Extending Partial Representations of Circle Graphs*

Steven Chaplick**, Radoslav Fulek**, and Pavel Klavík***

Abstract. *The partial representation extension problem* is a recently introduced generalization of the recognition problem. A *circle graph* is an intersection graph of chords of a circle. We study the partial representation extension problem for circle graphs, where the input consists of a graph G and a partial representation $\mathcal{R}'$ giving some pre-drawn chords that represent an induced subgraph of G. The question is whether one can extend $\mathcal{R}'$ to a representation $\mathcal{R}$ of the entire G, i.e., whether one can draw the remaining chords into a partially pre-drawn representation.

Our main result is a polynomial-time algorithm for partial representation extension of circle graphs. To show this, we describe the structure of all representation a circle graph based on split decomposition. This can be of an independent interest.

1 Introduction

Graph drawings and visualizations are important topics of graph theory and computer science. A frequently studied type of representations are so-called *intersection representations*. An intersection representation of a graph represents its vertices by some objects and encodes its edges by intersections of these objects, i.e., two vertices are adjacent if and only if the corresponding objects intersect. Classes of intersection graphs are obtained by restricting these objects; e.g., *interval graphs* are intersection graphs of intervals of the real line, *string graphs* are intersection graphs of strings in plane, and so on. These representations are well-studied; see e.g. [30].

For a fixed class $\mathcal{C}$ of intersection-defined graphs, a very natural computational problem is *recognition*. It asks whether an input graph G belongs to $\mathcal{C}$. In this paper, we study a recently introduced generalization of this problem called *partial representation extension* [23]. Its input gives with G a part of the representation and the problem asks whether this partial representation can be extended to a representation of the entire G; see Fig. 1 for an illustration. We

* Supported by ESF Eurogiga project GraDR as GAČR GIG/11/E023.

** Department of Applied Mathematics, Faculty of Mathematics and Physics, Charles University, Malostranské náměstí 25, 118 00 Prague, Czech Republic. E-mails: `chaplick@kam.mff.cuni.cz`, `radoslav.fulek@gmail.com`.

*** Computer Science Institute, Faculty of Mathematics and Physics, Charles University, Malostranské náměstí 25, 118 00 Prague, Czech Republic. E-mail: `klavik@iuuk.mff.cuni.cz`.
Supported by Charles University as GAUK 196213.

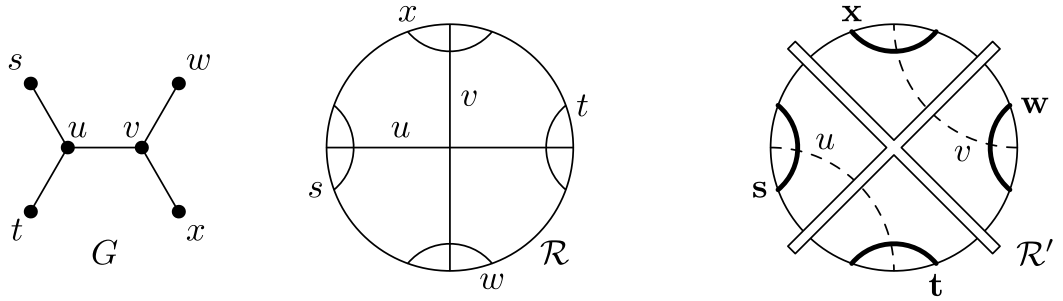

Fig. 1. On the left, a circle graph G with a representation $\mathcal{R}$ is given. A partial representation $\mathcal{R}'$ given on the right with the pre-drawn chords $\mathbf{s}$, $\mathbf{t}$, $\mathbf{w}$, and $\mathbf{x}$ is not extendible. The chords are depicted as arcs to make the figure more readable.

show that this problem can be solved in polynomial time for the class of *circle graphs.*

Circle Graphs. Circle graphs are intersection graphs of chords of a circle. They were first considered by Even and Itai [13] in the early 1970s in study of stack sorting techniques. Other motivations are due to their relations to Gauss words [12] (see Fig. 2) and matroid representations [11,5]. Circle graphs are also important regarding rank-width [27].

Let $\chi(G)$ denote the chromatic number of G, and let $\omega(G)$ denote the clique-number of G. Trivially we have $\omega(G) \le \chi(G)$ and the graphs for which every induced subgraph satisfies equality are the well-known *perfect graphs* [6]. In general, the difference between these two numbers can be arbitrarily high, e.g., there is a triangle-free graph with an arbitrary high chromatic number. Circle graphs are known to be *almost perfect* which means that $\chi(G) \le f(\omega(G))$ for some function f. The best known result for circle graphs [24] states that $f(k)$ is $\Omega(k \log k)$ and $\mathcal{O}(2^k)$.

Some hard computational problems, such as 3-colorability [15], or maximum weighted clique and independent set [16], become tractable on circle graphs. On the other hand, the problems such as vertex colorability [15] or Hamiltonicity [10] remain NP-complete even for circle graphs.

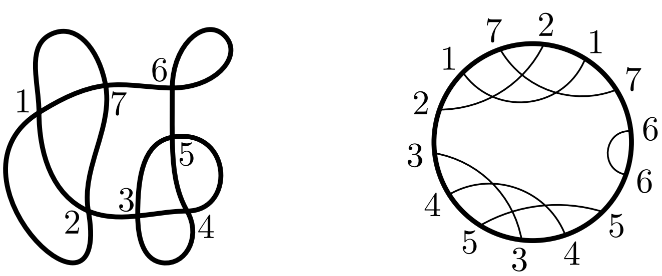

Fig. 2. A self-intersecting closed curve with n intersections numbered $1, \dots, n$ corresponds to a representation of circle graph with the vertices $1, \dots, n$ where the endpoints of the chords are placed according to the order of the intersections along the curve.

