Isomorphism Problem for S_d-Graphs

Deniz Ağaoğlu
Masaryk University, Brno, Czech Republic
agaoglu@mail.muni.cz

Petr Hliněný
Masaryk University, Brno, Czech Republic
hlineny@fi.muni.cz

Abstract

An H-graph is the intersection graph of connected subgraphs of a suitable subdivision of a fixed graph H, introduced by Biró, Hujter and Tuza (1992). We focus on S_d-graphs as a special case generalizing interval graphs. A graph G is an S_d-graph iff it is the intersection graph of connected subgraphs of a subdivision of a star S_d with d rays.

We give an FPT algorithm to solve the isomorphism problem for S_d-graphs with the parameter d. This solves an open problem of Chaplick, Töpfer, Voborník and Zeman (2016). In the course of our proof, we also show that the isomorphism problem of S_d-graphs is computationally at least as hard as the isomorphism problem of posets of bounded width.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

Keywords and phrases intersection graph, isomorphism testing, interval graph, H-graph

Digital Object Identifier 10.4230/LIPIcs.MFCS.2020.4

Related Version `https://arxiv.org/abs/1907.01495`

Funding Supported by the Czech Science Foundation, project no. 20-04567S.

Acknowledgements We would like to thank to Pascal Schweitzer for pointing us to the paper [13], and to Onur Çağırıcı for comments on this manuscript.

1 Introduction

A *graph* is a pair $G = (V, E)$ where $V = V(G)$ is the *finite* vertex set and $E = E(G)$ is the edge set. A *subdivision* of an edge $\{u, v\}$ of a graph G is the operation of replacing $\{u, v\}$ with a new vertex x and two new edges $\{u, x\}$ and $\{x, v\}$. Two graphs G_1 and G_2 are *isomorphic* ($G_1 \simeq G_2$) if there exists a bijection f from $V(G_1)$ to $V(G_2)$ such that $\{u, v\} \in E(G_1)$ if and only if $\{f(u), f(v)\} \in E(G_2)$ for all $\{u, v\} \subseteq V(G_1)$, and such f is called an *isomorphism*.

The *graph isomorphism problem* is a well-known problem in computer science which asks to determine whether or not the given two graphs are isomorphic. The complexity status of the graph isomorphism problem is still unknown, despite intense research culminating recently in Babai [3]. On the other hand, the isomorphism problem has been shown to be solvable in polynomial and even in linear time for many particular graph classes such as trees, planar and interval graphs [1, 17, 5].

The latter example (interval graphs) is an instance of a wider concept of intersection graphs which we briefly introduce now. The *intersection graph* for a family of sets is an undirected graph where each set is associated with a vertex of the graph and each pair of vertices are joined by an edge if and only if the corresponding sets have a non-empty intersection.

A graph G is an *interval graph* if the vertex set of G can be mapped into some set of intervals on the real line such that two vertices of G are adjacent if and only if the corresponding intervals intersect. Equivalently, G is an interval graph if and only if G is the intersection graph of subpaths of some suitable path. The isomorphism problem for interval graphs can be solved in linear time [5].

It is well-known that every interval graph is *chordal*, that is, it has no chordless cycle of length more than three. A graph G is chordal if and only if G is the intersection graph of subtrees of some suitable tree [15]. Unlike for interval graphs, the isomorphism problem is *GI-complete* for chordal graphs [20], which means that deciding whether two chordal graphs are isomorphic is polynomial-time equivalent to the graph isomorphism problem in general.

Our motivation is to explore this complexity jump of the isomorphism problem from interval graphs to chordal graphs. For instance, a strict subclass of chordal graphs is formed by the *split graphs*, whose vertex set can be partitioned into a clique and an independent set. Each split graph is the intersection graph of substars of a suitable star S_k (i.e., $K_{1,k}$ or the star with k rays). The isomorphism problem for split graphs is also *GI-complete* [9]. For a simple proof, see Proposition 2.2.

To obtain a finer resolution of graph classes in our study, we introduce so-called H-graphs [4]. For a fixed graph H, an *H-graph* is the intersection graph of connected subgraphs of a suitable subdivision of the graph H. Note that degree-2 vertices in H do not matter. Many intersection graph classes can be viewed as a case of H-graphs. For instance, among the mentioned classes; interval graphs are precisely K_2-graphs, chordal graphs are the union of T-graphs where T ranges over all trees, and split graphs are contained in the union of S_k-graphs where k ranges over all positive integers. Recently, various optimization problems such as maximum clique and minimum dominating set on H-graphs (for particular graphs H) have been shown to be solvable in polynomial and **FPT**-time [7, 8, 12].

As the previous example of isomorphism of split graphs shows, if we want to obtain tractable isomorphism cases among H-graphs, we must consider fixed H (unless, of course, isomorphism were polynomial in general). Hence, in the realm of parameterized complexity [10], we will consider the graph H as the parameter of the isomorphism problem of H-graphs. In particular, we will deal with the case of $H = S_d$ (the star of d rays) where d is a fixed parameter, that is with the *S_d-graph isomorphism problem*, which was stated as an open problem by [7].

FPT algorithms with structural parameters are not so common for the isomorphism problem; this is probably due to the fact that the parameterization restricts the structure of the input graph G_1 and separately that of G_2, but it is not at all clear how the restrictions on G_1 and on G_2 can be "meaningfully combined" in order to check for their isomorphism. Notable examples of nontriviality of **FPT**-time algorithms for graph isomorphism are the following algorithms for graphs of bounded tree-depth [6] and tree-width [16].

Our results

We prove that the S_d-graph isomorphism problem can be solved in **FPT**-time wrt. d, that is, in time $f(d) \cdot |V(G_1)|^{\mathcal{O}(1)}$ for some computable function f. In the course of proving the main result we first give a much simpler combinatorial **FPT**-time algorithm which assumes S_d-graphs of bounded maximum clique size (Theorem 3.1). Without bounding clique size, we prove in Section 4 (Theorem 4.1) that the S_d-graph isomorphism problem includes testing of isomorphism of posets of width d, for which there is *no published combinatorial algorithm* and which can be solved using the classical group-based approach pioneered in Babai [2].

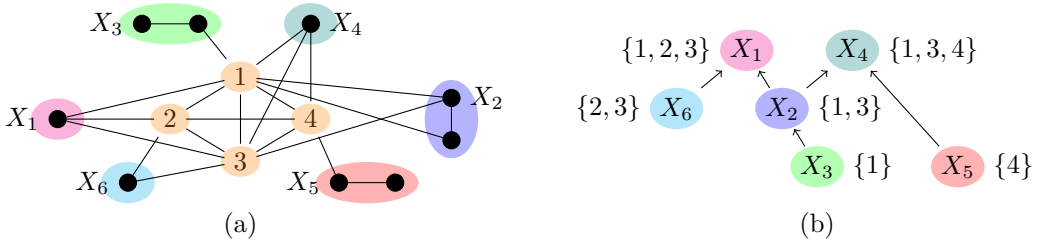

Figure 1 (a) An S_3-graph G with its maximal clique $C = \{1, 2, 3, 4\}$ in the center and the connected components of $G - C$. (b) The partially ordered set P on the components.

