Partial and Simultaneous Transitive Orientations via Modular Decompositions

Miriam Münch
Faculty of Computer Science and Mathematics, Universität Passau, Germany

Ignaz Rutter
Faculty of Computer Science and Mathematics, Universität Passau, Germany

Peter Stumpf
Faculty of Computer Science and Mathematics, Universität Passau, Germany

Abstract

A natural generalization of the recognition problem for a geometric graph class is the problem of extending a representation of a subgraph to a representation of the whole graph. A related problem is to find representations for multiple input graphs that coincide on subgraphs shared by the input graphs. A common restriction is the sunflower case where the shared graph is the same for each pair of input graphs. These problems translate to the setting of comparability graphs where the representations correspond to transitive orientations of their edges. We use modular decompositions to improve the runtime for the orientation extension problem and the sunflower orientation problem to linear time. We apply these results to improve the runtime for the partial representation problem and the sunflower case of the simultaneous representation problem for permutation graphs to linear time. We also give the first efficient algorithms for these problems on circular permutation graphs.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Mathematics of computing → Graph algorithms

Keywords and phrases representation extension, simultaneous representation, comparability graph, permutation graph, circular permutation graph, modular decomposition

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2022.51

Related Version *Full Version*: https://doi.org/10.48550/arXiv.2209.13175

Funding Funded by grant RU 1903/3-1 of the German Research Foundation (DFG).

1 Introduction

Representations and drawings of graphs have been considered since graphs have been studied [28]. A *geometric intersection representation* of a graph $G = (V, E)$ with regards to a class of geometric objects $\mathcal{C}$, is a map $R\colon V \to \mathcal{C}$ that assigns objects of $\mathcal{C}$ to the vertices of G such that G contains an edge uv if and only if the intersection $R(u) \cap R(v)$ is non-empty. In this way, the class $\mathcal{C}$ gives rise to a class of graphs, namely the graphs that admit such a representation. As an example, consider *permutation diagrams* where $\mathcal{C}$ consists

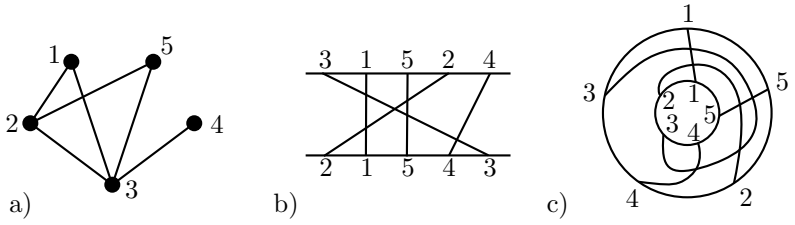

Figure 1 (a) a graph G with (b) a permutation diagram and (c) a circular permutation diagram.

of segments connecting two parallel lines ℓ_1, ℓ_2, see Figure 1b, which defines the class Perm of *permutation graphs*. Similarly, the class CPerm of *circular permutation graphs* is obtained by replacing ℓ_1, ℓ_2 with concentric circles and the geometric objects with curves from ℓ_1 to ℓ_2 that pairwise intersect at most once; see Figure 1c.

A key problem in this context is the *recognition problem*, which asks whether a given graph admits such a representation for a fixed class $\mathcal{C}$. Klavík et al. introduced the *partial representation extension problem* (RepExt($\mathcal{C}$)) for intersection graphs where a representation R' is given for a subset of vertices $W \subseteq V$ and the question is whether R' can be extended to a representation R of G, in the sense that $R|_W = R'$ [28]. They showed that RepExt can be solved in linear time for interval graphs. The problem has further been studied for proper/unit interval graphs [26], function and permutation graphs (Perm) [25], circle graphs [11], chordal graphs [27], and trapezoid graphs [29]. Related extension problems have also been considered, e.g., for planar topological [1, 24] and straight-line [32] drawings, for 1-planar drawings [14], for contact representations [10], and for rectangular duals [12].

A related problem is the *simultaneous representation problem* (SimRep($\mathcal{C}$)) where input graphs $G_1 \ldots, G_r$ that may share subgraphs are given and the question is whether they have representations $R_1, \ldots, R_r$ such that for $i, j \in \{1, \ldots, r\}$ the shared graph $H = G_i \cap G_j$ has the same representation in R_i and R_j, i.e., $R_i|_{V(H)} = R_j|_{V(H)}$. If more than two input graphs are allowed, usually the *sunflower case* (SimRep*($\mathcal{C}$)) is considered, where the shared graph $H = G_i \cap G_j$ is the same for any $i \neq j \in \{1, \ldots, r\}$. I.e., here the question is whether H has a representation that can be simultaneously extended to $G_1, \ldots, G_r$. Simultaneous representations were first studied in the context of planar drawings [4, 9], where the goal is to embed each input graph without edge crossings while shared subgraphs have the same induced embedding. Unsurprisingly, many variants are NP-complete [17, 35, 2, 15].

Motivated by applications in visualization of temporal relationships, and for overlapping social networks or schedules, DNA fragments of similar organisms and adjacent layers on a computer chip, Jampani and Lubiw introduced the problem SimRep for intersection graphs [23]. They provided polynomial-time algorithms for two chordal graphs and for SimRep*(Perm). They also showed that in general SimRep is NP-complete for three or more chordal graphs. The problem was also studied for interval graphs [22, 5, 6], proper/unit interval graphs [34], circular-arc graphs [6] and circle graphs [11].

Many of the considered graph classes are related to the class Comp of *comparability graphs* [19]. An *orientation* O of a graph $G = (V, E)$ assigns to each edge of G a direction. The orientation O is *transitive* if $uv, vw \in O$ implies $uw \in O$. A comparability graph is a graph for which there is a *transitive orientation*. A *partial orientation* is an orientation of a (not necessariliy induced) subgraph of G. Similar to RepExt and SimRep*, the problems OrientExt and SimOrient for comparability graphs ask for a transitive orientation of a graph that extends a given partial orientation and for transitive orientations that coincide on the shared graph, respectively. The key ingredient for the $O(n^3)$ algorithm solving RepExt(Perm) by Klavík et al. [25] is a polynomial-time solution for OrientExt based on the transitive orientation algorithm by Gilmore and Hoffman [18]. Likewise, the $O(n^3)$ algorithm solving SimRep*(Perm) by Jampani and Lubiw [23] is based on a polynomial-time algorithm for SimOrient based on the transitive orientation algorithm by Golumbic [19].

Contribution and Outline. In Section 2, we introduce modular decompositions which can be used to describe certain subsets of the set of all transitive orientations of a graph, e.g., those that extend a given partial representation. Based on this, we give a simple linear-time algorithm for OrientExt in Section 3. Afterwards, in Section 4, we develop an algorithm

Table 1 Known runtimes on the left and new runtimes on the right. For SimRep* we set $n = \sum_{i=1}^{r} |V(G_i)|$ and $m = \sum_{i=1}^{r} |E(G_i)|$. We use RepExt(Comp) and SimRep*(Comp) to refer to OrientExt and SimOrient, respectively, in a slight abuse of notation.