The complexity of recognition of circle graphs was a long standing open problem; see [30] for an overview. The first results, e.g., [13], gave existential characterizations which did not give polynomial-time algorithms. The mystery whether circle graphs can be recognized in polynomial time frustrated mathematicians for some years. It was resolved in the mid-1980s and several polynomial-time algorithms were discovered [4,14,25] (in time $\mathcal{O}(n^7)$ and similar). Later, a more efficient algorithm [29] based on *split decomposition* was given, and the current state-of-the-art recognition algorithm [17] runs in a quasi-linear time in the number of vertices and the number of edges of the graph.

The Partial Representation Extension Problem. It is quite surprising that this very natural generalization of the recognition problem was considered only recently. It is currently an active area of research which is inspiring a deeper investigation of many classical graph classes. For instance, a recent result of Angelini et al. [1] states that the problem is decidable in linear time for planar graphs. On the other hand, Fáry's Theorem claims that every planar graph has a straight-line embedding, but extension of such an embedding is NP-hard [28].

In the context of intersection-defined classes, this problem was first considered in [23] for interval graphs. Currently, the best known results are linear-time algorithms for interval graphs [3,22] and proper interval graphs [20], a quadratic-time algorithm for unit interval graphs [20], and polynomial-time algorithms for permutation and function graphs [19]. For chordal graphs (as subtree-in-a-tree graphs) several versions of the problems were considered [21] and all of them are NP-complete.

The Structure of Representations. To solve the recognition problem for G, one just needs to build a single representation. However, to solve the partial representation extension problem, the structure of all representations of G must be well understood. A general approach used in the above papers is the following. We first derive necessary and sufficient constraints from the partial representation $\mathcal{R}'$. Then we efficiently test whether some representation $\mathcal{R}$ satisfies these constraints. If none satisfies them, then $\mathcal{R}'$ is not extendible. And if some $\mathcal{R}$ satisfies them, then it extends $\mathcal{R}'$.

It is well-known that the split decomposition [8, Theorem 3] captures the structure of all representations of circle graphs. The standard recognition algorithms produce a special type of representations using split decomposition as follows. We find a *split* in G, construct two smaller graphs, build their representation recursively, and then join these two representations to produce $\mathcal{R}$. In Section 3, we give a simple recursive descriptions of all possible representations based on splits. Our result can be interpreted as "describing a structure like PQ-trees for circle graphs." It is possible that the proof techniques from other papers on circle graphs such as [7,17] would give a similar description. However, these techniques are more involved than our approach which turned out to be quite elementary and simple.

Restricted Representations. The partial representation extension problem belongs to a larger group of problems dealing with *restricted representations of graphs.* These problems ask whether there is some representation of an input graph G satisfying some additional constraints. We describe two examples of these problems.

An input of the *simultaneous representations problem*, shortly SIM, consists of graphs $G_1, \dots, G_k$ with some vertices common for all the graphs. The problem asks whether there exist representations $\mathcal{R}_1, \dots, \mathcal{R}_k$ representing the common vertices the same. This problem is polynomially solvable for permutation and comparability graphs [18]. They additionally show that for chordal graphs it is NP-complete when k is part of the input and polynomially solvable for $k = 2$. For interval graphs, a linear-time algorithm is known for $k = 2$ [3] and the complexity is open in general. For some classes, these problems are closely related to the partial representation extension problems. For example, there is an FPT algorithm for interval graphs with the number of common vertices as the parameter [23], and partial representations of interval graphs can be extended in linear time by reducing it to corresponding simultaneous representations problems [3].

The *bounded representation problem* [20] prescribes bounds for each vertex of the input graph and asks whether there is some representation satisfying these bounds. For circle graphs, the input specifies for each chord v a pair of arcs (A_v, A'_v) of the circle, and a solution is required to have one endpoint of v in A_v and the other one in A'_v. This problem is clearly a generalization of partial representation extension since one can describe a partial representation using singleton arcs. It is known to be polynomially solvable for interval and proper interval graphs [2], and surprisingly it is NP-complete for unit interval graphs [20]. The complexity for other classes of graphs is not known.

Our Results. We study the following problem (see Section 2 for definitions):

Problem:	Partial Representation Extension – REPEXT(CIRCLE)
Input:	A circle graph G and a partial representation $\mathcal{R}'$.
Output:	Is there a representation $\mathcal{R}$ of G extending $\mathcal{R}'$?

In Section 3, we describe a simple structure of all representations. This is used in Section 4 to obtain our main algorithmic result:

Theorem 1. *The problem* REPEXT(CIRCLE) *can be solved in polynomial time.*

To spice up our results, we show in Section 5 the following for the simultaneous representations problem of circle graphs:

Proposition 2. *For k part of the input, the problem* $\textsc{Sim}(\mathsf{CIRCLE})$ *is* NP-*complete.*

Corollary 3. *The problem* $\textsc{Sim}(\mathsf{CIRCLE})$ *is* FPT *in the size of the common subgraph.*

2 Definitions and Preliminaries

Circle Representations. A *circle representation* $\mathcal{R}$ is a collection $\{C_u \mid u \in V(G)\}$ of chords of a circle such that C_u intersects C_v if and only if $uv \in E(G)$. A graph is a *circle graph* if it has a circle representation, and we denote the class of circle graphs by CIRCLE.