To obtain the main result about S_d-graph isomorphism (Theorem 5.9) in Section 5, we combine the case of posets of bounded width with a specific adaptation of the general group-computing approach by Furst, Hopcroft and Luks [13]. Our algorithms do not need S_d-graph representations to be given on the input. The related natural question of an existence of purely combinatorial **FPT**-time algorithms for S_d-graph isomorphism and for isomorphism of posets of bounded width remains open.

We refer to the subsequent sections and the full paper `arXiv:1907.01495v2` for missing terminology and further details, including proofs of (*)-marked statements.

2 Posets, S_d-Graphs and their Recognition

Throughout the paper we assume readers familiar with the basic terminology of *posets* (partially ordered sets), see e.g. [11] and the full paper.

As defined above, an S_d-graph G is the intersection graph of connected subgraphs of a suitable subdivision S' of the star S_d. Note that every ray of S' actually defines an interval subgraph of G. Let C denote the clique of G formed by those representing subgraphs of S' which contain the central vertex of S', and call C the *central clique* of the representation. Clearly, we may assume that C is a maximal clique in G (or we simply adjust the intersection representation). In Figure 1 (a), we see an example of an S_d-graph G (actually, $d = 3$) with one of its maximal cliques C placed in the center and colored orange, and the connected components $X_1, \ldots, X_6$ of $G - C$ colored differently.

When dealing with S_d-graphs, we define the following poset P on the connected components of $G - C$. Let $N_C(X_i)$ denote the set of neighbors of the component X_i in the maximal clique C and we refer to $N_C(X_i)$ as the *attachment* of X_i. Since $X_i \cup C$ induces an interval subgraph of G, the neighborhoods of the vertices of X_i in C form a chain by inclusion. Then, the upper attachment of X_i, denoted by ${N_C}^U(X_i)$, is the maximum neighborhood in C among the vertices of X_i, and $N_C(X_i) = {N_C}^U(X_i)$. Analogously, the lower attachment of X_i, denoted by ${N_C}^L(X_i)$, is the minimum neighborhood in C among the vertices X_i. The minimal edge-cutset between C and X_i is called *attachment edges*, and the component X_i together with its attachment edges is called a *bridge* X_i *of* C.

After determining the attachments of all connected components of $G - C$, the poset P is constructed as follows. A pair (X_i, X_j) of components of $G - C$ is comparable in P, denoted by $X_i \preceq_P X_j$, if and only if ${N_C}^U(X_i) \subseteq {N_C}^L(X_j)$ holds. Otherwise, they are incomparable. When different connected components of $G - C$ have the same attachment, precisely ${N_C}^U(X_i) = {N_C}^L(X_i) = {N_C}^U(X_j) = {N_C}^L(X_j)$, we will always implicitly treat their union as one component / bridge of C in order to maintain antisymmetry of $\preceq_P$.

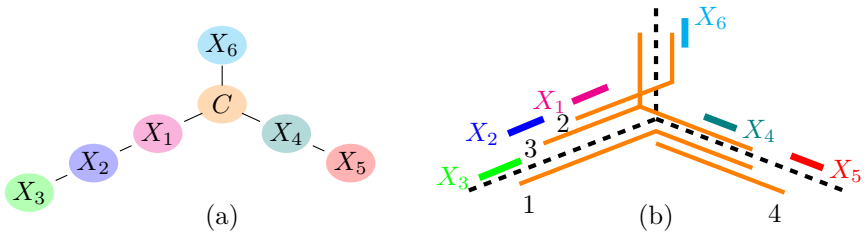

Figure 2 (a) The connected components of $G - C$ are placed on the $d = 3$ rays according to a chain cover of P. (b) The corresponding S_d-representation.

Observe that $X_i \preceq_P X_j$ if and only if there is an interval representation of the subgraph of G induced by $C \cup X_j \cup X_i$ in which C is "to the left" of X_j and X_i is "to the right" of X_j [7].

In Figure 1 (b), we see the poset P on the connected components of $G-C$ from Figure 1 (a). The three tuples of components (X_1, X_2, X_3), (X_4, X_5) and (X_6) form a chain cover of P. In Figure 2 (a), the components are placed on the rays of a subdivision of S_3 according to this chain cover, and C is placed in the center, and in Figure 2 (b), the corresponding S_d-representation of G is given.

Let $G[W]$ denote the induced subgraph of a graph G on a subset $W \subseteq V(G)$. The following characterization of S_d-graphs will be crucial for our algorithm:

▶ **Proposition 2.1** ([7, Lemma 5]). *A graph G is an S_d-graph if and only if there exists a maximal clique C in G such that; (i) for each connected component X_i of $G - C$, the induced subgraph $G[C \cup X_i]$ is an interval graph with C being the leftmost, and (ii) the poset P on the connected components of $G - C$ (P defined above) has a chain cover of size at most d.*

It easily follows that, for each chain $(X_1, \ldots, X_k)$ of a chain cover in P, the subgraph $G[C \cup X_1 \cup \ldots \cup X_k]$ has an interval representation with C being the leftmost clique. Recall; we can solve isomorphism of interval graphs in linear time [5]. Yet, these together do *not mean* that we could simply solve S_d-graph isomorphism by matching the central cliques and pairwise comparing the rays as interval graphs. Since the depth of P is not bounded, there can be exponentially many chain covers, leading to distinct representations which cannot be recognized as isomorphic when compared ray by ray. For example, the S_3-graph in Figure 1 has several different chain covers of size 3, e.g., also (X_1, X_6), (X_4, X_2, X_3) and (X_5).

The full extent of difficulty of dealing with non-unique placement of components on the rays of an S_d-representation can also be illustrated with the following:

▶ **Proposition 2.2** (alternative to [20]). *The isomorphism problem of S_d-graphs with d on the input is GI-complete.*

Proof. Let G_1 and G_2 be arbitrary graphs of the same size. We construct G'_i, $i = 1, 2$, as follows: subdivide every edge with a new vertex, and then make a clique on the original vertex set $V(G_i)$. Then, $G_1 \simeq G_2$ if and only if $G'_1 \simeq G'_2$. Since each G'_i is an S_d-graph for $d = |E(G_i)|$, with the central clique on $V(G_i)$, solving their isomorphism would solve also the isomorphism of given G_1 and G_2. ◀

3 Isomorphism of S_d-Graphs of Bounded Clique Size

To give a more accessible introduction to our approach, we first focus on an easier case of S_d-graphs of clique size at most p. In fact, this subcase does not require us to bound d, and so we will use p as the only parameter here.

Assume that we are given two isomorphic S_d-graphs G and H, and f is an isomorphism between G and H. Let C be a maximal clique of G, and let $D = f(C)$ be the corresponding maximal clique of H. Then each connected component X_i of $G-C$ is mapped to corresponding $Y_i = f(X_i)$ of $H - D$. Moreover, if C happens to be the central clique of some S_d-representation of G, then the subgraph induced by $C \cup X_i$ is an interval graph and we may use the interval graph isomorphism algorithm [5] to compare between $G[C \cup X_i]$ and $H[D \cup Y_i]$. Finally, even if we do not know the right central clique C and corresponding D in advance, we may use the fact that S_d graphs are chordal. Any chordal graph with n vertices has at most n maximal cliques which can be listed in linear time [19]. So we may simply try possible pairs C, D and not depend on particular S_d-representations of G and H.