	RepExt	SimRep*
Comp	$O((n+m)\Delta)$ [25]	$O(nm)$ [23]
Perm	$O(n^3)$ [25]	$O(n^3)$ [23]
CPerm	open	open

	RepExt	SimRep*
Comp	$O(n+m)$	$O(n+m)$
Perm	$O(n+m)$	$O(n+m)$
CPerm	$O(n+m)$	$O(n^2)$

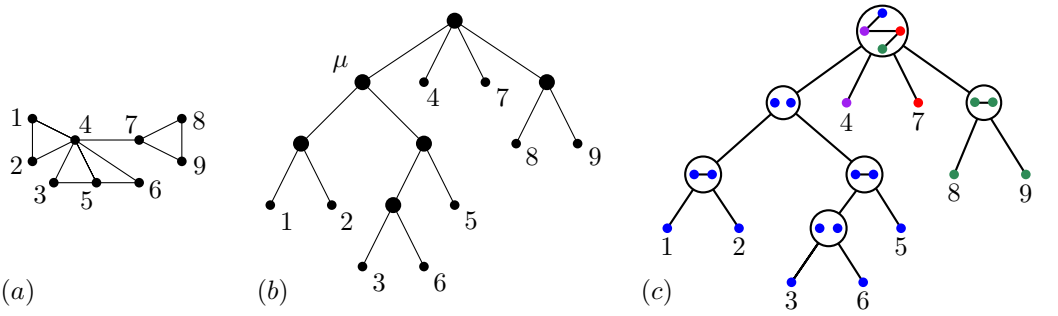

Figure 2 (a) A graph G. (b) The canonical modular decomposition of G with $L(\mu) = \{1, 2, 3, 5, 6\}$. (c) The quotient graphs corresponding to the nodes in the canonical modular decomposition.

for intersecting subsets of transitive orientations represented by modular decompositions and use this to give a linear-time algorithm for SimOrient. In Section 5 we give linear-time algorithms for RepExt(Perm) and SimRep*(Perm), improving over the $O(n^3)$-algorithms of Klavík et al. and Jampani and Lubiw, respectively. We also give the first efficient algorithms for RepExt(CPerm) and SimRep*(CPerm). Table 1 gives an overview of the state of the art and our results. Proofs marked with a $\star$ are omitted due to space restrictions. For the full proofs we refer to the long version [31] .

2 Modular Decompositions

Let $G = (V, E)$ be an undirected graph. We write $G[U]$ for the subgraph induced by a vertex set $U \subseteq V$. For a rooted tree T and a node μ of T, we write $T[\mu]$ for the subtree of T with root μ and $L(\mu)$ for the leaf-set of $T[\mu]$.

A *module* of G is a non-empty set of vertices $M \subseteq V$ such that every vertex $u \in V \setminus M$ is either adjacent to all vertices in M or to none of them. The singleton subsets and V itself are called the *trivial modules*. A module $M \subsetneq V$ is *maximal*, if there exists no module M' such that $M \subsetneq M' \subsetneq V$. If G has at least three vertices and no non-trivial modules, then it is called *prime*. We call a rooted tree T with root ρ and $L(\rho) = V$ a *(general) modular decomposition* for G if for every node μ of T the set $L(\mu)$ is a module; see Figure 2. Observe that for any two nodes $\mu_1, \mu_2 \in T$ such that neither of them is an ancestor of the other, G contains either all edges with one endpoint in $L(\mu_1)$ and one endpoint in $L(\mu_2)$ or none of them. For two vertices $u, v \in V$ we denote the lowest common ancestor of their corresponding leaves in T by $\mathrm{lca}_T(u, v)$. For a set of leaves L, we denote the lowest common ancestor by $\mathrm{lca}_T(L)$.

With each inner node μ of T we associate a *quotient graph* $G[\mu]$ that is obtained from $G[L(\mu)]$ by contracting $L(\nu)$ into a single vertex for each child ν of μ; see Figure 2. In the rest of this paper we identify the vertices of $G[\mu]$ with the corresponding children of μ.

Every edge $uv \in E$ is *represented* by exactly one edge $\text{rep}_T(uv)$ in one of the quotient graphs of T, namely in the quotient graph $G[\mu]$ of the lowest common ancestor μ of u and v. More precisely, if ν and λ are the children of μ with $u \in L(\nu)$ and $v \in L(\lambda)$, then $\text{rep}_T(uv) = \nu\lambda$. For an oriented edge uv, $\text{rep}_T(uv)$ is also oriented towards its endpoint ν with $v \in L(\nu)$. If T is clear from the context, the subscript can be omitted. Let μ be a node in T. For a vertex $u \in L(\mu)$ we denote the child λ of μ with $u \in L(\lambda)$ by $\text{rep}_\mu(u)$.

A node μ in a modular decomposition T is called *empty*, *complete* or *prime* if the quotient graph $G[\mu]$ is *empty*, *complete* or *prime*, respectively. By $K(T)$, $P(T)$ we denote the set of all complete and prime nodes in T, respectively. For every graph G there exists a uniquely defined modular decomposition, that we call *the canonical modular decomposition* of G, introduced by Gallai [16], such that each quotient graph is either prime, complete or empty and, additionally, no two adjacent nodes are both complete or are both empty; see Figure 2. Note that in the literature, these are referred to as modular decompositions, whereas we use that term for general modular decompositions. For a prime node μ in the canonical modular decomposition of G, for every child ν of μ, $L(\nu)$ is a maximal module in $G[L(\mu)]$ and for every maximal module M in $G[L(\mu)]$ there exists a child ν of μ with $L(\nu) = M$. McConnell and Spinrad showed that the canonical modular decomposition can be computed in $O(|V| + |E|)$ time [30]. Let μ be a node in a modular decomposition for G. A *μ-set* U is a subset of $L(\mu)$ that contains for each child λ of μ at most one leaf in $L(\lambda)$. If U contains for every child λ of μ a vertex in $L(\lambda)$, we call it *maximal*.

▶ **Lemma 1** (⋆). *Let T be a modular decomposition of a graph G. After a linear-time preprocessing we can assume that each node of T is annotated with its quotient graph. Moreover, the following queries can be answered in $O(1)$ time:*

1) *Given a non-root node ν of T, find the vertex of the quotient graph of ν's parent that corresponds to ν.*
2) *Given a vertex v in a quotient graph $G[\mu]$, find the child of μ that corresponds to v.*
3) *Given an edge e of G, determine* $\text{rep}(e)$*, the quotient graph that contains* $\text{rep}(e)$*, and which endpoint of* $\text{rep}(e)$ *corresponds to which endpoint of e.*

Additionally, given a node μ in T one can find a maximal μ-set U in $O(|U|)$ time.

Let μ be a node in a modular decomposition T of G and let $\mu_1\mu_2$ be an edge in $G[\mu]$. Let $\vec{T}[G]$ denote an assignment of directions to all edges in $G[\mu]$ for every node μ in T. Such a $\vec{T}[G]$ is *transitive* if it is transitive on every $G[\mu]$. We obtain an orientation of G from an orientation $\vec{T}[G]$ of the quotient graphs of T as follows. Every undirected edge uv in E with $\text{rep}(uv) = \mu_1\mu_2 \in \vec{T}[G]$, is directed from u to v. We say that T *represents* an orientation O of G, if there exists an orientation $\vec{T}[G]$ of the quotient graphs of T that gives us O. We denote the set of all transitive orientations of G represented by T by $\text{to}(T)$. We get an orientation of the quotient graphs of T from an orientation of G, if for each oriented edge $\mu_1\mu_2$, all edges represented by $\mu_1\mu_2$ are oriented from $L(\mu_1)$ to $L(\mu_2)$. Let T now be the canonical modular decomposition of G. Then T represents exactly the transitive orientations of G [16]. It follows that G is a comparability graph if and only if T can be oriented transitively.

If G is a comparability graph, every prime quotient graph $G[\mu] = (V_\mu, E_\mu)$ has exactly two transitive orientations, one the reverse of the other [19], and with the algorithm by McConnell and Spinrad [30] we can compute one of them in $O(|V_\mu| + |E_\mu|)$ time. Hence the time to compute the canonical modular decomposition in which every prime node is labeled with a corresponding transitive orientation is $O(|V| + |E|)$.

3 Transitive Orientation Extension

The *partial orientation extension problem* for comparability graphs OrientExt is to decide for a comparability graph G with a partial orientation W, i.e. an orientation of some of its edges, whether there exists a transitive orientation O of G with $W \subseteq O$. The notion of partial orientations and extensions extends to modular decompositions. We get a partial orientation of the quotient graphs of T from W such that exactly the edges that represent at least one edge in W are oriented and all edges in W that are represented by the same oriented edge $\mu_1\mu_2$, are directed from $L(\mu_1)$ to $L(\mu_2)$.