Notice that a representation of a circle graph is completely determined by its circular order of the endpoints of the chords in the representation, and two chords C_u and C_v cross if and only if their endpoints alternate in this order. For convenience we label both endpoints of the chord representing a vertex by the same label as the vertex.

A *partial representation* $\mathcal{R}'$ is a representation of an induced subgraph G'. The vertices of G' are *pre-drawn* vertices and the chords of $\mathcal{R}'$ are *pre-drawn chords*. A representation $\mathcal{R}$ *extends* $\mathcal{R}'$ if $C_u = C'_u$ for every $u \in V(G')$.

Interval Overlap Graphs. Suppose that we pick an arbitrary point of the circle that is not an endpoint of a chord (depicted by white cross in Fig. 3). We cut the circle at this point and straighten it into a segment. Moreover, we turn each chord into an arc connecting its two endpoints. Notice that two chords C_u and C_v cross if their endpoints appear in the order $uvuv$ or $vuvu$ from left to right. See Fig. 3 on the right. Alternatively, circle graphs are called *interval overlap graphs*. Their vertices can be represented by intervals and two vertices

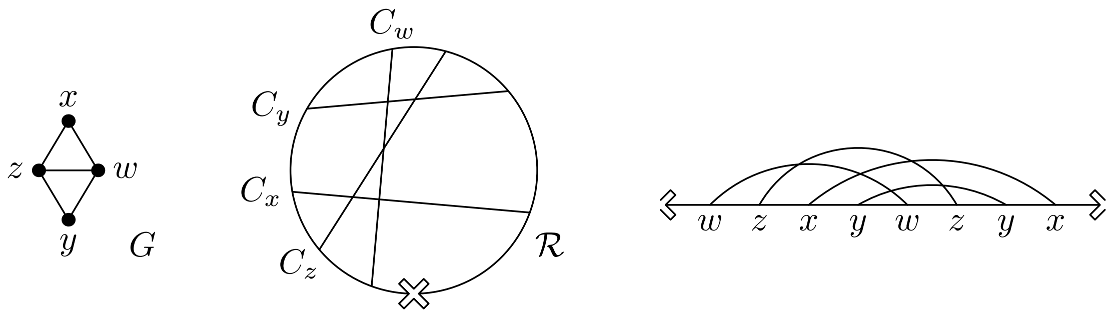

Fig. 3. An example of a circle graph with a circle graph representation on the left; an interval overlap representation of the same graph on the right.

are adjacent if and only if their intervals overlap which means they intersect and one is not a subset of the other.

Word representations. A sequence τ over an alphabet of symbols Σ is a *word*. A *circular word* represents a set of words which are cyclical shifts of one another. In the sequel, we represent a circular word by a word from its corresponding set of words. We denote words and circular words by small Greek letters.

For a word τ and a symbol u we write $u \in \tau$, if u appears at least once in τ. Thus, τ is also used to denote the set of symbols occurring in τ. A word τ is a *subword* of σ, if τ appears consecutively in σ. A word τ is a *subsequence* of σ, if the word τ can be obtained from σ by deleting some symbols. We say that u *alternates* with v in τ, if $uvuv$ or $vuvu$ is a subsequence of τ. The corresponding definitions also apply to circular words. If σ and τ are two words, we denote their concatenation by $\sigma\tau$.

The above interpretation of circle graphs as interval overlap graphs allows to associate each representation $\mathcal{R}$ of G with the unique circular word τ over V. The word τ is obtained by the circular order of the endpoints of the chords in $\mathcal{R}$ as they appear along the circle when traversed clockwise. The occurrences of u and v alternate in τ if and only if $uv \in E(G)$. For example $\mathcal{R}$ in Fig. 1 corresponds to the circular word $\tau = susxvxtutwvw$.

Let G be a circle graph, and let $\mathcal{R}$ be its representation with the corresponding circular word τ. If G' is an induced subgraph of G, then the subsequence of τ consisting of the vertices in G' is a circular word σ. This σ corresponds to a representation $\mathcal{R}'$ of G' which is extended by $\mathcal{R}$.

3 Structure of Representations of Splits

Let G be a connected graph. A *split* of G is a partition of the vertices of G into four parts A, B, $\mathfrak{s}(A)$ and $\mathfrak{s}(B)$, such that:

- For every $a \in A$ and $b \in B$, we have $ab \in E(G)$.
- There is no edge between $\mathfrak{s}(A)$ and $B \cup \mathfrak{s}(B)$, and between $\mathfrak{s}(B)$ and $A \cup \mathfrak{s}(A)$.
- Both sides of the split have at least two vertices: $|A \cup \mathfrak{s}(A)| \geq 2$ and $|B \cup \mathfrak{s}(B)| \geq 2$.

Fig. 4 shows two possible representations of a split. Notice that a split is uniquely determined just by the sets A and B, since $\mathfrak{s}(A)$ consists of connected components of $G \setminus (A \cup B)$ attached to A, and similarly for $\mathfrak{s}(B)$ and B. We refer to this split as a split *between* A and B.

In this section, we examine the recursive structure of every possible representation of G based on splits.

