none of the cycles in the range of π are attached to cycles outside its range. This would mean that the graph corresponding to the configuration consists of at least two disjoint components, which contradicts the fact the augmentation algorithm we use only produces connected configurations.

6 Experiments

We report on results obtained by employing different combinations of orderly generation (recall Section 5) and the SAT encodings for enumerating cycleset decompositions and labeling graph-cycleset pairs (recall Sections 3 and 4, respectively). In particular, we confirm the results earlier reported in [25] for genus $g = 2$ using two semi-independent ways. Furthermore, we

Table 3: Statistics for different steps of the approaches.

Approach	Genus	Orientable graphs	Labelable graphs	Pairs	Hits
G+SAT^2	2	12	4	274	152
OG+labelSAT	2	12	4	84	15
OG+labelSAT	3	1399	26	5 872	67
OG+labelSAT	4	–	127	6 125 906	491

exhaustively treat the cases of $g = 3$ and $g = 4$, altogether ruling out 4 further groups out of the 23 T_i's. We also report on the runtime distribution of employing the MiniSAT solver [14] through the PySAT interface [23] for finding cyclesets and labelings. All experiments were run on computing nodes with Xeon E5-2680 v4 2.4-GHz processors and 256-GB RAM under CentOS 7. Our implementation, empirical data and witness graphs found are available via https://bitbucket.org/coreo-group/periodic-apartments/.

In the following, we will refer by G+SAT^2 to the approach consisting of (i) generating all connected, bipartite, and 3-regular graphs with $16(g-1)$ nodes and $24(g-1)$ edges (for a given genus g) using the off-the-shelf Multigraph tool; (ii) using the SAT encoding of Section 3 to enumerate the cyclesets of the graphs in (i); and (iii) using the SAT encoding of Section 4 to determine the existence of a labeling for the graph-cycleset pairs from (ii). In contrast, we will refer by OG+labelSAT to the approach consisting of (i) generating the graph-cycleset pairs directly with the orderly approach of Section 5 followed by (ii) SAT checking the existence of a labeling for each pair.

Confirmation of Earlier Results for Genus 2 Kangaslampi and Vdovina exhaustively treated the genus–2 case [25]. Their approach consisted of (i) generating all connected, bipartite, 3-regular graphs with 16 nodes and 24 edges while treating simple and non-simple graphs separately; (ii) for each of these 773 graphs a depth-first search for determining the sets of 6 8-cycles, and (iii) specialized depth-first search over each of the graph-cycleset pairs to determine if the pair admits a labeling. As reported in [25], this approach does not scale beyond $g = 2$. Our approach differs in terms of generating directly a more strict set of graphs and in employing a SAT solver for checking the existence of labelings.

We exhaustively treated the case of genus $g = 2$ using both G+SAT^2 and OG+labelSAT, and checked that the results obtained with both methods agree with those reported in [25], i.e., we found that exactly the same five groups (T_1, T_2, T_7, T_9, and T_{18}) are ruled out at $g = 2$. Table 3 gives more detailed statistics on the different steps on the approaches. The numbers of graphs for which cyclesets exists (column **Orientable graphs**) and which admit a valid labeling (**Labelable graphs**) are the same for G+SAT^2 and OG+labelSAT (as should be). The total number of graph-cycleset pairs and the number of ones admitting a labeling are listed in columns **Pairs** and **Hits**, respectively. The lower numbers in these columns for OG+labelSAT compared to G+SAT^2 witness the stronger symmetry-breaking in OG+labelSAT resulting in a noticeably smaller average number of cyclesets per graph. The SAT-based labeling phase of both approaches was quite efficient for genus 2, with cumulative runtimes of 420 seconds for G+SAT^2 and 143 for OG+labelSAT. The SAT-based cycleset generation phase of G+SAT^2 over the 773 graphs from phase (i) took a total of 348 seconds, while the orderly generation phase of OG+labelSAT took 3 seconds. This suggests that OG+labelSAT scales better of the two to large genera. We will report on new results for genera 3 and 4 focusing on OG+labelSAT.