▶ **Lemma 2** ($\star$). *Let T be the canonical modular decomposition of a comparability graph G and let W be a partial orientation of G that gives us a partial orientation P of the quotient graphs of T. Then W extends to a transitive orientation of G if and only if P extends to a transitive orientation of the quotient graphs of T.*

▶ **Theorem 3** ($\star$). *OrientExt can be solved in linear time.*

Sketch of proof. We confirm that the partial orientation actually gives us a partial orientation P of the quotient graphs of the canonical modular decomposition T. Otherwise we can reject. By Lemma 2 we now just need to check for each node μ of T whether P can be extended to $G[\mu]$. To this end, we use that μ is empty, complete or prime. Since transitive orientations of cliques are total orders and prime graphs have at most two transitive orientations, the existence of an extension can easily be decided in each case. ◀

4 Sunflower Orientations

The idea to solve SimOrient is to obtain for each input graph G_i a restricted modular decomposition of the shared graph H that represents exactly those transitive orientations of H that can be extended to G_i. The restricted modular decompositions can be expressed by constraints for the canonical modular decomposition of H. These constraints are then combined to represent all transitive orientations of H that can be extended to each input graph G_i. With this the solution is straightforward.

Let H be a comparability graph. Then we define a *restricted modular decomposition* (T, D) of H to be a tuple where T is a modular decomposition of H where every node is labeled as complete, empty or prime, such that for every node μ labeled as complete or empty, $H[\mu]$ is complete or empty, respectively, and D is a function that assigns to each prime labeled node μ a transitive orientation D_μ, called *default orientation*. In the following, when referring to the type of a node μ in a restricted modular decomposition, we mean the type that μ is labeled with. A transitive orientation of (T, D), is a transitive orientation of the quotient graphs of T where every prime node μ has orientation D_μ or its reversal D_μ^{-1}. Let $\mathrm{to}(T, D)$ denote the set of transitive orientations of H we get from transitive orientations of (T, D). We say (T, D) *represents* these transitive orientations. Note that for the canonical modular decomposition B of H we have $\mathrm{to}(T, D) \subseteq \mathrm{to}(B)$.

Let G be a comparability graph with an induced subgraph H. A modular decomposition T of G gives us a restricted modular decomposition $(T|_H, D)$ of H as follows; see Figure 3. We obtain $T|_H$ from T by (i) removing all leaves that do not correspond to a vertex of H and then (ii) iteratively contracting all inner nodes of degree at most 2. With a bottom-up traversal we can compute $T|_H$ in time linear in the size of T. A node μ in $T|_H$ *stems from* $\mathrm{lca}_T(L(\mu))$. Every node $\mu \in T|_H$ that stems from a prime node $\mu' \in T$ we label as prime and set D_μ to a transitive orientation of $G[\mu']$ restricted to the edges of H. The remaining nodes are labeled according to the type of their quotient graph. Note that $H[\mu]$ is isomorphic to an induced subgraph of $G[\mu']$.

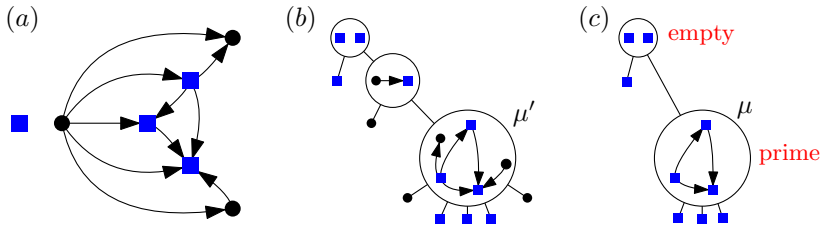

Figure 3 (a) A graph G with an induced subgraph H (blue square vertices). (b) A modular decomposition T of G with a transitive orientation. (c) The restricted modular decomposition $(T|_H, D)$ of H derived from the transitive orientation in (b). Note that $H[\mu]$ is a clique but μ is labeled prime.

▶ **Lemma 4** (⋆). *Let T be a modular decomposition of a graph G with an induced subgraph H. Then $\text{to}(T|_H, D)$ is the set of orientations of H extendable to transitive orientations of G.*

Consider the canonical modular decompositions $T^1, \ldots, T^r$ of the input graphs $G_1, \ldots, G_r$, and let $(T^1|_H, D_1), \ldots, (T^r|_H, D_r)$ be the corresponding restricted modular decompositions. Then we are interested in the intersection $\text{to}(G_1, \ldots, G_r) = \bigcap_{i=1}^{r} \text{to}(T^i|_H, D_i)$ since it contains all transitive orientations of H that can be extended to all input graphs. However, the trees $T^1|_H, \ldots, T^r|_H$ have different shapes, which makes it difficult to compute a representation of $\text{to}(G_1, \ldots, G_r)$ directly. Instead, we describe $\text{to}(T^1|_H, D_1), \ldots, \text{to}(T^r|_H, D_r)$ (whose intersection is simple to compute) with constraints on the canonical modular decomposition B of H.

Let (T, D) be a restricted modular decomposition of H and let B be the canonical modular decomposition of H. We collect constraints on the orientations of individual nodes μ of B that are imposed by $\text{to}(T, D)$. Afterwards we show that the established constraints are sufficient to describe $\text{to}(T, D)$. If μ is empty, then $H[\mu]$ is empty and has a unique transitive orientation, which requires no constraints. The other types of μ are discussed in the following two sections.

4.1 Constraints for Prime Nodes

In this section we observe that prime nodes in B correspond to prime nodes in (T, D) and that the dependencies between their orientations can be described with 2-SAT formulas. Recall that the transitive orientation of a prime comparability graph is unique up to reversal. We consider each prime node $\mu \in B$ equipped with a transitive orientation D_μ, also called a *default orientation*. All default orientations for B can be computed in linear time [30].

▶ **Lemma 5.** *Let $\mu \in B$ be prime. Then $\mu' = \text{lca}_T(L(\mu))$ is prime, every μ-set is a μ'-set, and for any edge uv with $\text{rep}_B(uv) \in H[\mu]$ we have $\text{rep}_T(uv) \in H[\mu']$.*

Proof. We first show that every μ-set is also a μ'-set. Assume there is a μ-set U that is not a μ'-set. Since $U \subseteq L(\mu) \subseteq L(\mu')$, there exist two vertices $u \neq v \in U$ with $\lambda = \text{rep}_{\mu'}(u) = \text{rep}_{\mu'}(v)$. From $L(\lambda)$ being a module of $H[L(\mu')]$ and $L(\mu) \subseteq L(\mu')$ it follows that $X = L(\lambda) \cap L(\mu)$ is a module of $H[L(\mu)]$. By the definition of μ', we have $X \subsetneq L(\mu)$. Since μ is prime, there is a child ν of μ with $u, v \in X \subseteq L(\nu)$ contradicting U being a μ-set.

As a direct consequence of μ-sets being μ'-sets, we also have for any edge uv with $\text{rep}_B(uv) \in H[\mu]$ that $\text{rep}_T(uv) \in H[\mu']$ since $\{u, v\}$ is a μ-set. Now let U be a maximal μ-set. Then $H[U]$ is isomorphic to $H[\mu]$ and thus prime. Since U is also a μ'-set and the subgraph of $H[\mu']$ induced by the vertices representing U is isomorphic to $H[U]$, μ' is neither empty, nor complete and thus prime. ◀

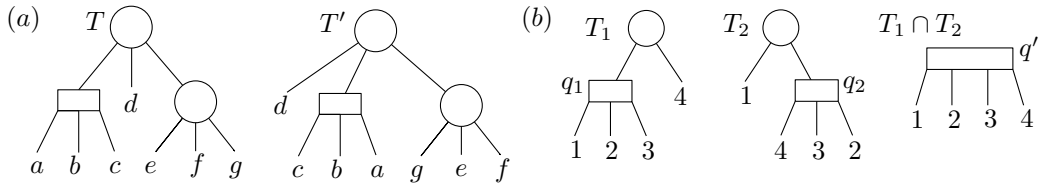

Figure 4 (a) Two equivalent PQ-trees T, T' with $fr(T) = abcdefg$ and $fr(T') = dcbagef$. (b) The Q-node q' in $T_1 \cap T_2$ contains q_1 forwards and q_2 backwards.