The previous simple observations suggest the following procedure:

1. Select a suitable maximal clique C of G (cf. Proposition 2.1 and the next step) and loop through all maximal cliques D of H.
2. Find the connected components $X_1, X_2, \ldots, X_k$ of $G - C$, and the connected components $Y_1, Y_2, \ldots, Y_l$ of $H - D$. Check that all $G[C \cup X_i]$ and $H[D \cup Y_j]$ are interval graphs.
3. If $|C| = |D| = q$ and $k = l$, then consider an arbitrary bijective labeling $C \to \{1, 2, \ldots, q\}$. Loop through all possible bijective labelings $D \to \{1, 2, \ldots, q\}$.
4. Pairwise, for $1 \leq i, j \leq k$, compare the induced subgraphs $G[C \cup X_i]$ and $H[D \cup Y_j]$ using the interval graph isomorphism algorithm [5] and respecting the labels on C and D. If the comparison admits overall a perfect matching of isomorphic pairs, i.e., of pairs $\{(i, m_i) : i = 1, \ldots, k\} \subseteq \{1, \ldots, k\}^2$ such that $G[C \cup X_i] \simeq H[D \cup Y_{m_i}]$, then return that G and H are isomorphic.

If the procedure does not return that G and H are isomorphic for any C, D and their labeling, then output that G and H are non-isomorphic.

Note that efficiency of this procedure strongly depends on the bound $q \leq p$ of maximum clique size of G and H. In the general case, we could be processing up to $n!$ distinct permutations of the vertices of D which would not give a polynomial time algorithm.

We hence easily conclude (with details in the full paper):

▶ **Theorem 3.1.** *(*) Isomorphism of S_d-graphs with n vertices where the maximal clique sizes are bounded by a fixed parameter p can be solved in $\mathcal{O}(p! \cdot pn^3)$ time, which belongs to* ***FPT*** *with respect to p (and regardless of d).*

4 From S_d-graphs to Posets of Width d

To extend the previous isomorphism algorithm to S_d-graphs without bounding their clique size, we need to consider isomorphism of posets of bounded width. Recall that a poset is of *width* d if its elements can be covered by d chains (and not by less than d chains).

To justify this shift (and to show that it is unlikely a simple combinatorial algorithm could exist for S_d-graph isomorphism), we give a reduction that S_d-graph isomorphism solves the isomorphism problem for posets of width d.

We are given a poset P of width d with n elements, and we form an S_d-graph G from P (in polynomial time) as follows:

1. Model P by set inclusion between the sets $M_1, \ldots, M_n$, where each M_i consists of all comparable elements with i from the lower levels and itself. Formally, $M_i = \{j \in P : j \preceq_P i\}$ for all $i \in P$.
2. Take the union $M = \bigcup_{i \in P} M_i$ and form the central clique C of size $|M| + 2$ by adding $|M|$ vertices corresponding to the elements of M, two dummy vertices, and all edges on C.

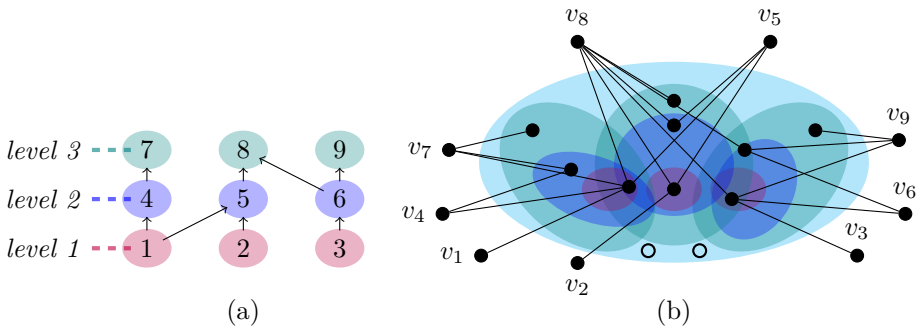

Figure 3 (a) An example poset P (its Hasse diagram), and its levels. (b) The corresponding S_d-graph G where the cyan set corresponds to the central maximal clique.

3. For each M_i, add a new vertex v_i to G whose attachment end vertices are exactly the subset of vertices in C corresponding to M_i.

Note that two dummy vertices are added to C to ensure that C is the unique maximum-size clique of G. See an illustration in Figure 3.

Since the poset P is of width d, its elements can be partitioned into d chains. We can thus distribute the corresponding vertices of $G - C$ to the d rays of S_d, and this straightforwardly results in an S_d-representation of G. Since C is unique as the maximum clique in any such G, we easily get that for two posets P_1 and P_2 of width d, the constructed graphs G_1 and G_2 are isomorphic if and only if P_1 is isomorphic to P_2.

This reduction can be carried out in polynomial time, and hence:

▶ **Theorem 4.1.** *(*) The isomorphism problem of posets of width d reduces in polynomial time to the isomorphism problem of S_d-graphs. This holds even if the poset elements are colored and the colors must be preserved by the isomorphism.*

5 Isomorphism of S_d-graphs in General

Here, we focus on S_d-graphs without bounding their clique size, and give an **FPT**-time algorithm solving their isomorphism problem with respect to d. Since, with d on the input this problem is GI-complete (Proposition 2.2), we need the parameterization by d.

In contrast to Section 3, the maximum clique size in the compared S_d-graphs can grow up to n, resulting in up to $n!$ different labelings to be considered on the central maximal cliques if we use the simple approach. Therefore, while we can efficiently compare the bridges of the central cliques to isomorphism (as interval subgraphs), trying all possible labelings of the central cliques becomes inefficient. Instead, we will encode isomorphism types of the bridges as colors and consider the isomorphism problem for the underlying colored posets of bounded width (recall Proposition 2.1). This will be combined with additional checks which ensure that a bijection between the central cliques, compatible with all isomorphisms between the corresponding pairs of bridges of the central cliques, would exist.

We need to introduce the following algebraic view of the isomorphism problem. Consider two structures $\mathcal{A}$ and $\mathcal{B}$ (graphs, posets). Construct their disjoint union $\mathcal{A} \uplus \mathcal{B}$ and compute its *automorphism group*, i.e., the group of all isomorphisms of $\mathcal{A} \uplus \mathcal{B}$ onto itself. Then $\mathcal{A}$ is isomorphic to $\mathcal{B}$, if and only if the automorphism group of $\mathcal{A} \uplus \mathcal{B}$ contains a permutation exchanging the ground sets of $\mathcal{A}$ and $\mathcal{B}$. In fact, assuming that the structures $\mathcal{A}$ and $\mathcal{B}$ are "connected", it is enough to look for a permutation mapping some point of $\mathcal{A}$ to some of $\mathcal{B}$, and only among generators of the automorphism group. See, e.g., [13] which we will use here.

We start with a high-level overview of our approach. Recall that the elements of any poset R can be partitioned into *levels* $L_i \subseteq R$ where $i \geq 1$; L_1 is formed by the minimal elements of R, and L_{i+1} is inductively formed by the minimal elements of $R \setminus (L_1 \cup \ldots \cup L_i)$.