New Results beyond Genus 2 Kangaslampi and Vdovina were unable to scale their approach beyond genus 2. In contrast, our OG+labelSAT approach, using trivial parallelization, allowed for an efficient exhaustive analysis of genera 3 and 4. As a result, we are able to rule out four

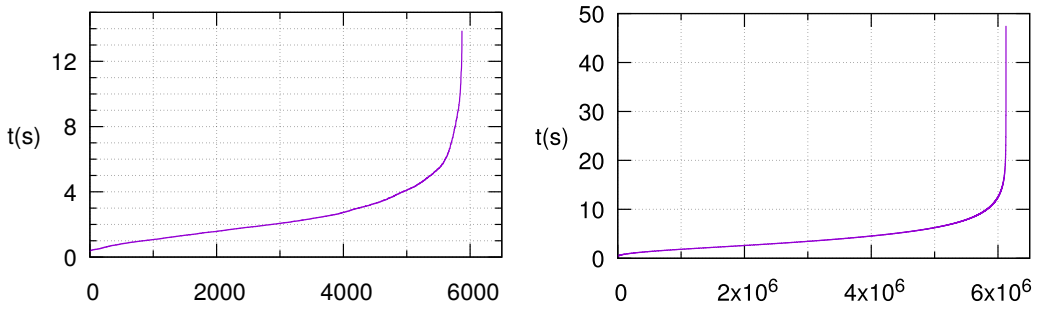


Figure 9: Runtime distribution of the labeling phase of OG+labelSAT for genus 3 (left) and genus 4 (right). The total runtime of 23 sat calls (one for each T_i) per graph-cycleset pair is plotted.

more groups (summarized in Table 4). Concrete witnesses (labeled oriented graphs) for these four groups T_6, T_{13}, T_{15}, and T_{16} are provided in Figure 10.

We are able to rule out three more groups at genus 3, and a further group at genus 4. While Multigraph (phase (i) of G+SAT^2) would generate 13 703 003 409 graphs at genus 3, orderly generation at genus 3 resulted in 5872 graph-cycleset pairs, out of which 81 admit a labeling; see Table 3. For genus 4 we generated 6 125 906 graph-cycleset pairs, out of which 491 admit a labeling. At genus 3, orderly generation took approximately 14 hours and the labeling phase less than 8 hours. Orderly generation of genus 4 configurations, on the other hand, took 883 500 hours, and checking them for labeling took 7 241 hours. Figure 9 shows the total runtime of the labeling phase per graph-cycleset pair over the 23 T_i's for genus 3 (left) and genus 4 (right). Each individual SAT call took less than 2 seconds at genus 3 and 12 seconds at genus 4.