For a modular decomposition T' of H with a node λ and $O \in \text{to}(T')$ let $O_{\downarrow\lambda}$ denote the orientation of the quotient graph $H[\lambda]$ we get from O. Note that for a transitive orientation D_λ of $H[\lambda]$ and a λ-set U each orientation $O \in \text{to}(T')$ with $O_{\downarrow\lambda} = D_\lambda$ gives us the same orientation on $H[U]$. We say that D_λ *induces* this orientation on $H[U]$.

Let $\mu \in B$ be prime, let $\mu' = \text{lca}_T(L(\mu))$ and let U be a maximal μ-set (and by Lemma 5 a μ'-set). We set $D^\delta_{\mu'} = D_{\mu'}$ if D_μ, $D_{\mu'}$ induce the same transitive orientation on the prime graph $H[U]$ and we set $D^\delta_{\mu'} = D^{-1}_{\mu'}$ if the induced orientations are the reversal of each other. Note that $D^\delta_{\mu'}$ does not depend on the choice of U. From the definition of $D^\delta_{\mu'}$ and the observation that $O_{\downarrow\mu}, O_{\downarrow\mu'}$ are both determined by O restricted to $H[U]$ we directly get the following lemma.

▶ **Lemma 6.** *For $O \in \text{to}(T, D)$ we have $O_{\downarrow\mu} = D_\mu \Leftrightarrow O_{\downarrow\mu'} = D^\delta_{\mu'}$*

We express the choice of a transitive orientation for a prime node μ by a Boolean variable x_μ that is `true` for the default orientation and `false` for the reversed orientation.

According to Lemma 6 we set ψ_μ to be $x_\mu \leftrightarrow x_{\mu'}$ if $D^\delta_{\mu'} = D_{\mu'}$ and $x_\mu \not\leftrightarrow x_{\mu'}$ if $D^\delta_{\mu'} = D^{-1}_{\mu'}$. Note that for a prime node $\mu' \in T$ there may exist more than one prime node μ in B such that $\mu' = \text{lca}_T(L(\mu))$, and we may hence have multiple prime nodes that are synchronized by these constraints. We describe these dependencies with the formula $\psi = \bigwedge_{\mu \in P(B)} \psi_\mu$. With the above meaning of variables, any choice of orientations for the prime nodes of B that can be induced by T necessarily satisfies ψ. With Lemma 1 we can compute ψ efficiently.

▶ **Lemma 7 (⋆).** *We can compute ψ in $O(|V(H)| + |E(H)|)$ time.*

4.2 Constraints for Complete Nodes

Next we consider the case where μ is complete. The edges represented in $H[\mu]$ may be represented by edges in more than one quotient graph in T, each of which can be complete or prime. Depending on the type of the involved quotient graphs in T we get new constraints for the orientation of $H[\mu]$.

Note that choosing a transitive orientation of $H[\mu]$ is equivalent to choosing a linear order of the vertices of $H[\mu]$. As we will see, each node ν of T that represents an edge of $H[\mu]$ imposes a consecutivity constraint on a subset of the vertices of $H[\mu]$. Therefore, the possible orders can be represented by a *PQ-tree* that allows us to represent all permissible permutations of the elements of a set U in which certain subsets $S \subseteq U$ appear consecutively.

PQ-trees were first introduced by Booth and Lueker [7, 8]. A PQ-tree T over a finite set U is a rooted tree whose leaves are the elements of U and whose internal nodes are labeled as P- or Q-nodes. A P-node is depicted as a circle, a Q-node as a rectangle; see Figure 4. Two PQ-trees T and T' are *equivalent* , if T can be transformed into T' by arbitrarily permuting the children of arbitrarily many P-nodes and reversing the order of arbitrarily many Q-nodes; see Figure 4a. A transformation that transforms T into an equivalent tree T' is called an

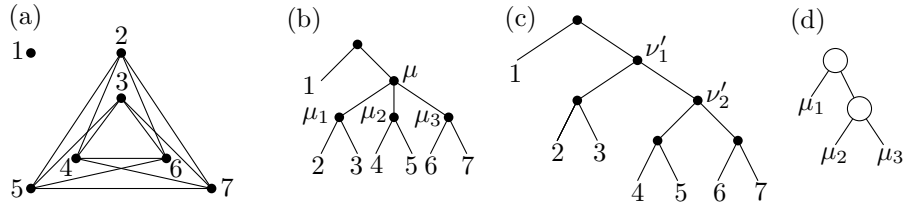

Figure 5 (a) A graph H. (b) The canonical modular decomposition B of H. (c) A restricted modular decompositions T of H. (d) The PQ-tree S_μ. The active nodes of T with regard to the μ-set $\{2, 4, 6\}$ are ν'_1, ν'_2, 2, 4 and 6.

equivalence transformation. The *frontier* $\mathrm{fr}(T)$ of a PQ-tree T is the order of its leaves from left to right. The tree T *represents* the frontiers of all equivalent *PQ*-trees. The PQ-tree that does not have any nodes is called the *null tree*.

Let T_1, T_2 be two PQ-trees over a set U. Their *intersection* $T = T_1 \cap T_2$ is a PQ-tree that represents exactly the linear orders of U represented by both T_1 and T_2. It can be computed in $O(|U|)$ time [7]. For every Q-node q in T_1 node $q' = \mathrm{lca}_T(L(q))$ is also a Q-node. We say that q' *contains* q *forwards*, if $T_1[q]$ can be transformed by an equivalence transformation that does not reverse q into a PQ-tree T' such that $\mathrm{fr}(T[q])$ contains $\mathrm{fr}(T[q'])$; see Figure 4b. Else q' *contains* q *backwards*. Similarly every Q-node in T_2 is contained in exactly one Q-node in T (either forwards or backwards). Haeupler et al. [20] showed that one can modify Booth's algorithm such that given two PQ-trees T_1, T_2 it not only outputs $T_1 \cap T_2$ but also for every Q-node in T_1, T_2 which Q-node in T contains it and in which direction.

▶ **Lemma 8** (⋆). *Let $T_1, \ldots, T_k$ be PQ-trees over a set U. Then we can compute their intersection $T = \bigcap_{i=1}^{k} T_i$ and determine for every Q-node q in $T_1, \ldots, T_k$ the Q-node in T that contains q and in which direction in $O(k \cdot |U|)$ time.*

Let $\mu' = \mathrm{lca}_T(L(\mu))$ for the rest of this section and let $U \subseteq V$ be a maximal μ-set. We call a node of T *active* if it is either a leaf in U or if at least two of its subtrees contain leaves in U. Denote by A the set of active nodes in T and observe that A can be turned into a tree S_μ by connecting each node $\nu' \in A \setminus \{\mu'\}$ to its lowest active ancestor; see Figure 5. Let now $\vec{B}[H]$ be an orientation of the quotient graphs of B induced by an orientation $\vec{T}[G] \in \mathrm{to}(T, D)$ and consider a node $\nu' \neq \mu'$ of S_μ. Let $X = U \cap L(\nu')$ and let $Y = U \setminus L(\nu')$. Since U is a μ-set of a complete node, any pair of vertices in $X \times Y$ is adjacent. Moreover, for each $y \in Y$, the edges from y to X are all oriented towards X, or they are all oriented towards y, since every node of T that determines the orientation of such an edge contains all vertices of X in a single child. This implies that in the order of the μ-set given by the order of $H[\mu]$, the set $L(\nu')$ is consecutive. Moreover, if ν' is prime, its default orientation $D_{\nu'}$ induces a total order on the active children of ν' that is fixed up to reversal. Hence we turn S_μ into a PQ-tree by first turning all complete nodes into P-nodes and all prime nodes into Q-nodes with the children ordered according to the linear order determined by the default orientation which we call the *initial* order of the Q-node. Finally, we replace each leaf $v \in U$ by the corresponding vertex $\mathrm{rep}_\mu(v)$ of $H[\mu]$; see Figure 5. As argued above, the linear order of $H[\mu]$ is necessarily represented by S_μ.