▶ **Procedure 5.1.** *(*)* Consider two connected n-vertex S_d-graphs G and H with (chosen) central maximal cliques $C \subseteq G$ and $D \subseteq H$, $|C| = |D|$, and assume the connected components $\mathcal{X} = \{X_1, \ldots, X_k\}$ of $G - C$ and $\mathcal{Y} = \{Y_1, \ldots, Y_k\}$ of $H - D$. Let $K = G \uplus H$ be the disjoint union of our graphs. For $Z \in \mathcal{X} \cup \mathcal{Y}$, denote shortly by $K(Z)$: the graph $G[C \cup Z]$ if $Z \in \mathcal{X}$, or the graph $H[D \cup Z]$ if $Z \in \mathcal{Y}$ (so $K(Z)$ is always an induced subgraph of K).

We will test the isomorphism between G and H with respect to C and D as follows:

1. Let P and Q be the posets constructed on $\mathcal{X}$ and $\mathcal{Y}$, respectively, as in Section 2. Make the disjoint union $R = P \uplus Q$, and compute the levels of R. For each pair of components $Z, Z' \in \mathcal{X} \cup \mathcal{Y}$ on the same level of R, compare the interval subgraphs $K(Z)$ and $K(Z')$ to isomorphism and color Z and Z' by the computed isomorphism type. That is, Z and Z' receive the same color if and only if $K(Z) \simeq K(Z')$.
2. Compute the automorphism group Γ of R, respecting the colors of elements from the previous step. We use Theorem 5.2 and Corollary 5.3 here.
3. Compute the subgroup Γ' of Γ, consisting of those automorphisms ϱ of R for which there exists a permutation f_ϱ of the set $C \cup D$ such that the following holds; for every component $Z \in \mathcal{X} \cup \mathcal{Y}$, there is an isomorphism from $K(Z)$ to $K(\varrho(Z))$ whose restriction to the intersection with $C \cup D$ equals the respective restriction of f_ϱ. (In other words, ϱ is such that there exists an automorphism of the graph $K = G \uplus H$ which stabilizes the union of central cliques $C \cup D$ and maps the components from $\mathcal{X} \cup \mathcal{Y}$ according to ϱ.) We use Lemma 5.4, and Theorem 5.6 with Corollary 5.8 here.
4. If P and Q are swapped in some automorphism from Γ', then return that G and H are *isomorphic*.

If the above procedure does not say that G and H are isomorphic for any pair of maximal cliques $C \subseteq G$ and $D \subseteq H$, we return that G and H are *non-isomorphic*.

For a quick reference, we will say that an automorphism $\varrho \in \Gamma$ satisfying the condition in step 3 of this procedure is *consistent on the central cliques* $C \cup D$.

The above high-level approach is not much different from the simple one used in Section 3; however, one difference is crucial. While in the simple approach we explicitly processed all bijections between the central cliques C and D, here we consider no explicit permutations on $C \cup D$. Only, in step 3 above, we indirectly check for an existence of a permutation f_ϱ on $C \cup D$ witnessing consistency of $\varrho \in \Gamma$ on the central cliques. We now briefly explain why this check is not a trivial task, and subsequently describe how it is done.

So, let $\varrho \in \Gamma$ be an automorphism of the colored poset R from step 2 of Procedure 5.1, and assume ϱ swaps P and Q in R. Since ϱ respects the colors in R, we have that for every $X_i \in \mathcal{X}$ and $Y_j = \varrho(X_i)$, the graph $G[C \cup X_i]$ is isomorphic to $H[D \cup Y_j]$. Unfortunately, this is not yet enough to have G isomorphic to H. Consider, for example, the case illustrated by posets P and Q from Figure 4; the components in $\mathcal{X}$ and $\mathcal{Y}$ are all singleton vertices, and hence the colors (isomorphism types) are fully determined by the cardinality of the shown attachment lists. Even though P and Q can be swapped respecting inclusion and the colors, the graphs G and H represented by them are not isomorphic. The reason is that the intersection of some of the attachment lists (here of the blue elements in Figure 4) is of different cardinality in P than in Q. We will later show, in Lemma 5.4, that it is enough to additionally ensure that problems with cardinalities of intersections of attachment sets like this one do not happen, to claim that G and H are indeed isomorphic.

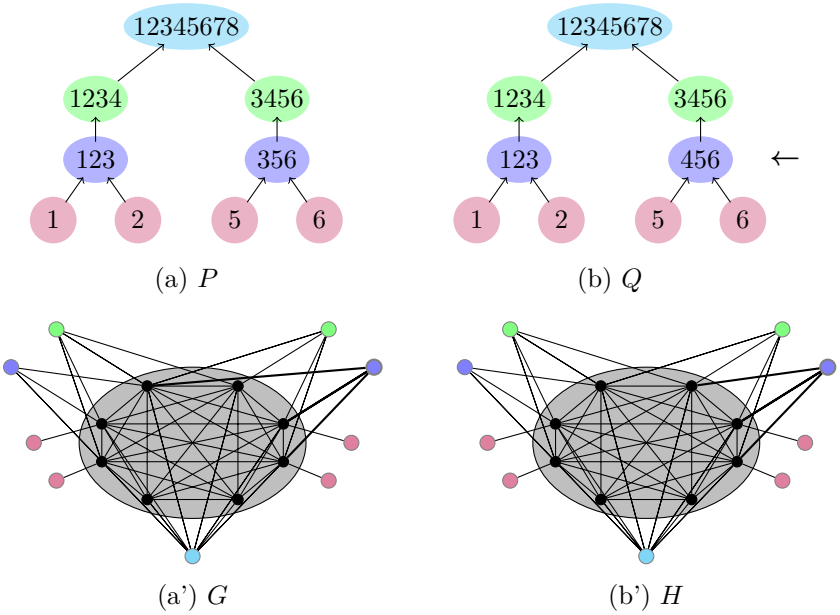

Figure 4 Two isomorphic colored posets (a) P and (b) Q. These come from graphs (a') G and (b') H with the central clique $C = D = \{1, 2, \ldots, 8\}$ (shaded gray), such that the components of $G - C$ (of $H - C$) are 9 singleton vertices having attachments in exactly the listed vertices of C. However, G and H are not isomorphic since the two components (blue) in $G - C$ with size-3 attachment have a common neighbor (3), while the analogous two components (again blue and thick in the graphs) in $H - C$ have no common neighbor.

Computing the automorphism group of a poset of bounded width

For step 2 of Procedure 5.1, we show that the posets of width d are a special case of so-called d-bounded color multiplicity graphs, whose automorphisms can be computed efficiently by Theorem 5.2. Again, we refer to the full paper for additional details.

A *d-bounded color multiplicity graph* is a graph G whose vertex set is arbitrarily partitioned into k color classes $V(G) = V_1 \cup \ldots \cup V_k$ (each V_i is just any subset of $V(G)$). The number k of colors is arbitrary, but for all $1 \leq i \leq k$ we have $|V_i| \leq d$. This is a classical result:

▶ **Theorem 5.2** (Babai [2], with Furst, Hopcroft and Luks [14]). *Let G be a d-bounded color multiplicity graph. The color-preserving automorphism group of G (i.e., its generators) can be determined by an **FPT**-time algorithm with the parameter d.*

Consider a poset R of width $\leq d$ and the levels $L_1, \ldots, L_k$ of R, where $|L_i| \leq d$ for $1 \leq i \leq k$. Note that any automorphism of R preserves the levels. Having the levels L_i as color classes, we may forget about the edge directions between comparable elements of R (as it is implicit from lower to higher levels), and so view R as a d-bounded color multiplicity graph. Hence we immediately get:

▶ **Corollary 5.3** ([2, 14]). *The automorphism group of a poset R of width d can be determined by an **FPT**-time algorithm with the parameter d.* ◀

Checking consistency of a poset automorphism on the central cliques

Now we move to step 3 of Procedure 5.1. To precisely specify what we are going to check there – about consistency of an automorphism ϱ of the combined poset $R = P \uplus Q$, we need a few preliminary facts. For a component $Z \in \mathcal{X} \cup \mathcal{Y}$ (of $K - (C \cup D) = (G - C) \uplus (H - D)$), the $a \geq 1$ distinct neighborhoods (attachment sets) of vertices of Z in $C \cup D$ form a sequence $N_1(Z), \ldots, N_a(Z) \subseteq C \cup D$ ordered by strict inclusion $N_1(Z) \subsetneq N_2(Z) \subsetneq \cdots \subsetneq N_a(Z)$, and we denote this family by $\mathcal{N}_{C \cup D}(Z) := \{N_1(Z), \ldots, N_a(Z)\}$.