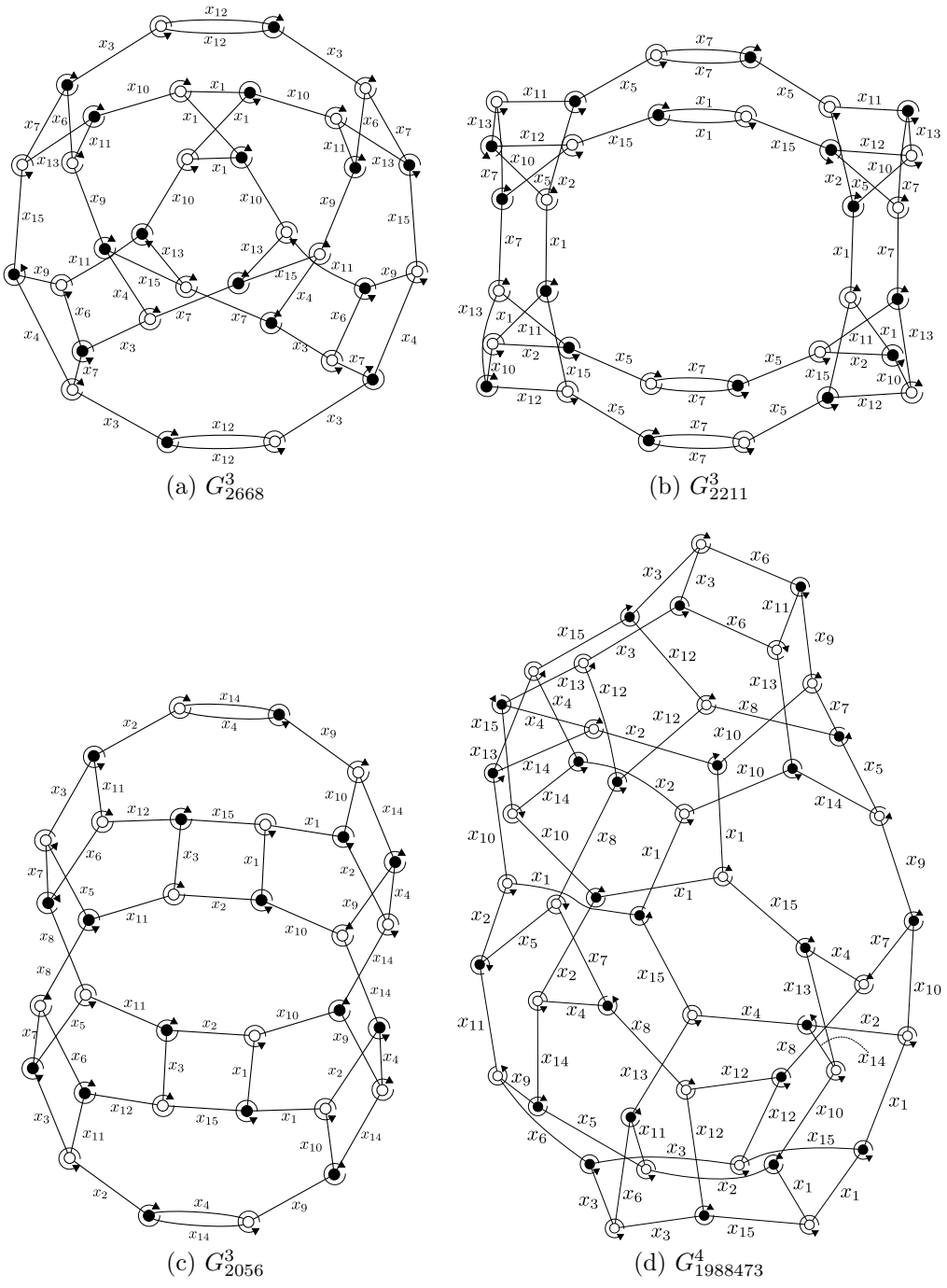
7 Conclusions

We presented a computational study of the applicability of combinations of SAT solving and orderly generation to a problem arising from geometric group theory, dealing in particular with determining whether one of 23 specific groups earlier put forth by Kangaslampi and Vdovina [24] would serve as a counterexample to the famous subgroup conjecture of Gromov. While earlier computational treatment of this problem setting was restricted to genus 2 [25], we showed that a combination of SAT solving and orderly generation allows for scaling to the significantly larger search spaces induced by genera 3 and 4. As a result, we both provided an independent confirmation of the earlier results for genus 2 [25], and ruled out four more groups out of the 23 as counterexamples to Gromov's subgroup conjecture by exhaustively treating genus 3 and 4.

Acknowledgments This work was financially supported by Academy of Finland (grants 276412, 312662, and 322869). Computational resources were provided by Finnish Grid and Cloud Infrastructure [15]. The authors thank Riikka Kangaslampi for discussions and Markus Meringer and Gunnar Brinkmann for correspondence on orderly generation and Multigraph.

Table 4: Groups ruled out at genus g with those not ruled out at smaller value of g in bold.

Approach	**Genus 2**	**Genus 3**	**Genus 4**
G+SAT^2	T_1, T_2, T_7, T_9, T_{18}		
OG+labelSAT	T_1, T_2, T_7, T_9, T_{18}	T_1, T_2, $\boldsymbol{T_6}$, T_7, T_9, $\boldsymbol{T_{13}}$, $\boldsymbol{T_{16}}$, T_{18}	T_1, T_2, T_7, T_9, $\boldsymbol{T_{15}}$, T_{18}



(a) G^3_{2668} (b) G^3_{2211}

(c) G^3_{2056} (d) $G^4_{1988473}$

Figure 10: A graph labeled using (a) T_6, (b) T_{16}, (c) T_{13}, and (d) T_{15}. The arrows around the nodes of the graphs denote the orientations arising from the cycleset allowing the graph to be labeled. Bipartiteness is indicated using colors black and white, and the x_i's denote the labels of edges. The triplet of each node can be deduced from the labels of its incident edges.

References

[1] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2009.

[2] Joshua Brakensiek, Marijn Heule, and John Mackey. The resolution of Keller's conjecture. *CoRR*, abs/1910.03740, 2019. http://arxiv.org/abs/1910.03740.

[3] Florian Brandl, Felix Brandt, Manuel Eberl, and Christian Geist. Proving the incompatibility of efficiency and strategyproofness via SMT solving. *Journal of the ACM*, 65(2):6:1–6:28, 2018.

[4] Felix Brandt and Christian Geist. Finding strategyproof social choice functions via SAT solving. *Journal of Artificial Intelligence Research*, 55:565–602, 2016.

[5] Felix Brandt, Christian Geist, and Dominik Peters. Optimal bounds for the no-show paradox via SAT solving. *Mathematical Social Sciences*, 90:18–27, 2017.

[6] Gunnar Brinkmann. Minibaum webpage. http://caagt.ugent.be/minibaum/. Accessed: 2020-01-24.

[7] Gunnar Brinkmann. Fast generation of cubic graphs. *Journal of Graph Theory*, 23(2):139–149, 1996.

[8] Danny Calegari. Surface subgroups from homology. *Geometry & Topology*, 12(4):1995–2007, 2008.