We show that tree S_μ is independent from the choice of the maximal μ-set U. We use that any node of T has a laminar relation to the children of μ.

▶ **Lemma 9** (⋆). *For any child μ_1 of μ and any node κ' of $V(T)$ we have $L(\mu_1) \subseteq L(\kappa') \cap L(\mu)$ or $L(\kappa') \cap L(\mu) \subseteq L(\mu_1)$.*

▶ **Lemma 10** (⋆). *Let S^1_μ, S^2_μ be the PQ-trees for two maximal μ-sets U_1, U_2. Then $S^1_\mu = S^2_\mu$.*

By construction, each Q-node q of S_μ stems from a prime node ν' in T, and the orientation of $T[\nu']$ determines the orientation of q, namely q is reversed if and only if $T[\nu']$ is oriented as $D^{-1}_{\nu'}$. Since a single prime node ν' of T may give rise to Q-nodes in several PQ-trees S_μ, we need to ensure that the orientations of these Q-nodes are either all consistent with the default orientation of $T[\nu']$ or they are all consistent with its reversal. To model this, we introduce a Boolean variable x_q for each Q-node q in one of the PQ-trees with the interpretation that $x_q = \texttt{true}$ if and only if q has its initial order. We require x_q to be equal to the variable that orients the prime node corresponding to q. More precisely, for every prime node ν' in $T[\mu']$ that gives rise to q we add the constraint $(x_{\nu'} \leftrightarrow x_q)$ to χ_μ, where the variable $x_{\nu'}$ is the variable that encodes the orientation of the prime node ν'. We construct a Boolean formula by setting $\chi = \bigwedge_{\mu \in K(B)} \chi_\mu$.

▶ **Lemma 11.** *We can compute all PQ-trees S_μ and the formula χ in $O(n+m)$ time.*

Proof. As a preprocessing we run a DFS on T starting at the root and store for every node ν its discovery-time $\nu.d$, i.e., the timestamp when ν is first discovered, and its finish-time $\nu.f$, i.e., the timestamp after all its neighbors have been examined. We also employ the preprocessing from Lemma 1. We construct all PQ-trees and χ with the following steps.

1. Take a maximal μ-set U_μ for every $\mu \in K(B)$.
2. For every $\mu \in K(B)$ compute the set of active nodes and for every active node compute its parent in S_μ.
3. For every $\mu \in K(B)$ determine for each inner node of S_μ whether it is a P- or a Q-node. If it is a Q-node, determine the linear order of its children, and construct the formula χ_μ.

Step 1 can be done in $O(n)$ time by Lemma 1.

For Step 2, note that each active node is a least common ancestor of two leaves in U_μ. While it is easy to get all active nodes as least common ancestors, getting the edges of S_μ requires more work. Observe that the DFS on T visits the nodes of S_μ in the same order as a DFS on S_μ. Consider S_μ embedded such that the children of each node are ordered from left to right by their discovery-times. This also orders the leaves from left to right by their discovery-times. Let λ be an inner node of S_μ. Let λ_1, λ_2 be two neighboring children of λ with λ_1 to the left of λ_2. Then λ is the least common ancestor of the rightmost leaf in $L(\lambda_1)$ and the leftmost leaf in $L(\lambda_2)$. Hence, each node of S_μ is a least common ancestor for a consecutive pair of leaves.

We add for every node u in a set U_μ a tuple $(\mu, u.d)$ to an initially empty list L. We then sort the tuples in L in linear time using radix sort [13]. In the sorted list, for every $\mu \in K(B)$ all tuples $(\mu, u.d)$ are consecutive and the consecutive sublist is sorted by discovery time.

For $\mu \in K(B)$ let L_μ be a list containing the vertices in U_μ ordered by their discovery time which we get directly from the consecutive sublist of L containing the tuples corresponding to μ. For every pair $u, v \in U_\mu$ adjacent in L_μ we compute $\lambda = \mathrm{lca}_T(u, v)$ using the lowest-common-ancestor data structure for static trees by Harel and Tarjan [21] and insert λ into L_μ between u and v. For a vertex $u \in U_\mu$ its parent in S_μ is the neighbor in L_μ that has a lower position in T. Note that u is a descendent in T of all its neighbors in L_μ. Hence if u has two neighbors in L_μ one of them is a descendent of the other. Thus the parent of u in S_μ is the neighbor with the higher discovery time. Now we remove all vertices in U_μ and possible duplicates of the remaining nodes from L_μ. Note that still every λ is a descendent in T of its neighbors in L_μ. Hence we iteratively choose a node λ in L_μ whose discovery time is higher than the discovery time of its neighbors, compute its parent in S_μ by comparing the discovery times of its neighbors with each other and remove λ from L_μ.

In Step 3, we turn each active node that stems from a complete node into a P-node and each active node that stems from a prime node into a Q-node. For a Q-node q that stems from a prime node ν, we determine the linear order of its children as follows. Take the set X of vertices of $H[\nu]$ that correspond to children of μ in S_μ, determine the orientation of the complete graph on X induced by D_ν and sort it topologically. In total this take $O(n+m)$ time for all active nodes in all PQ-trees. Using the information computed up to this point, it is straightforward to output the formula χ. ◀

Finally, we combine the constraints from the complete nodes with the constraints from the prime nodes by setting $\varphi_T = \psi \wedge \chi$. The formula φ_T allows us to describe a restricted set of transitive orientations of G. We define $S_T = \{S_\mu \mid \mu \in K(B)\}$. The canonical modular decomposition (B, S_T, φ_T) of H where every complete node is labeled with the corresponding PQ-tree together with φ_T we call a *constrained* modular decomposition.

We say that a transitive orientation O of H *induces a variable assignment satisfying* φ_T if it induces an assignments of the variables corresponding to prime nodes in B and Q-nodes such that φ_T is satisfied for an appropriate assignment for the variables corresponding to prime nodes in T. Let $\mathrm{to}(B, S_T, \varphi_T)$ denote the set containing all transitive orientations $O \in \mathrm{to}(B)$ where for every complete node $\mu \in B$ the order $O_{\downarrow\mu}$ corresponds to a total order represented by S_μ and that induces a variable assignment that satisfies φ_T.

4.3 Correctness

We now show that $\mathrm{to}(B, S_T, \varphi_T) = \mathrm{to}(T, D)$. To this end we use that Lemma 5 allows to find for an edge uv that is represented in a prime node μ of B the prime node $\mu' = \mathrm{lca}_T(L(\mu))$ of T where it is represented. This allows us to establish the identity of certain nodes. The following lemma does something similar for complete nodes.

▶ **Lemma 12.** *Let uv, wx be edges of H represented in complete nodes μ, ν of B and by the same edge in a complete node of T. Then $\mu = \nu$.*

Proof. Let ω' be the node in T with $\mathrm{rep}_T(uv) \in H[\omega']$. Assume $\mu \neq \nu$. Then one of them is the ancestor of the other or they are both distinct from $\lambda = \mathrm{lca}_B(\mu, \nu)$. First consider the case that μ is an ancestor of ν. Then μ has a child μ_1 such that $L(\nu) \subseteq L(\mu_1)$. Note that $\mathrm{rep}_\mu(u) \neq \mathrm{rep}_\mu(v)$, hence we have $\mathrm{rep}_\mu(u) \neq \mu_1$ or $\mathrm{rep}_\mu(v) \neq \mu_1$. Without loss of generality assume $\mathrm{rep}_\mu(u) \neq \mu_1$. Since $u \in L(\omega') \cap L(\mu) \setminus L(\mu_1)$ we have $L(\mu_1) \subseteq L(\omega')$ by Lemma 9 and analogously it follows that $L(\mathrm{rep}_\mu(u)) \subseteq L(\omega')$. Since $\mathrm{rep}_T(wx) = \mathrm{rep}_T(uv) \in H[\omega']$ it is $\omega' = lca_T(L(\mu_1)$.