We call the *multiset* of sets $\mathcal{U} := \biguplus_{Z \in \mathcal{X} \cup \mathcal{Y}} \mathcal{N}_{C \cup D}(Z)$ the *attachment collection* of $\mathcal{X} \cup \mathcal{Y}$ of our graph K ($\mathcal{U}$ is a multiset since the same attachment set may occur several times in $\mathcal{U}$ if the occurrences come from distinct bridges of $C \cup D$ which are incomparable in R). Recall the notation $K(Z)$ from Procedure 5.1. If an automorphism ϱ of R maps Z to $Z' = \varrho(Z)$ then, in particular, $K(Z)$ is isomorphic to $K(Z')$. While there may exist different isomorphisms from $K(Z)$ to $K(Z')$, they all define (because of the inclusion order) the same unique mapping of the attachment sets from $\mathcal{N}_{C \cup D}(Z)$ to $\mathcal{N}_{C \cup D}(Z')$. So, the automorphism ϱ of R induces a unique corresponding permutation on $\mathcal{U}$, denoted here by $\tilde{\varrho}$.

In general, for a set family $\mathcal{U}$ we call a *cardinality Venn diagram* of $\mathcal{U}$ the vector $(\ell_{\mathcal{U},\mathcal{U}_1} : \emptyset \neq \mathcal{U}_1 \subseteq \mathcal{U})$ such that $\ell_{\mathcal{U},\mathcal{U}_1} := \left| \bigcap_{A \in \mathcal{U}_1} A \setminus \bigcup_{B \in \mathcal{U} \setminus \mathcal{U}_1} B \right|$. That is, we record the cardinality of every internal cell of the Venn diagram of $\mathcal{U}$. Let $\tilde{\varrho}(\mathcal{U}_1) = \{\tilde{\varrho}(A) : A \in \mathcal{U}_1\}$ for $\mathcal{U}_1 \subseteq \mathcal{U}$.

It is reasonably easy to see that an automorphism ϱ of R is consistent on the central cliques $C \cup D$ of K, if and only if the corresponding permutation $\tilde{\varrho}$ on the attachment collection $\mathcal{U}$ of $\mathcal{X} \cup \mathcal{Y}$ preserves the values of all cells of the cardinality Venn diagram of $\mathcal{U}$. This is precisely formulated as follows:

▶ **Lemma 5.4.** *Let $K = G \uplus H$ and C, D, the sets $\mathcal{X}$ and $\mathcal{Y}$, and the posets P, Q and R (on the ground set $C \cup D$) be as in Procedure 5.1. Let $\mathcal{U}$ be the attachment collection of $\mathcal{X} \cup \mathcal{Y}$ in K, and assume an automorphism ϱ of R and the corresponding permutation $\tilde{\varrho}$ of $\mathcal{U}$.*

There is an automorphism f of K such that $f(C \cup D) = C \cup D$ and, for every component $Z \in \mathcal{X} \cup \mathcal{Y}$, f maps $V(Z)$ to $V(\varrho(Z))$, if and only if the cardinality Venn diagrams of $\mathcal{U}$ and of $\tilde{\varrho}(\mathcal{U})$ are the same, meaning that $\ell_{\mathcal{U},\mathcal{U}_1} = \ell_{\mathcal{U},\tilde{\varrho}(\mathcal{U}_1)}$ for all $\emptyset \neq \mathcal{U}_1 \subseteq \mathcal{U}$.

Proof. $\Rightarrow$ Suppose that there exists an automorphism f of K such that $f(C \cup D) = C \cup D$ and, for every component $Z \in \mathcal{X} \cup \mathcal{Y}$, f maps the vertices of Z to the vertices of $\varrho(Z)$. Then, each pair $K(Z)$ and $K(\varrho(Z))$ are isomorphic interval graphs, and if there are a distinct neighborhoods $N_1(Z), \ldots, N_a(Z)$ of vertices of Z in $C \cup D$, then there are a neighborhoods $N_1(\varrho(Z)), \ldots, N_a(\varrho(Z))$ of vertices of $\varrho(Z)$ in $C \cup D$ determined by the restriction of f to $C \cup D$. Since f is an automorphism of K, indeed, $\tilde{\varrho}(N_i(Z)) = N_i(\varrho(Z))$ for $1 \leq i \leq a$. Moreover, by our assumption, if $v \in N_i(Z)$ then $f(v) \in N_i(\varrho(Z))$, and vice versa.

Now consider arbitrary $\emptyset \neq \mathcal{U}_1 \subseteq \mathcal{U}$. By the previous; if $v \in \bigcap_{A \in \mathcal{U}_1} A \setminus \bigcup_{B \in \mathcal{U} \setminus \mathcal{U}_1} B$, then $f(v) \in \bigcap_{A \in \mathcal{U}_1} \tilde{\varrho}(A) \setminus \bigcup_{B \in \mathcal{U} \setminus \mathcal{U}_1} \tilde{\varrho}(B) = \bigcap_{A \in \tilde{\varrho}(\mathcal{U}_1)} A \setminus \bigcup_{B \in \mathcal{U} \setminus \tilde{\varrho}(\mathcal{U}_1)} B$ (since $\tilde{\varrho}$ is a permutation of $\mathcal{U}$), and vice versa. Consequently, $\ell_{\mathcal{U},\mathcal{U}_1} = \ell_{\mathcal{U},\tilde{\varrho}(\mathcal{U}_1)}$ for all $\emptyset \neq \mathcal{U}_1 \subseteq \mathcal{U}$.