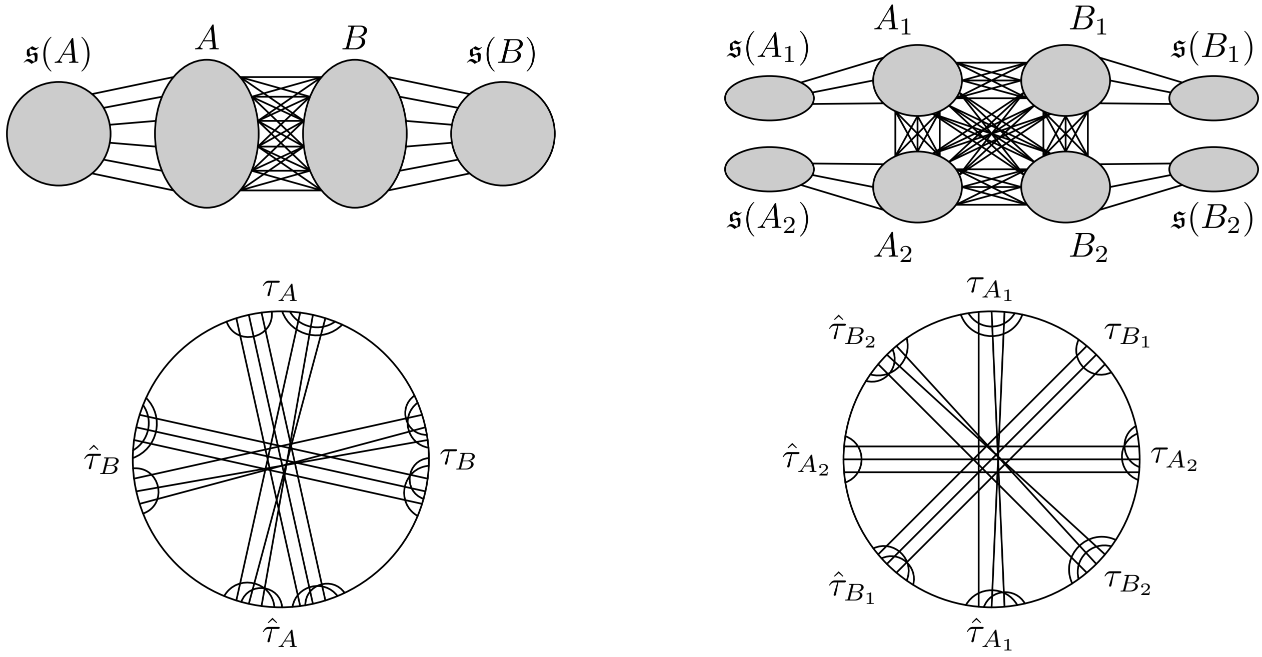

Fig. 4. Two different representations of G with the split between A and B.

3.1 Split Structure of a Representation

Let $\mathcal{R}$ be a representation of a graph G with a split between A and B. The representation $\mathcal{R}$ corresponds to a unique circular word τ and we consider the circular subsequence γ induced by $A \cup B$. The maximal subwords of γ consisting of A alternate with the maximal subwords of γ consisting of B. We denote all these maximal subwords $\gamma_1, \dots, \gamma_{2k}$ according to their circular order; so $\gamma = \gamma_1\gamma_2\cdots\gamma_{2k}$. Without loss of generality, we assume that γ_1 consists of symbols from A. We call γ_i an *A-word* when i is odd, and a *B-word* when i is even.

We first investigate for each γ_i which symbols it contains.

Lemma 4. *For the subwords $\gamma_1, \dots, \gamma_k$ the following holds:*

(a) Each γ_i contains each symbol at most once.
(b) The opposite words γ_i and γ_{i+k} contains the same symbols.
(c) Let $i \neq j$. If $x \in \gamma_i$ and $y \in \gamma_j$, then $xy \in E(G)$.

Proof. (a) Since $ab \in E(G)$ for every $a \in A$ and $b \in B$, we know that a and b alternate in the circular word γ. So if some γ_i would contain the both occurrences of, let's say, a, then a and b would not alternate in γ.

(b) Let γ_i be, let's say, an A-word. We first prove that all the other occurrences of the symbols from γ_i are contained in one word γ_j; so we get a matching between the words. Suppose that this is not true and there is $x \in \gamma_i, \gamma_j$ and $y \in \gamma_i, \gamma_{j'}$ for $j \neq j'$. There is at least one B-word γ_ℓ placed in between γ_j and

$\gamma_{j'}$ (in the part of the circle not containing γ_i). It is not possible for $z \in \gamma_\ell$ to alternate with both x and y, which gives a contradiction with $xz, yz \in E(G)$.

Now, let γ_i and γ_j be two matched A-words. Then every pair of matched B-words must occur on opposite sides of the circle with respect to γ_i and γ_j. Therefore the same number of B-words occur on both sides of γ_i and γ_j, and thus $j = i + k$.

(c) This is implied by (a) and (b) since the occurrences of x and y alternate in γ. □

If $A, B \subset V(G)$ give rise to a split in G, we call the vertices of A and B the *long vertices* with respect to the split between A and B. Similarly the vertices $\mathfrak{s}(A)$ and $\mathfrak{s}(B)$ are called *short vertices* with respect to the split between A and B. In the sequel, if the split is clear from the context, we will just call some vertices long and some vertices short.