Assume that μ_1 is prime. Then by Lemma 5, ω' is prime which is a contradiction to the assumption that ω' is complete. Hence μ_1 must be complete but as B is the modular decomposition of H, no two adjacent nodes in B are complete. Thus μ is not an ancestor of ν and analogously we get that ν is not an ancestor of μ.

It remains to consider the case that $\nu \neq \lambda \neq \mu$. Let λ_1 be the child of λ such that $L(\mu) \subseteq L(\lambda_1)$ and let λ_2 be the child of λ such that $L(\nu) \subseteq L(\lambda_2)$. Again λ has to be complete since otherwise by Lemma 5 ω' would be prime which is a contradiction. By Lemma 9 we have that $L(\lambda_1) \cup L(\lambda_2) \subseteq L(\omega')$.

Assume that λ_1 is prime. Then by Lemma 5, ω' is prime which leads to a contradiction. Hence λ_1 must be complete but again as B is the modular decomposition of H, no two adjacent nodes in B are complete. ◀

▶ **Theorem 13.** *Let B be the canonical modular decomposition for a graph H and let T be a restricted modular decomposition for H. Then* $\mathrm{to}(B, S_T, \varphi_T) = \mathrm{to}(T, D)$ *and* $\mathrm{to}(B, S_T, \varphi_T)$ *can be computed in time that is linear in the size of H.*

Proof. Let $O_H \in \mathrm{to}(T, D)$ and let $\vec{B}[H]$ be the orientation of the quotient graphs of B inducing O_H. Then we have already seen that it is necessary that every complete node μ in B is oriented according to a total order represented by S_μ and that O_H induces a variable assignment that satisfies φ_T. Hence $O_H \in \mathrm{to}(B, S_T, \varphi_T)$.

Conversely, let $O_H \in \mathrm{to}(B, S_T, \varphi_T)$ and assume $O_H \notin \mathrm{to}(T, D)$. Then O_H is either not represented by T or does not induce D. I.e., O_H contains two directed edges uv, wx with $\mathrm{rep}_T(uv)$ and $\mathrm{rep}_T(wx)$ in the same quotient graph $H[\omega']$, such that $\mathrm{rep}_T(uv) = \mathrm{rep}_T(xw)$, or ω' is prime and $\mathrm{rep}_T(uv) \in D_{\omega'}$ but $\mathrm{rep}_T(wx) \in D_{\omega'}^{-1}$. Note that if ω' is prime, then the first case implies the second one. Let $\mu = \mathrm{lca}_B(u, v)$ and $\nu = \mathrm{lca}_B(w, x)$. We distinguish cases based on the types of μ and ν. Let $\mu' = \mathrm{lca}_T(L(\mu))$, $\nu' = \mathrm{lca}_T(L(\nu))$ and let $\vec{B}[H]$ be the orientation of the quotient graphs of B that induces O_H. Without loss of generality, assume $\mathrm{rep}_B(uv) \in D_\mu$ and $\mathrm{rep}_B(wx) \in D_\nu$.

Case 1: μ and ν are both prime. By Lemma 5 we have that $\mu' = \omega' = \nu'$ is prime. By construction, φ_T enforces that uv, wx are either represented in $D_{\omega'}$ or in $D_{\omega'}^{-1}$. Hence, this case does not occur.

Case 2: μ is prime and ν is complete. By Lemma 5 we have that $\mu' = \omega'$ is prime. Let U be a ν-set containing w and x. Since $\mathrm{rep}_T(wx) \in H[\omega']$, node ω' is active with respect to U. Hence the PQ-tree S_ν contains a Q-node q that stems from ω'.

By construction φ_T contains the constraints $(x_{\omega'} \leftrightarrow x_q)$ and $(x_{\omega'} \leftrightarrow x_\mu)$ but $\vec{B}[H]$ induces $x_\mu = \texttt{true}$, $x_q = \texttt{false}$. Hence the variable assignment induced by $\vec{B}[H]$ does not satisfy φ_T and thus $O_H \notin \mathrm{to}(B, S_T, \varphi_T)$.

Case 3: μ is complete and ν is prime. Similar to Case 2.

Case 4: μ, ν are both complete. Here we further distinguish two subcases depending on the type of ω'. First assume that ω' is prime. Let V_1' be a μ-set containing u, v and let V_2' be a ν-set containing w, x. Since $\mathrm{rep}_T(uv)$ and $\mathrm{rep}_T(wx)$ are edges in $H[\omega']$, node ω' is active with respect to both V_1', V_2'. Hence S_μ, S_ν contain Q-nodes q_1, q_2 stemming from ω'. By construction φ_T contains the constraints $(x_{\omega'} \leftrightarrow x_{q_1})$ and $(x_{\omega'} \leftrightarrow x_{q_2})$, but $\vec{B}[H]$ induces $x_{q_1} = \texttt{true}$, $x_{q_2} = \texttt{false}$. Hence the variable assignment induced by O_H does not satisfy φ_T and thus $O_H \notin \mathrm{to}(B, S_T, \varphi_T)$.

It remains to consider the case that ω' is complete. By Lemma 12 we have $\mu = \nu$. Since ω' is not prime, we must have $\mathrm{rep}_T(uv) = \mathrm{rep}_T(xw)$ by assumption. Let U be a μ-set. Since $\mathrm{rep}_T(uv) = \mathrm{rep}_T(xw) \in H[\omega']$, node ω' is active with respect to U and $\mathrm{rep}_{\omega'}(u) = \mathrm{rep}_{\omega'}(x)$, $\mathrm{rep}_{\omega'}(v) = \mathrm{rep}_{\omega'}(w)$. Hence S_μ contains a P-node that stems from ω' and demands a total order of the children of μ where $\mathrm{rep}_\mu(u)$, $\mathrm{rep}_\mu(x)$ are either both smaller than both $\mathrm{rep}_\mu(v)$, $\mathrm{rep}_\mu(w)$, or $\mathrm{rep}_\mu(u)$, $\mathrm{rep}_\mu(x)$ are both greater than both $\mathrm{rep}_\mu(v)$, $\mathrm{rep}_\mu(w)$. Since $\vec{B}[H]$ induces $\mathrm{rep}_\mu(u) < \mathrm{rep}_\mu(v)$ but $\mathrm{rep}_\mu(x) > \mathrm{rep}_\mu(w)$, $H[\mu]$ is not oriented according to a total order represented by S_μ and thus $O_H \notin \mathrm{to}(B, S_T, \varphi_T)$.

By Lemmas 7 and 11 the formula $\varphi_T = \psi \wedge \chi$ and S_T can be computed in linear time. ◀

Let T be a modular decomposition of a graph G with an induced subgraph H. Let B be the canonical modular decomposition of H and let $T|_H$ be the restricted modular decomposition for H we get from T. From Lemma 4 and Theorem 13 we directly get the following corollary.

▶ **Corollary 14.** *The set* $\mathrm{to}(B, S_{T|_H}, \varphi_{T|_H})$ *contains exactly those transitive orientations of* H *that can be extended to a transitive orientation of* G*.*

Let $(B, S^1, \varphi_1), \ldots, (B, S^r, \varphi_r)$ be constrained modular decompositions for H. Let $S_\mu = \bigcap_{i=1}^r S_\mu^i$ and $S = \{S_\mu \mid \mu \in K(B)\}$. The intersection (B, S, φ) of $(B, S^1, \varphi_1), \ldots, (B, S^r, \varphi_r)$ is the constrained modular decomposition of H where every complete node μ is labeled with the PQ-tree S_μ equipped with the 2-Sat-formula $\varphi = (\bigwedge_{i=1}^r \varphi_i) \wedge \varphi'$ where φ' synchronizes the Q-nodes in the S_μ's with the Q-nodes in the S_μ^i's as follows. Recall that for every Q-node q in a tree S_μ^i there exists a unique Q-node q' in S_μ that contains q; either forward or backward. For every $i \in \{1, \ldots, r\}$, every $\mu \in K(B)$ and every Q-node q in S_μ^i we determine the Q-node q' in S_μ that contains q and add the clause $(x_q \leftrightarrow x_{q'})$ if q has its forward orientation in q' and $(x_q \not\leftrightarrow x_{q'})$ otherwise.