$\Leftarrow$ Suppose that $\ell_{\mathcal{U}_1} = \ell_{\tilde{\varrho}(\mathcal{U}_1)}$ holds for all $\emptyset \neq \mathcal{U}_1 \subseteq \mathcal{U}$. Then, in particular, for every such $\mathcal{U}_1$ there exists a bijection from $\bigcap_{A \in \mathcal{U}_1} A \setminus \bigcup_{B \in \mathcal{U} \setminus \mathcal{U}_1} B$ to $\bigcap_{A \in \tilde{\varrho}(\mathcal{U}_1)} A \setminus \bigcup_{B \in \mathcal{U} \setminus \tilde{\varrho}(\mathcal{U}_1)} B$. The composition of these bijections results in a permutation f_0 of $C \cup D$. Informally stating, f_0 respects all attachment sets of all components $Z \in \mathcal{X} \cup \mathcal{Y}$ under the permutation ϱ. Hence, for every $Z \in \mathcal{X} \cup \mathcal{Y}$, there is an isomorphism f_Z from $K(Z)$ to $K(\varrho(Z))$ which extends f_0. And, since the members of $\mathcal{X} \cup \mathcal{Y}$ are pairwise disjoint components of $K - (C \cup D)$, the composition of all f_Z over $Z \in \mathcal{X} \cup \mathcal{Y}$ is well-defined and it is an automorphism of the graph K satisfying the desired properties. ◀

At first sight, the condition of Lemma 5.4 may not seem efficient since $\mathcal{U}$ has up to $2n$ attachment sets, and so an exponential number of Venn diagram cells. Though, only $\leq 2n$ of the cells may be nonempty since the ground set of $\mathcal{U}$ is of cardinality $|C \cup D| \leq 2n$, and so we can handle the situation as follows:

▶ **Lemma 5.5.** *For any $\mathcal{U}' \subseteq \mathcal{U}$ such that $\tilde{\varrho}(\mathcal{U}') = \mathcal{U}'$, one can in $\mathcal{O}(n^2)$ time test whether the equalities $\ell_{\mathcal{U}',\mathcal{U}_1} = \ell_{\mathcal{U}',\tilde{\varrho}(\mathcal{U}_1)}$ hold for all $\emptyset \neq \mathcal{U}_1 \subseteq \mathcal{U}'$.*

Proof. We loop through all vertices w of $C \cup D$, and for each w we record in $\mathcal{O}(n)$ time to which of the sets in $\mathcal{U}'$ this w belongs to. Summing the obtained records at the end precisely gives the $\mathcal{O}(n)$ nonzero values $\ell_{\mathcal{U}',\mathcal{U}_1}$ over $\emptyset \neq \mathcal{U}_1 \subseteq \mathcal{U}'$.

We analogously compute the $\mathcal{O}(n)$ nonzero values $\ell_{\mathcal{U}',\tilde{\varrho}(\mathcal{U}_1)}$ over $\mathcal{U}_1$, and then compare the two sets of values with respect to each $\mathcal{U}_1$ and matching $\tilde{\varrho}(\mathcal{U}_1)$. ◀

Computing the subgroup of consistent poset automorphisms

Knowing how to efficiently test whether an automorphism $\varrho \in \Gamma$ in step 3 of Procedure 5.1 is consistent (Lemmas 5.4 and 5.5), we would like to finish a computation of the subgroup $\Gamma' \subseteq \Gamma$. This, however, cannot be done directly by processing all members of the group Γ which can be exponentially large. Instead, inspired by famous Babai's "tower-of-groups" procedure (cf. [2] and Theorem 5.2), we iteratively compute a chain of subgroups $\Gamma = \Gamma_0 \supseteq \Gamma_1 \supseteq \ldots \supseteq \Gamma_h = \Gamma'$ (where $h = \mathcal{O}(nd \log d)$) leading to the result. Here, by "computing a group" we mean to output a set of its generators.

There are two important ingredients making this computation work. First, we look for a manageable combinatorially defined "gradual refinement" of the condition tested by Lemma 5.4. Our intention is to define Γ_i as the subgroup of Γ respecting the i-th step of this refinement. By manageable we mean that the ratio of orders (sizes) of consequent groups Γ_i and Γ_{i+1} in the chain is always bounded and, at the same time, that the number of refinement steps (h) is not too big. Second, having such manageable refinement steps, we then stepwise apply another classical result (which, by the way, also leads to Theorem 5.2):

▶ **Theorem 5.6** (Furst, Hopcroft and Luks [13, Cor. 1])**.** *Let Π be a permutation group given by its generators, and Π_1 be any subgroup of Π such that one can test in polynomial time whether $\pi \in \Pi_1$ for any $\pi \in \Pi$ (membership test). If the ratio $|\Pi|/|\Pi_1|$ is bounded by a function of a parameter d, then a set of generators of Π_1 can be computed in **FPT** time (with respect to d).*

Due to space restriction, we have to leave additional comments on this result to the full paper.

Returning back to the first ingredient, we closely analyze what happens if an automorphism $\varrho \in \Gamma$ does not pass the cardinality Venn diagram test of Lemma 5.4. In a nutshell, every such failure must be witnessed by a subcollection of at most d sets of $\mathcal{U}$. Precisely:

▶ **Lemma 5.7.** *Let $\mathcal{U}$ be a set family and $\tilde{\varrho}$ be a permutation of $\mathcal{U}$. If there exists $\mathcal{U}_1$ such that $\emptyset \neq \mathcal{U}_1 \subseteq \mathcal{U}$ and $\ell_{\mathcal{U},\mathcal{U}_1} \neq \ell_{\mathcal{U},\tilde{\varrho}(\mathcal{U}_1)}$, then there exist $\mathcal{U}_2, \mathcal{U}_3 \subseteq \mathcal{U}$ such that $|\mathcal{U}_2| \leq 2$ or $\mathcal{U}_2$ is an antichain in inclusion, $\emptyset \neq \mathcal{U}_3 \subseteq \mathcal{U}_2$ and $\ell_{\mathcal{U}_2,\mathcal{U}_3} \neq \ell_{\tilde{\varrho}(\mathcal{U}_2),\tilde{\varrho}(\mathcal{U}_3)}$.*

In our case, $\mathcal{U}_2$ is an antichain of attachment sets of one of G or H, and hence $|\mathcal{U}_2| \leq d$.

For simplicity, we say that $\mathcal{U}' \subseteq \mathcal{U}$ is *Venn-good* if $\ell_{\mathcal{U}',\mathcal{U}_0} = \ell_{\tilde{\varrho}(\mathcal{U}'),\tilde{\varrho}(\mathcal{U}_0)}$ holds true for all $\emptyset \neq \mathcal{U}_0 \subseteq \mathcal{U}'$, and we call $\mathcal{U}_0$ a *witness* (of $\mathcal{U}'$ not being Venn-good) if $\ell_{\mathcal{U}',\mathcal{U}_0} \neq \ell_{\tilde{\varrho}(\mathcal{U}'),\tilde{\varrho}(\mathcal{U}_0)}$. Recalling that the permutation $\tilde{\varrho}$ of $\mathcal{U}$ is determined by an automorphism ϱ of our poset R, we also more precisely say that $\mathcal{U}'$ is Venn-good *for* this automorphism ϱ.

Proof. Notice that $\mathcal{U}$ itself is not Venn-good, and $\mathcal{U}_1$ is a witness. Choose $\mathcal{U}_2 \subseteq \mathcal{U}$ such that $\mathcal{U}_2$ is *not* Venn-good and it is minimal such by inclusion, and assume (for a contradiction) that there are $A_1, A_2 \in \mathcal{U}_2$ such that $A_1 \subseteq A_2$. If $\tilde{\varrho}(A_1) \not\subseteq \tilde{\varrho}(A_2)$, then already $\mathcal{U}_2 := \{A_1, A_2\}$ is not Venn-good (with a witness $\{A_1\}$), and so let $\tilde{\varrho}(A_1) \subseteq \tilde{\varrho}(A_2)$.