[9] Danny Calegari and Alden Walker. Random groups contain surface subgroups. *Journal of the American Mathematical Society*, 28(2):383–419, 2015.

[10] François Dahmani and Vincent Guirardel. Foliations for solving equations in groups: free, virtually free, and hyperbolic groups. *Journal of Topology*, 3(2):343–404, 2010.

[11] François Dahmani and Vincent Guirardel. The isomorphism problem for all hyperbolic groups. *Geometric and Functional Analysis*, 21(2):223–300, 2011.

[12] Pierre de La Harpe. *Topics in geometric group theory*. Chicago Lectures in Mathematics. University of Chicago Press, 2000.

[13] Cornelia Druţu and Michael Kapovich. *Geometric group theory*, volume 63 of *Colloquium Publications*. American Mathematical Society, 2018.

[14] Niklas Eén and Niklas Sörensson. An extensible SAT-solver. In Enrico Giunchiglia and Armando Tacchella, editors, *Theory and Applications of Satisfiability Testing, 6th International Conference, SAT 2003. Santa Margherita Ligure, Italy, May 5-8, 2003 Selected Revised Papers*, volume 2919 of *Lecture Notes in Computer Science*, pages 502–518. Springer, 2003.

[15] Finnish Grid and Cloud Infrastructure, 2019. urn:nbn:fi:research-infras-2016072533.

[16] David Futer and Anne Thomas. Surface quotients of hyperbolic buildings. *International Mathematics Research Notices*, 2012(2):437–477, 2011.

[17] Christian Geist and Ulrich Endriss. Automated search for impossibility theorems in social choice theory: Ranking sets of objects. *Journal of Artificial Intelligence Research*, 40:143–174, 2011.

[18] Cameron Gordon, Darren Long, and Alan Reid. Surface subgroups of coxeter and artin groups. *Journal of Pure and Applied Algebra*, 189(1):135 – 148, 2004.

[19] Cameron Gordon and Henry Wilton. On surface subgroups of doubles of free groups. *Journal of the London Mathematical Society*, 82(1):17–31, 2010.

[20] Paul R. Herwig, Marijn J.H. Heule, Martijn van Lambalgen, and Hans van Maaren. A new method to construct lower bounds for Van der Waerden numbers. *The Electronical Journal of Combinatorics*, 14(1), 2007.

[21] Marijn Heule. Schur number five. In Sheila McIlraith and Kilian Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 6598–6606. AAAI Press, 2018.

[22] Marijn Heule, Oliver Kullmann, and Victor Marek. Solving and verifying the boolean pythagorean

triples problem via cube-and-conquer. In Nadia Creignou and Daniel Le Berre, editors, *Theory and Applications of Satisfiability Testing - SAT 2016 - 19th International Conference, Bordeaux, France, July 5-8, 2016, Proceedings*, volume 9710 of *Lecture Notes in Computer Science*, pages 228–245. Springer, 2016.

[23] Alexey Ignatiev, António Morgado, and João Marques-Silva. Pysat: A python toolkit for prototyping with SAT oracles. In Olaf Beyersdorff and Christoph M. Wintersteiger, editors, *Theory and Applications of Satisfiability Testing - SAT 2018 - 21st International Conference, SAT 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 9-12, 2018, Proceedings*, volume 10929 of *Lecture Notes in Computer Science*, pages 428–437. Springer, 2018.

[24] Riikka Kangaslampi and Alina Vdovina. Cocompact actions on hyperbolic buildings. *International Journal of Algebra and Computation*, 20(4):591–603, 2010.

[25] Riikka Kangaslampi and Alina Vdovina. Hyperbolic triangular buildings without periodic planes of genus 2. *Experimental Mathematics*, 26(1):54–61, 2017.

[26] Sang-hyun Kim and Sang-il Oum. Hyperbolic surface subgroups of one-ended doubles of free groups. *Journal of Topology*, 7(4):927–947, 2014.

[27] Boris Konev and Alexei Lisitsa. Computer-aided proof of erdős discrepancy properties. *Artificial Intelligence*, 224:103–118, 2015.

[28] Oliver Kullmann. Green-tao numbers and SAT. In Ofer Strichman and Stefan Szeider, editors, *Theory and Applications of Satisfiability Testing - SAT 2010, 13th International Conference, SAT 2010, Edinburgh, UK, July 11-14, 2010. Proceedings*, volume 6175 of *Lecture Notes in Computer Science*, pages 352–362. Springer, 2010.

[29] Eugene Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. *Journal of computer and system sciences*, 25(1):42–65, 1982.

[30] Ronald Read. Every one a winner or how to avoid isomorphism search when cataloguing combinatorial configurations. *Annals of Discrete Mathematics*, 2:107–120, 1978.