▶ **Lemma 15 (⋆).** *It is* $\mathrm{to}(B, S, \varphi) = \bigcap_{i=1}^r \mathrm{to}(B, S^i, \varphi_i)$ *and we can compute* (B, S, φ) *in linear time from* $\{(B, S^i, \varphi_i) \mid 1 \le i \le r\}$*.*

Now consider the case where $(B, S^1, \varphi_1), \ldots, (B, S^r, \varphi_r)$ are the constrained modular decompositions we get from the restricted modular decompositions $T_1|_H, \ldots, T_r|_H$. By Lemma 12 and Theorem 13 we have $\mathrm{to}(B, S, \varphi) = \bigcap_{i=1}^r \mathrm{to}(B, S_i, \varphi_i) = \bigcap_{i=1}^r \mathrm{to}(T_i, D)$ and by Lemma 4 $G_1, \ldots, G_r$ are simultaneous comparability graphs if and only if φ is satisfiable and S does not contain the null tree.

▶ **Theorem 16.** *SimOrient can be solved in linear time.*

Proof. Let $G_1 = (V_1, E_1), \ldots, G_r = (V_r, E_r)$ be r-sunflower graphs with $n_i = |V_i|$ and $m_i = |E_i|$ for all $1 \le i \le r$ and let $n = \sum_{i=1}^r n_i$, $m = \sum_{i=1}^r m_i$. We solve SimOrient as follows.

1. Compute the canonical modular decomposition T_i for every G_i and the canonical modular decomposition B of H in $O(n + m)$ time by McConnell and Spinrad [30].
2. Compute $T_i|_H$ for every i in $O(n)$ time in total.
3. Compute (B, S^i, φ_i) for every i, in $O(n + m)$ time in total by Theorem 13.
4. Compute (B, S, φ) in linear time by Lemma 12.
5. Check whether S contains the null tree and whether φ is satisfiable in linear time.

We execute Step 5 as follows. For $i \in \{1, \ldots, r\}$, φ_i contains one variable and one constraint per prime node in B, one variable per prime node in $T_i|_H$ and one variable and one constraint per Q-node in S_μ^i. Since S_μ^i has $O(n_i)$ nodes, it contains $O(n_i)$ Q-nodes. Hence in total every φ_i contains $O(n_i)$ variables and clauses. Note that φ' contains one clause per Q-node in the PQ-trees in S. Hence φ' contains $O(n)$ clauses and variables and thus the 2-SAT formula φ can be solved in $O(n)$ time by Aspvall et al. [3].

If S does not contain the null tree and φ has a solution, we get simultaneous transitive orientations of $G_1, \ldots, G_r$ in linear time by proceeding as follows. We orient every complete quotient graph of B according to a total order induced by the corresponding PQ-tree where every Q-node is oriented according to the solution of φ. For a prime quotient graph $H[\mu]$ we choose D_μ if in the chosen solution of φ we have $x_\mu = \texttt{true}$ and D_μ^{-1} otherwise. Together, all these orientations of quotient graphs of B induce a transitive orientation on H and by applying the linear-time algorithm from Section 3 to solve OrientExt we can extend it to a transitive orientation of G_i for every $i \in \{1, \ldots, r\}$. ◀

5 Applications to Permutation Graphs

With the results from Section 3 and 4 we can also solve RepExt(Perm) and SimRep* (Perm) efficiently. For RepExt(Perm) we exploit that a given partial representation D' of a permutation graph G is extendible if and only if for every prime node μ in the canonical modular decomposition of G the partial representation of $G[\mu]$ induced by D' is extendible. Since $G[\mu]$ has only a constant number of representations this can be checked in linear time.

▶ **Theorem 17** ($\star$). *RepExt(Perm) can be solved in linear time.*

Jampani and Lubiw showed that sunflower permutation graphs are simultaneous permutation graphs if and only if they are simultaneous comparability graphs and simultaneous co-comparability graphs [23]. We show that it is possible to use the algorithm from Section 4 to solve SimOrient also for co-comparability graphs and thus SimRep*(Perm) for given sunflower permutation graphs $G_1, \ldots, G_r$ with shared subgraph H in linear time. Recall that a graph G and its complement have the same canonical modular decomposition [16]. In linear time we can not explicitly compute the complements of the input graphs and the corresponding quotient graphs. Hence we can not store a default orientation for the prime quotient graphs $\overline{H}[\mu]$. We can, however compute a *default* permutation diagram representing $H[\mu]$ from its default orientation. This suffices to answer all queries concerning the default orientation of $\overline{H}[\mu]$ in linear time.

▶ **Theorem 18** ($\star$). *SimRep*(Perm) can be solved in linear time.*

Switching a vertex v in a circular permutation graph G, i.e. connecting it to all vertices it was not adjacent to in G and removing all edges to its former neighbors, gives us the graph $G_v = (V, E_v)$ with $E_v = (E \setminus \{vx \in E \mid x \in V\}) \cup \{vx \mid x \in V, vx \notin E\}$ [33]. The graph we obtain by switching all neighbors of a vertex v we denote by $G_{N(v)}$. Switching the neighborhood of a vertex v reduces RepExt(CPerm) and SimRep*(CPerm) to RepExt(Perm) and SimRep*(Perm). Since $G_{N(v)}$ may potentially have a quadratic number of edges we need quadratic time to solve SimRep*(CPerm).

▶ **Theorem 19** ($\star$). *SimRep*(CPerm) can be solved in $O(n^2)$ time.*

For RepExt(Perm) however we can choose a vertex v of minimum degree which ensures that the size of $G_{N(v)}$ is linear in the size of G [36]. Hence the problem can be solved in linear time.

▶ **Theorem 20** ($\star$). *RepExt(CPerm) can be solved in linear time.*

6 Conclusion

We showed that the orientation extension problem and the simultaneous orientation problem for sunflower comparability graphs can be solved in linear time using the concept of modular decompositions. Further we were able to use these algorithms to solve the partial representation problem for permutation and circular permutation graphs and the simultaneous representation problem for sunflower permutation graphs also in linear time. For the simultaneous representation problem for circular permutation graphs we gave a quadratic-time algorithm.

It remains an open problem whether the simultaneous representation problem for sunflower circular permutation graphs can be solved in subquadratic time. Furthermore it would be interesting to examine whether the concept of modular decomposition is also applicable to

solve the partial representation and the simultaneous representation problem for further graph classes, e.g. trapezoid graphs. There may also be other related problems that can be solved for comparability, permutation and circular permutation graphs with the concept of modular decompositions.

References

1 Patrizio Angelini, Giuseppe Di Battista, Fabrizio Frati, Vít Jelínek, Jan Kratochvíl, Maurizio Patrignani, and Ignaz Rutter. Testing planarity of partially embedded graphs. *ACM Transactions on Algorithms*, 11(4):32:1–32:42, 2015.

2 Patrizio Angelini, Giordano Da Lozzo, and Daniel Neuwirth. On some $\mathcal{NP}$-complete SEFE problems. In Sudebkumar Prasant Pal and Kunihiko Sadakane, editors, *In Proceedings of the 8th International Workshop on Algorithms and Computation (WALCOM'14)*, volume 8344 of *Lecture Notes in Computer Science*, pages 200–212. Springer, 2014. doi:10.1007/978-3-319-04657-0_20.