Let $\mathcal{U}_3$ be a witness of $\mathcal{U}_2$ not being Venn-good, and for $j = 2, 3$ denote: $\mathcal{U}_j^0 := \mathcal{U}_j \setminus \{A_1, A_2\}$, $\mathcal{U}_j^1 := \big(\mathcal{U}_j \cup \{A_1\}\big) \setminus \{A_2\}$, $\mathcal{U}_j^2 := \big(\mathcal{U}_j \cup \{A_2\}\big) \setminus \{A_1\}$, $\mathcal{U}_j^3 := \mathcal{U}_j \cup \{A_1, A_2\}$. By our minimality assumption, all three subfamilies $\mathcal{U}_2^0$, $\mathcal{U}_2^1$ and $\mathcal{U}_2^2$ are Venn-good. We first easily derive

$$\begin{aligned} \ell_{\mathcal{U}_2,\mathcal{U}_3^0} = \ell_{\mathcal{U}_2\setminus\{A_1\},\mathcal{U}_3^0} = \ell_{\mathcal{U}_2^2,\mathcal{U}_3^0} &= \ell_{\tilde{\varrho}(\mathcal{U}_2^2),\tilde{\varrho}(\mathcal{U}_3^0)} = \ell_{\tilde{\varrho}(\mathcal{U}_2),\tilde{\varrho}(\mathcal{U}_3^0)}\,, \\ \ell_{\mathcal{U}_2,\mathcal{U}_3^1} = 0 &= 0 = \ell_{\tilde{\varrho}(\mathcal{U}_2),\tilde{\varrho}(\mathcal{U}_3^1)}\,, \\ \ell_{\mathcal{U}_2,\mathcal{U}_3^3} = \ell_{\mathcal{U}_2\setminus\{A_2\},\mathcal{U}_3^3\setminus\{A_2\}} = \ell_{\mathcal{U}_2^1,\mathcal{U}_3^1} &= \ell_{\tilde{\varrho}(\mathcal{U}_2^1),\tilde{\varrho}(\mathcal{U}_3^1)} = \ell_{\tilde{\varrho}(\mathcal{U}_2),\tilde{\varrho}(\mathcal{U}_3^3)}\,. \end{aligned}$$

Then, using trivial $\ell_{\mathcal{U}_2^0,\mathcal{U}_3^0} = \ell_{\mathcal{U}_2,\mathcal{U}_3^0} + \ell_{\mathcal{U}_2,\mathcal{U}_3^1} + \ell_{\mathcal{U}_2,\mathcal{U}_3^2} + \ell_{\mathcal{U}_2,\mathcal{U}_3^3}$ and its counterpart under $\tilde{\varrho}$,

$$\begin{aligned} \ell_{\mathcal{U}_2,\mathcal{U}_3^2} &= \ell_{\mathcal{U}_2^0,\mathcal{U}_3^0} - \ell_{\mathcal{U}_2,\mathcal{U}_3^0} - 0 - \ell_{\mathcal{U}_2,\mathcal{U}_3^3} \\ &= \ell_{\tilde{\varrho}(\mathcal{U}_2^0),\tilde{\varrho}(\mathcal{U}_3^0)} - \ell_{\tilde{\varrho}(\mathcal{U}_2),\tilde{\varrho}(\mathcal{U}_3^0)} - \ell_{\tilde{\varrho}(\mathcal{U}_2),\tilde{\varrho}(\mathcal{U}_3^3)} = \ell_{\tilde{\varrho}(\mathcal{U}_2),\tilde{\varrho}(\mathcal{U}_3^2)}. \end{aligned}$$

However, $\mathcal{U}_3 \in \{\mathcal{U}_3^0, \mathcal{U}_3^1, \mathcal{U}_3^2, \mathcal{U}_3^3\}$, and so one of the latter four equalities contradicts the assumption that $\mathcal{U}_3$ witnessed $\mathcal{U}_2$ not being Venn-good. ◀

▶ **Corollary 5.8.** *Let a poset R, its automorphism ϱ, attachment collection $\mathcal{U}$ and permutation $\tilde{\varrho}$ of $\mathcal{U}$ be as in Lemma 5.4. We have that $\mathcal{U}$ is Venn-good (wrt. $\tilde{\varrho}$), if and only if every $\mathcal{U}'$ is Venn-good, where $\mathcal{U}' \subseteq \mathcal{U}$ is the subcollection of attachment sets of the union of some (any) d levels of the poset R.* ◀

Corollary 5.8 shows a clear road to computing the subgroup $\Gamma' \subseteq \Gamma$ in step 3 of Procedure 5.1: in every refinement step of a chain $\Gamma = \Gamma_0 \supseteq \Gamma_1 \supseteq \ldots \supseteq \Gamma_h = \Gamma'$, we add a requirement that the subcollection of attachment sets of some d-tuple of levels of the poset R is Venn-good for every member of the next subgroup. All these steps are manageable (cf. Theorem 5.6); the ratio $|\Gamma_i|/|\Gamma_{i+1}|$ is bounded from above by the maximum number of subpermutations of Γ_i on the respective d levels, which is $\leq (2d)!^d$. However, since the height of R is $\Theta(n)$, we cannot afford to check all d-tuples of levels this way. Fortunately, it is also not necessary by the following argument.

By Lagrange's group theorem, $|\Gamma_{i+1}|$ divides $|\Gamma_i|$, and so either $\Gamma_{i+1} = \Gamma_i$ or $|\Gamma_{i+1}| \leq \frac{1}{2}|\Gamma_i|$. Hence the number of strict refinement steps in our chain of subgroups is $h \leq \log|\Gamma|$, which is affordable in an **FPT** time algorithm. Since $|\Gamma| \leq (2d)!^n$, we get $h = \mathcal{O}(nd \log d)$. Furthermore, d-tuples of levels giving such suitable h strict refinement steps can be, one at each step, computed by a procedure outlined as follows:

- **i.** Recall that the levels of R are $L_1, L_2, \ldots, L_k$. Find minimal index j_1 such that the attachment sets of $L_1 \cup \cdots \cup L_{j_1}$ are not Venn-good for some generator of Γ_i.
- **ii.** Analogously, find minimal $j_2 < j_1$ such that the attachment sets of $L_1 \cup \cdots \cup L_{j_2} \cup L_{j_1}$ are not Venn-good for some generator of Γ_i.
- **iii.** Find analogously indices $j_3, j_4, \ldots, j_a$, until we stop with $j_a = 1$, or we reach $a = d$ (in the former case, to be even, we may add arbitrary $d - a$ other levels to our collection).

Wrapping up with Procedure 5.1

▶ **Theorem 5.9.** *The isomorphism problem of S_d-graphs can be solved by an **FPT**-time algorithm with the fixed parameter d.*

Due to space restrictions, we can only refer to the full paper for a detailed listing of the steps of the complete isomorphism testing procedure.

Proof. First of all, if given graphs G and H are not connected, we add a universal vertex to each (adjacent to all other vertices), which does not change the problem.

Then we apply Procedure 5.1. This actually runs $\mathcal{O}(n)$ iterations of choices of C and D; we first greedily find any valid central clique C of an S_d-representation of G, and then iterate all maximal cliques $D \subseteq H$. In each iteration, we routinely compute in polynomial time the sets of components $\mathcal{X}$ and $\mathcal{Y}$, and the posets P and Q on them and $R = P \uplus Q$. If any of P, Q has width greater than d, then we reject this iteration. Similarly, we reject the iteration if any of the graphs $K(Z)$ for $Z \in \mathcal{X} \cup \mathcal{Y}$ is not interval [5]. Otherwise, we compute colors on R such that $Z, Z' \in \mathcal{X} \cup \mathcal{Y}$ on the same level of R receive the same color iff $K(Z) \simeq K(Z')$, using the isomorphism algorithm of [5]. This finishes step 1.