Consider a connected component C of $\mathfrak{s}(A)$ (for a component of $\mathfrak{s}(B)$ the same argument applies) and consider the subsequence of τ induced by $A \cup B \cup C$. By Lemma 4(a)-(b) and the fact that no vertex of $\mathfrak{s}(A)$ is adjacent to B, this subsequence almost equals γ. The only difference is that one subword γ_i is replaced by a subword which additionally contains all occurrences of the vertices of C. By accordingly adding the vertices of all components of $\mathfrak{s}(A)$ and $\mathfrak{s}(B)$ to γ, we get τ. Thus, τ consists of the circular subwords $\tau_1, \ldots, \tau_{2k}$ concatenated in this order, where τ_i is obtained from γ_i by adding the components of $\mathfrak{s}(A)$ or $\mathfrak{s}(B)$ attached to it. In particular, we also have the following:

Lemma 5. *If two long vertices $x, y \in A$ are connected by a path having the internal vertices in $\mathfrak{s}(A)$, then x and y belong to the same pair γ_i and γ_{i+k} in any representation.*

Proof. If x and y belong to different subwords γ_i and γ_j, where $i < j$ and $j \neq i, i + k$, of γ, by Lemma 4(a)-(b) any path connecting x and y has an internal vertex adjacent to a vertex of B. However, no vertex in $\mathfrak{s}(A)$ is adjacent to a vertex of B. □

3.2 Conditions Forced by a Split

Now, we want to investigate the opposite relation. Namely, what can one say about a representation from the structure of a split? Suppose that x and y are two long vertices. We want to know the properties of x and y which force every representation $\mathcal{R}$ to have a subword γ_i of γ containing both x and y.

We define a relation $\sim$ on $A \cup B$ where $x \sim y$ means that x and y has to be placed in the same subword γ_i of γ. This relation is given by two conditions:

(C1) Lemma 4(c) states that if $xy \notin E(G)$, then $x \sim y$, i.e., if x and y are placed in different subwords, then C_x intersects C_y.

(C2) Lemma 5 gives $x \sim y$ when x and y are connected by a non-trivial path with all the inner vertices in $\mathfrak{s}(A) \cup \mathfrak{s}(B)$.

Let us take the transitive closure of $\sim$, which we denote by $\sim$ thereby slightly abusing the notation. Thus, we obtain an equivalence relation $\sim$ on $A \cup B$. Notice that every equivalence class of $\sim$ is either fully contained in A or in B. For the graph in Fig. 4, the relation $\sim$ has four equivalence classes A_1, A_2, B_1 and B_2.

Now, let $\varPhi$ be an equivalence class of $\sim$. We denote by $\mathfrak{s}(\varPhi)$ the set consisting of all the vertices in the connected components of $G \setminus (A \cup B)$ which have a vertex adjacent to a vertex of $\varPhi$. Since $\sim$ satisfies (C2), we know that the sets $\mathfrak{s}(\varPhi)$ of the equivalent classes of $\sim$ define a partition of $\mathfrak{s}(A) \cup \mathfrak{s}(B)$.

Recognition Algorithms Based on Splits. The splits are used in the current state-of-the-art algorithms for recognizing circle graphs. If a circle graph contains no split, it is called a *prime graph*. The representation of a prime graph is uniquely determined (up to the orientation of the circle) and can be constructed efficiently. There is an algorithm which finds a split between two sets A and B in linear time [9]. In fact, the entire *split decomposition tree* (i.e., the recursive decomposition tree obtained via splits) can be found in linear time. Usually the representation $\mathcal{R}$ is constructed as follows.

We define two graphs G_A and G_B where G_A is a subgraph of G induced by the vertices corresponding to $A \cup \mathfrak{s}(A) \cup \{v_A\}$ where the vertex v_A is adjacent to all the vertices in A and non-adjacent to all the vertices in $\mathfrak{s}(A)$, and G_B is defined similarly for B, $\mathfrak{s}(B)$, and v_B. Then we apply the algorithm recursively on G_A and G_B and construct their representations $\mathcal{R}_A$ and $\mathcal{R}_B$; see Fig. 5. It remains to join the representations $\mathcal{R}_A$ and $\mathcal{R}_B$ in order to construct $\mathcal{R}$.

To this end we take $\mathcal{R}_A$ and replace C_{v_A} by the representation of $B \cup \mathfrak{s}(B)$ in $\mathcal{R}_B$. More precisely, let the circular ordering of the endpoints of chords defined by $\mathcal{R}_A$ be $v_A \tau_A v_A \hat{\tau}_A$ and let the circular ordering defined by $\mathcal{R}_B$ be $v_B \tau_B v_B \hat{\tau}_B$. The constructed $\mathcal{R}$ has the corresponding circular ordering $\tau_A \tau_B \hat{\tau}_A \hat{\tau}_B$. It is easy to see that $\mathcal{R}$ is a correct circle representation of G.

Structure of All Representations. The above algorithm constructs a very specific representation $\mathcal{R}$ of G, and a representation like the one in Fig. 4 on the right cannot be constructed by the algorithm. In what follows we describe the structure of all the representations of a circle graph G, based on different circular orderings of the classes of $\sim$.

We choose an arbitrary circular ordering $\varPhi_1, \dots, \varPhi_\ell$ of the classes of $\sim$. Let G_i be a graph constructed from G by contracting the vertices $V(G) \setminus \big(\varPhi_i \cup \mathfrak{s}(\varPhi_i)\big)$ into one vertex v_i; i.e., G_i is defined similarly to G_A and G_B above. Let $\mathcal{R}_1, \dots, \mathcal{R}_\ell$ be arbitrary representations of $G_1, \dots, G_\ell$. We join these representations as follows. Let $v_i \tau_i v_i \hat{\tau}_i$ be the circular ordering of $\mathcal{R}_i$. We construct