3 Bengt Aspvall, Michael F. Plass, and Robert E. Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters*, 8(3):121–123, 1979.

4 Thomas Bläsius, Stephen G. Kobourov, and Ignaz Rutter. Simultaneous embedding of planar graphs. *Computing Research Repository*, abs/1204.5853, 2012. arXiv:1204.5853.

5 Thomas Bläsius and Ignaz Rutter. Simultaneous PQ-ordering with applications to constrained embedding problems. *ACM Transactions on Algorithms*, 12(2):16:1–16:46, 2015. doi:10.1145/2738054.

6 Jan Bok and Nikola Jedličková. A note on simultaneous representation problem for interval and circular-arc graphs. *Computing Research Repository*, 2018. arXiv:1811.04062.

7 Kellogg S. Booth. *PQ-tree algorithms*. PhD thesis, University of California, Berkeley, 1975.

8 Kellogg S. Booth and George S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *Journal of Computer and System Sciences*, 13(3):335–379, 1976.

9 Peter Brass, Eowyn Cenek, Cristian A. Duncan, Alon Efrat, Cesim Erten, Dan P. Ismailescu, Stephen G. Kobourov, Anna Lubiw, and Joseph S.B. Mitchell. On simultaneous planar graph embeddings. *Computational Geometry*, 36(2):117–130, 2007. doi:10.1016/j.comgeo.2006.05.006.

10 Steven Chaplick, Paul Dorbec, Jan Kratochvíl, Mickael Montassier, and Juraj Stacho. Contact representations of planar graphs: Extending a partial representation is hard. In Dieter Kratsch and Ioan Todinca, editors, *40th International Workshop on Graph-theoretic concepts in computer science (WG'14)*, volume 8747 of *Lecture Notes in Computer Science*, pages 139–151. Springer, 2014.

11 Steven Chaplick, Radoslav Fulek, and Pavel Klavík. Extending partial representations of circle graphs. *Journal of Graph Theory*, 91(4):365–394, 2019.

12 Steven Chaplick, Philipp Kindermann, Jonathan Klawitter, Ignaz Rutter, and Alexander Wolff. Extending partial representations of rectangular duals with given contact orientations. In Tiziana Calamoneri and Federico Corò, editors, *Proceedings of the 12th International Conference on Algorithms and Complexity, (CIAC '21)*, volume 12701 of *Lecture Notes in Computer Science*, pages 340–353. Springer, 2021. doi:10.1007/978-3-030-75242-2_24.

13 Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT press, Cambridge, 2009.

14 Eduard Eiben, Robert Ganian, Thekla Hamm, Fabian Klute, and Martin Nöllenburg. Extending partial 1-planar drawings. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *Proceedings of the 47th International Colloquium on Automata, Languages, and Programming (ICALP'20)*, volume 168 of *LIPIcs*, pages 43:1–43:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ICALP.2020.43.

15 Alejandro Estrella-Balderrama, Elisabeth Gassner, Michael Jünger, Merijam Percan, Marcus Schaefer, and Michael Schulz. Simultaneous geometric graph embeddings. In Seok-Hee Hong, Takao Nishizeki, and Wu Quan, editors, *Proceedings of 15th International Symposium on Graph Drawing (GD '07)*, pages 280–290. Springer, 2008. doi:10.1007/978-3-540-77537-9_28.

16 Tibor Gallai. Transitiv orientierbare graphen. *Acta Mathematica Academiae Scientiarum Hungarica*, 18(1-2):25–66, 1967.

17 Elisabeth Gassner, Michael Jünger, Merijam Percan, Marcus Schaefer, and Michael Schulz. Simultaneous graph embeddings with fixed edges. In Fedor V. Fomin, editor, *32nd International Workshop on Graph-Theoretic Concepts in Computer Science (WG'06)*, pages 325–335. Springer, 2006. doi:10.1007/11917496_29.

18 Paul C Gilmore and Alan J Hoffman. A characterization of comparability graphs and of interval graphs. *Canadian Journal of Mathematics*, 16:539–548, 1964.

19 Martin Charles Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Elsevier, London, 2004.

20 Bernhard Haeupler, Krishnam R. Jampani, and Anna Lubiw. Testing simultaneous planarity when the common graph is 2-connected. *Journal of Graph Algorithms and Applications*, 17(3):147–171, 2013.

21 Dov Harel and Robert E. Tarjan. Fast algorithms for finding nearest common ancestors. *SIAM Journal on Computing*, 13(2):338–355, 1984.

22 Krishnam Raju Jampani and Anna Lubiw. Simultaneous interval graphs. In Otfried Cheong, Kyung-Yong Chwa, and Kunsoo Park, editors, *Proceedings of the 21st International Symposium on Algorithms and Computation (ISAAC' 10)*, pages 206–217. Springer, 2010. doi:10.1007/978-3-642-17517-6_20.

23 Krishnam Raju Jampani and Anna Lubiw. The simultaneous representation problem for chordal, comparability and permutation graphs. *Journal of Graph Algorithms and Applications*, 16(2):283–315, 2012.

24 Vít Jelínek, Jan Kratochvíl, and Ignaz Rutter. A Kuratowski-type theorem for planarity of partially embedded graphs. *Computational Geometry*, 46(4):466–492, 2013.

25 Pavel Klavík, Jan Kratochvíl, Tomasz Krawczyk, and Bartosz Walczak. Extending partial representations of function graphs and permutation graphs. In Leah Epstein and Paolo Ferragina, editors, *20th Annual European Symposium on Algorithms (ESA'12)*, Lecture Notes in Computer Science, pages 671–682. Springer, 2012.

26 Pavel Klavík, Jan Kratochvíl, Yota Otachi, Ignaz Rutter, Toshiki Saitoh, Maria Saumell, and Tomáš Vyskočil. Extending partial representations of proper and unit interval graphs. *Algorithmica*, 77(4):1071–1104, 2017.

27 Pavel Klavík, Jan Kratochvíl, Yota Otachi, and Toshiki Saitoh. Extending partial representations of subclasses of chordal graphs. *Theoretical Computer Science*, 576:85–101, 2015.

28 Pavel Klavík, Jan Kratochvíl, Yota Otachi, Toshiki Saitoh, and Tomáš Vyskočil. Extending partial representations of interval graphs. *Algorithmica*, 78(3):945–967, 2017.

29 Tomasz Krawczyk and Bartosz Walczak. Extending partial representations of trapezoid graphs. In Hans L. Bodlaender and Gerhard J. Woeginger, editors, *43rd International Workshop on Graph-Theoretic Concepts in Computer Science (WG'17)*, pages 358–371. Springer, 2017.

30 Ross M McConnell and Jeremy P Spinrad. Modular decomposition and transitive orientation. *Discrete Mathematics*, 201(1-3):189–241, 1999.

31 Miriam Münch, Ignaz Rutter, and Peter Stumpf. Partial and simultaneous transitive orientations via modular decomposition, 2022. doi:10.48550/ARXIV.2209.13175.

32 Maurizio Patrignani. On extending a partial straight-line drawing. *International Journal of Foundations of Computer Science*, 17(5):1061–1070, 2006.

33 Doron Rotem and Jorge Urrutia. Circular permutation graphs. *Networks*, 12(4):429–437, 1982.

34 Ignaz Rutter, Darren Strash, Peter Stumpf, and Michael Vollmer. Simultaneous representation of proper and unit interval graphs. In Michael A. Bender, Ola Svensson, and Grzegorz Herman, editors, *27th Annual European Symposium on Algorithms (ESA'19)*, volume 144, page 80. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019.

35 Marcus Schaefer. Toward a theory of planarity: Hanani-Tutte and planarity variants. In *20th International Symposium on Graph Drawing (GD '12)*, pages 162–173. Springer, 2012. doi:10.1007/978-3-642-36763-2_15.

36 R Sritharan. A linear time algorithm to recognize circular permutation graphs. *Networks: An International Journal*, 27(3):171–174, 1996.