Step 2 – computing the automorphism group Γ of R, is done by Corollary 5.3.

Step 3 – finding the subgroup $\Gamma' \subseteq \Gamma$, is accomplished by an iterated application of Theorem 5.6; the refinement steps are defined by d-tuples of levels of R according to Corollary 5.8 and the above outlined procedure for finding them, and there are $\mathcal{O}(nd \log d)$ such steps. The membership test in Theorem 5.6 is provided by Lemma 5.5.

Finally, we straightforwardly check in step 4 on the generators of Γ' whether some of them maps an element of P to an element of Q.

If any iteration of Procedure 5.1 succeeds in step 4, then, by Lemma 5.4, there exists an automorphism of the graph $K = G \uplus H$ which moreover swaps G and H. Then $G \simeq H$. Conversely, assume $G \simeq H$. Since G is an S_d-graph, we find a maximal central clique $C \subseteq G$, and since all maximal cliques $D \subseteq H$ are tried, we get into an iteration with D being the isomorphic image of C. Then the posets P and Q (wrt. C, D) must be isomorphic respecting their colors, which follows from $G \simeq H$. Therefore, there exists an automorphism in Γ', and also such a generator $\varrho \in \Gamma'$ since G and H are connected, swapping P and Q. This ϱ, in particular, preserves the computed cardinality Venn diagram by Lemma 5.4. Some iteration hence succeeds and returns that $G \simeq H$. ◀

▶ Remark 5.10. We do not explicitly state the runtime in Theorem 5.9, partly due to space restrictions and partly since it is not really useful and since neither [13] which we use states explicit runtime. Here we briefly remark that Procedure 5.1 loops $\mathcal{O}(n)$ times with suitable C and different choices of D, analogously to the procedure of Theorem 3.1, and this initial setup of the procedure altogether takes time $\mathcal{O}(d^2n^3)$. Then we have to account for $\mathcal{O}(n)$ calls to steps 2 and 3 of Procedure 5.1, that is, $\mathcal{O}(n)$ computations of the subgroups Γ and Γ'. Each time this part is dominated by step 3 which performs $\mathcal{O}(nd \log d)$ calls to the algorithm of Theorem 5.6 [13] in order to compute Γ' from Γ. Reading the fine details of [13], and adjusting it (the "sift table") to our setting, leads to an estimate of $\mathcal{O}(n^3) \cdot d!^{\mathcal{O}(d)}$ for each of these calls. After summarizing, we get the total estimate of $\mathcal{O}(n^5) \cdot d!^{\mathcal{O}(d)}$.

6 Conclusions

Our **FPT** algorithm for isomorphism of S_d-graphs is a natural extension of isomorphism of interval graphs. It is also natural to ask for further extensions to H-graphs. However, already for $H = K_3$ (the well-known case of circular-arc graphs), efficient isomorphism testing is a highly nontrivial task, see e.g. Krawczyk [18]. With H containing at least two cycles, moreover, the problem becomes GI-complete. Thus we would like to consider the case of $H = T$ being a tree.

It is relatively straightforward (though technical) to extend our approach to an XP algorithm for isomorphism testing of T-graphs, with the number of leaves of T as the parameter (a straightforward extension can not be **FPT**-time since we need to "guess" the right central clique in each internal node of T). We, however, aim to get an **FPT**-time algorithm for T-graph isomorphism which is the subject of future research.

References

1 A. V. Aho, J. E. Hopcroft, and J. D. Ullman. The design and analysis of computer algorithms. *Addison–Wesley, Reading, Mass. 1974*, 1974.

2 L. Babai. Monte carlo algorithms in graph isomorphism testing. *Tech. Rep. 79-10, Université de Montréal*, 1979. 42 pages.

3 L. Babai. Graph isomorphism in quasipolynomial time [extended abstract]. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016*, pages 684–697. ACM, 2016. doi:10.1145/2897518.2897542.

4 M. Biró, M. Hujter, and Z. Tuza. Precoloring extension. i. interval graphs. *Discrete Mathematics **100***, pages 267–279, 1992.

5 K. S. Booth and G. S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *J. Comput. Syst. Sci.*, 13:335–379, 1976.

6 A. Bouland, A. Dawar, and E. Kopczynski. On tractable parameterizations of graph isomorphism. In *Parameterized and Exact Computation, IPEC 2012. Proceedings*, volume 7535 of *Lecture Notes in Computer Science*, pages 218–230. Springer, 2012. doi:10.1007/978-3-642-33293-7_21.

7 S. Chaplick, M. Töpfer, J. Voborník, and P. Zeman. On H-topological intersection graphs. In *WG*, volume 10520 of *Lecture Notes in Computer Science*, pages 167–179. Springer, 2017.

8 S. Chaplick and P. Zeman. Combinatorial problems on H-graphs. *Electronic Notes in Discrete Mathematics*, pages 61:223–229, 2017.

9 F. R. K. Chung. On the cutwidth and the topological bandwidth of a tree. *SIAM J. Alg. Discr. Meth.*, 6:268–277, 1985.

10 R. Downey and M. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

11 P. A. Fejer and D. A. Simovici. Partially ordered sets. *In: Mathematical Foundations of Computer Science. Texts and Monographs in Computer Science. Springer, New York, NY*, 1991.

12 F. V. Fomin, P. A. Golovach, and J. F. Raymond. On the tractability of optimization problems on H-graphs. In *26th Annual European Symposium on Algorithms, ESA 2018*, volume 112 of *LIPIcs*, pages 30:1–30:14. Schloss Dagstuhl, 2018. doi:10.4230/LIPIcs.ESA.2018.30.

13 M. L. Furst, J. E. Hopcroft, and E. M. Luks. Polynomial-time algorithms for permutation groups. *21st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 36–41, 1980.

14 M. L. Furst, J. E. Hopcroft, and E. M. Luks. A subexponential algorithm for trivalent graph isomorphism. In *Proc. 11th Southeastern Conf. Combinatorics, Graph Theory, and Computing, Congressum Numerantium 3*, 1980.

15 F. Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory, Series B*, 16:47–56, 1974.

16 M. Grohe, D. Neuen, P. Schweitzer, and D. Wiebking. An improved isomorphism test for bounded-tree-width graphs. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018*, volume 107 of *LIPIcs*, pages 67:1–67:14. Schloss Dagstuhl, 2018. doi:10.4230/LIPIcs.ICALP.2018.67.

17 J. E. Hopcroft and J. K. Wong. Linear time algorithm for isomorphism of planar graphs. In *STOC*, pages 172–184, 1974.

18 T. Krawczyk. Testing isomorphism of circular-arc graphs - Hsu's approach revisited. *CoRR*, abs/1904.04501, 2019. `arXiv:1904.04501`.

19 D. J. Rose, G. Lueker, and R. E. Tarjan. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on Computing*, 5:266–283, 1976.

20 V. N. Zemlyachenko, N. M. Korneenko, and R. I. Tyshkevich. Graph isomorphism problem. *J. of Soviet Mathematics*, 29:1426–1481, 1985.