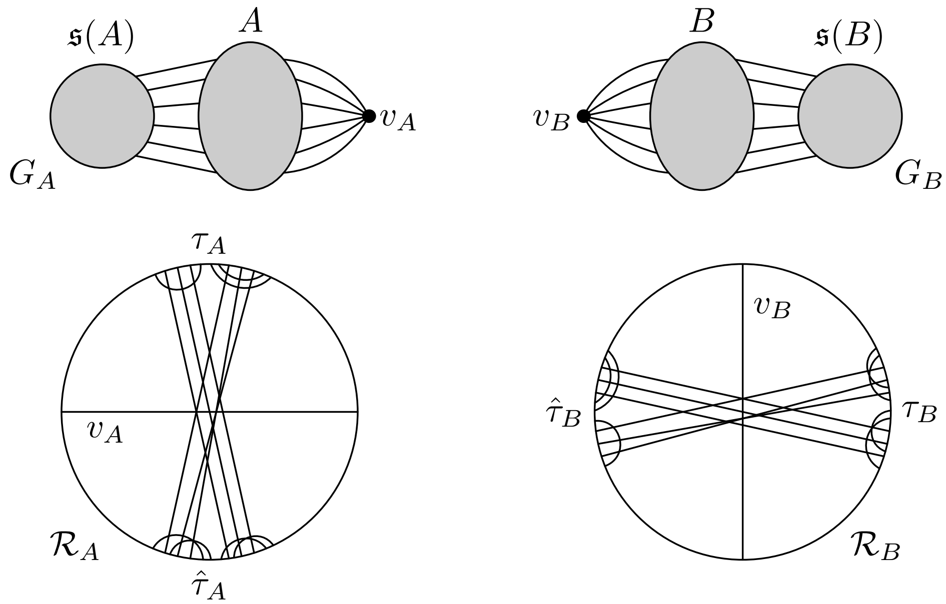

Fig. 5. The graphs G_A and G_B together with some constructed representations $\mathcal{R}_A$ and $\mathcal{R}_B$. By joining these representations, we get the left representation of Fig. 4.

$\mathcal{R}$ as the circular ordering

$$\tau_1\tau_2\ldots\tau_{k-1}\tau_k\hat{\tau}_1\hat{\tau}_2\ldots\hat{\tau}_{k-1}\hat{\tau}_k. \tag{1}$$

In Fig. 4, we obtain the representation on the left by the circular ordering $A_1A_2B_1B_2$ of the classes of $\sim$ and the representation on the right by $A_1B_1A_2B_2$.

First, we show that every representation obtained in this way is correct.

Lemma 6. *Every circular ordering (1) constructed as above defines a circle representation of G.*

Proof. Every long vertex $u \in \Phi_i$ alternates with v_i in $\mathcal{R}_i$ and every short vertex $v \in \mathfrak{s}(\Phi_i)$ has both occurrences either in τ_i, or in $\hat{\tau}_i$, since it is not adjacent to v_i. Thus, we get a correct representation $\mathcal{R}$ of G. □

Second, we prove that every representation $\mathcal{R}$ of G can be constructed like this.

Lemma 7. *Let τ be the circular word corresponding to a representation $\mathcal{R}$ of G. Then the symbols of $\Phi_i \cup \mathfrak{s}(\Phi_i)$ form exactly two subwords of τ.*

Proof. Let $\mathcal{R}$ be a representation of G and consider how it represents $A \cup B$. We get the subwords $\gamma_1, \ldots, \gamma_{2k}$ of the endpoints of $A \cup B$, as described in Section 3.1.

We claim that Φ_i is a subset of some γ_j. Suppose that some $x \in \Phi_i$ is contained in γ_j. We now use the fact that Φ_i is an equivalence class of $\sim$ to show that $\Phi_i \subseteq \gamma_j$. Let $y \in \Phi_i$ such that one of the conditions (C1) or (C2) applies to x and y. By the transitivity of $\sim$, it is sufficient to show that $y \in \gamma_j$. If (C1) applies to x and y, $y \in \gamma_j$ by Lemma 4(c). Otherwise, (C2) applies, and we are done by Lemma 5. Hence, by Lemma 4(a) all the vertices of Φ_i appear exactly once in γ_j and once in and γ_{j+k}.

Furthermore, we claim that the vertices of Φ_i form a subword of γ_j and γ_{j+k}. For the sake of contradiction suppose that a symbol $z \in \gamma_j \setminus \Phi_i$ is placed between $x \in \Phi_i$ and $y \in \Phi_i$. First, we assume that (C1) or (C2) applies to x and y.

- If (C1) applies to x and y, we know $xy \notin E(G)$. Observe that in this case z cannot be joined by an edge with both x and z. Thus $z \sim x$ or $z \sim y$, which in turn implies that $z \in \Phi_i$ (contradiction).
- Otherwise, (C2) applies to x and y. Since z has to be adjacent to both x and y, a path P from x to y having all the internal vertices in $\mathfrak{s}(\Phi_i)$ has at least one internal vertex adjacent to z. Thus, $z \sim x$ and $z \sim y$ by (C2) (again contradiction).

Finally, if $x \sim y$ and neither of (C1) and (C2) applies, we can easily proceed by an inductive argument. Indeed, if $x \sim y' \sim y$ and a vertex $z \in \gamma_j \setminus \Phi_i$ is placed between x and y in γ_j, then z is also placed in γ_j either between x and y', or between y' and y.

By the above argument, each class Φ_i forms two subwords of γ. By adding the short vertices $\mathfrak{s}(\Phi_i)$, we obtain two subwords of τ for each class Φ_i. □

Now, we are ready to prove the main structural proposition, which is inspired by Section IV.4 of the thesis of Naji [25].

Proposition 8. *Let $\sim$ be the equivalence relation defined by (C1) and (C2) on $A \cup B$. Then every representation $\mathcal{R}$ corresponds to some circular ordering $\Phi_1, \ldots, \Phi_\ell$ and to some representations $\mathcal{R}_1, \ldots, \mathcal{R}_\ell$ of $G_1, \ldots, G_\ell$. More precisely, $\mathcal{R}$ can be constructed by arranging $\mathcal{R}_1, \ldots, \mathcal{R}_\ell$ as in (1): $\tau_1 \ldots \tau_k \hat{\tau}_1 \ldots \hat{\tau}_k$.*

Proof. Let $\mathcal{R}$ be any representation with the corresponding circular word τ. According to Lemma 7, we know $\Phi_i \cup \mathfrak{s}(\Phi_i)$ forms two subwords τ_i and $\hat{\tau}_i$ of τ. For $i \neq j$, the edges between Φ_i and Φ_j form a complete bipartite graph. The subwords τ_i, $\hat{\tau}_i$, τ_j and $\hat{\tau}_j$ alternate, i.e., appear as $\tau_i\tau_j\hat{\tau}_i\hat{\tau}_j$ or $\tau_j\tau_i\hat{\tau}_j\hat{\tau}_i$ in τ. Thus, if we start from some point along the circle, the order of τ_i's gives a circular ordering $\Phi_1, \ldots, \Phi_\ell$ of the classes. The representation $\mathcal{R}_i$ is given by the circular ordering $v_i\tau_i v_i\hat{\tau}_i$. □

4 Algorithm

In this section, we give a polynomial-time algorithm for the partial representation extension problem of circle graphs. Our algorithm is based on the structure of all representations described in Section 3.

Dealing with Disconnected Graphs. To apply the structural properties of Section 3, we need to work with connected graphs. The partial representation extension problems cannot be trivially restricted to connected inputs, as in the case of most graph problems. In particular, for some classes the problems are polynomial-time solvable for connected inputs and FPT in the number of components for disconnected inputs, but NP-complete in general; see e.g. [20,21]. The reason is that the components are placed together in one representation and they restrict each other.

In the case of circle graphs, we can deal with this disconnected inputs easily. First, τ' cannot contain $axby$ as a subsequence where a, b belong to one component component and x, y to another one. If this happens, we immediately output "no". Otherwise the question of extendibility is equivalent to testing whether each component C is extendible where the partial representation of C is given by the subsequence of τ' containing all occurrences of the vertices of C. So from now on we assume that the input graph G is connected.

Overview. Let τ' be the circular word corresponding to the given partial representation $\mathcal{R}'$. We want to extend τ' to a circular word τ corresponding to a representation $\mathcal{R}$ of G.

Our algorithm proceeds recursively via split decomposition.

1. If G is prime, we have two possible representations (one is reversal of the other) and we test whether one of them is compatible with $\mathcal{R}'$.
2. Otherwise, we find a split and compute the $\sim$ relation.
3. We test whether some ordering $\Phi_1, \dots, \Phi_\ell$ of these classes along the circle is compatible with the partial representation $\mathcal{R}'$. This order is partially prescribed by short and long pre-drawn chords.
4. If no ordering is compatible, we stop and output "no". If there is an ordering which is compatible with $\mathcal{R}'$, we recurse on the graphs $G_1, \dots, G_\ell$ constructed according to the equivalence classes of $\sim$.

Now we describe everything in detail.

Prime, Degenerate and Trivial Graphs. A graph is called *prime* if it contains no split. If G is a prime graph, then it has at most two different representations $\mathcal{R}$ and $\hat{\mathcal{R}}$ [9] where one is reversal of the other. We just need to test whether one of them extends $\mathcal{R}'$.

A graph is called *degenerate* if it is isomorphic to a complete graph K_n or a star S_n. If G is degenerate, the partial representation extension problem can be easily solved for it. The reason why degenerate graphs are considered as a special case is that they are very easy to deal with, and thus they can be used to speed-up the algorithm. Of course, one could ignore them and just apply the split decomposition.

A split between A and B is called *trivial* if for one side, let us say A, we have $|A| = 1$ and $|\mathfrak{s}(A)| = 1$; so $\mathfrak{s}(A)$ is one leaf attached to A. If G contains only trivial splits, then we call it *trivial*. By removing all leaves from G, we obtain H which is either a prime graph, or a degenerate graph. If H is prime, we have two representations $\mathcal{R}_H$ and $\hat{\mathcal{R}}_H$. We just need to test whether we can add pre-drawn leaves of G. If H is degenerate, each representation of H is restricted by the order of the pre-drawn leaves of G, but this can again be easily checked.

Dealing with Non-trivial Splits. So we have a non-trivial split between A and B which can be constructed in polynomial time [9]. We compute the equivalence relation $\sim$ and we want to find an ordering of its equivalence classes. For a class $\varPhi$ of $\sim$, we define the *extended class* $\varPsi$ of $\sim$ as $\varPhi \cup \mathfrak{s}(\varPhi)$. We can assume that each extended class has a vertex pre-drawn in the partial representation, otherwise any representation of it is good. So $\sim$ has ℓ equivalence classes, and all of them appear in τ'.

Now, τ' is composed of k *maximal subwords*, each containing only symbols of one extended class $\varPsi$. We denote these maximal subwords as $\tau'_1, \dots, \tau'_k$ according to their circular order in τ', so $\tau' = \tau'_1 \cdots \tau'_k$. According to Proposition 8, each extended class $\varPsi$ corresponds to at most two different maximal subwords. Also, if two extended classes $\varPsi$ and $\hat{\varPsi}$ correspond to two different maximal subwords, then occurrences of these subwords in τ' alternate. Otherwise we reject the input.

Case 1: An extended class corresponds to two maximal subwords. We denote this class by $\varPsi_1$ and put this class as first in the ordering. By renumbering, we may assume that $\varPsi_1$ corresponds to τ'_1 and τ'_t. Then one circular order of the classes can be determined as follows. We have $\varPsi_1 < \varPsi$ for any other class $\varPsi$. Let $\varPsi_i$ and $\varPsi_j$ be two distinct classes. If $\varPsi_i$ corresponds to τ'_a and $\varPsi_j$ corresponds to τ'_b such that either $a < b < t$ or $t < a < b$, we put $\varPsi_i < \varPsi_j$. We obtain the ordering of the classes as any linear extension of $<$. One can observe that $<$ is acyclic, otherwise the maximal subwords would not alternate correctly.

Now, we have ordered the extended classes $\varPsi_1, \dots, \varPsi_\ell$ and the corresponding classes $\varPhi_1, \dots, \varPhi_\ell$. We construct each G_i with the vertices $\varPsi_i \cup \{v_i\}$ as in Section 3.2, so v_i is adjacent to $\varPhi_i$ and non-adjacent to $\mathfrak{s}(\varPhi_i)$. As the partial representation $\mathcal{R}'_i$ of G_i, we put the word $v_i \tau'_i v_i \tau'_j$ where $\varPsi_i$ corresponds to τ_i and τ_j (possibly one of them is empty). We test recursively, whether each representation $\mathcal{R}'_i$ of G_i is extendible to a representation of $\mathcal{R}_i$. If yes, we join $\mathcal{R}_1, \dots, \mathcal{R}_\ell$ as in Proposition 8. Otherwise, the algorithm outputs "no".

Lemma 9. *For Case 1, the representation $\mathcal{R}'$ is extendible if and only if the representations $\mathcal{R}'_1, \dots, \mathcal{R}'_\ell$ of the graphs $G_1, \dots, G_\ell$ are extendible.*

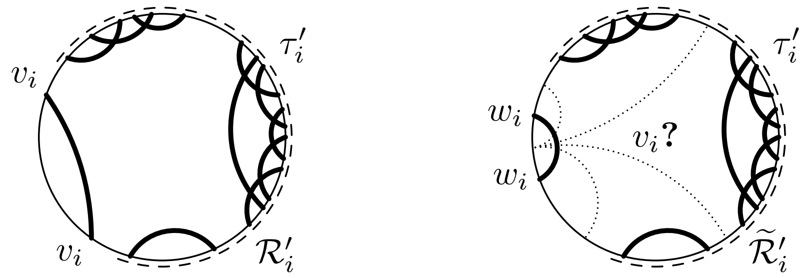

Fig. 6. The partial representation $\widetilde{\mathcal{R}}_i'$ is less restrictive with the respect to the position of v_i. Therefore it might be extendible even when $\mathcal{R}_i'$ is not.

Proof. Suppose that $\mathcal{R}$ extends $\mathcal{R}'$. According to Proposition 8, the representations of $\Psi_1, \ldots, \Psi_\ell$ are somehow ordered along the circle, and so we obtain representations $\mathcal{R}_1, \ldots, \mathcal{R}_\ell$ extending $\mathcal{R}_1', \ldots, \mathcal{R}_\ell'$.

For the other implications, we just take $\mathcal{R}_1, \ldots, \mathcal{R}_\ell$ and put them in $\mathcal{R}$ together as in (1). The ordering $<$ was constructed exactly in such a way that $\mathcal{R}$ extends $\mathcal{R}'$. □

Case 2: No extended class corresponds to two maximal subwords. In this case, we have the ordering of the classes according to their appearance in τ', so Ψ_i corresponds to the subword τ_i'. According to Proposition 8, we know that in any representation $\mathcal{R}$ of G the class Ψ_i corresponds to two subwords τ_i and $\hat{\tau}_i$. The difficulty here arises from the potential for τ_i' to be a subsequence of only one of τ_i and $\hat{\tau}_i$.

We solve this as follows. Instead of constructing just one graph G_i with one partial representation $\mathcal{R}_i'$, we construct an additional graph $\widetilde{G}_i$ with a partial representation $\widetilde{\mathcal{R}}_i'$ as follows. The graph $\widetilde{G}_i$ is G_i with an additional leaf w_i attached to v_i. The partial representation $\mathcal{R}_i'$ corresponds to the word $\tau_i' v_i v_i$ and the partial representation $\widetilde{\mathcal{R}}_i'$ corresponds to $\tau_i' w_i w_i$. The difference is that $\widetilde{\mathcal{R}}_i'$ is less restrictive and only one endpoint of v_i is prescribed (i.e., the location of the "other" end of v_i is not restricted). We can easily observe that if $\mathcal{R}_i'$ is extendible, then $\widetilde{\mathcal{R}}_i'$ is also extendible. See Fig. 6 for a comparison of the two partial representations $\mathcal{R}_i'$ and $\widetilde{\mathcal{R}}_i'$.

The following lemma is fundamental for the algorithm, and it states that at most one class can be forced to use $\widetilde{G}_i$ with $\widetilde{\mathcal{R}}_i'$, if τ' is extendible:

Lemma 10. *The representation $\mathcal{R}'$ is extendible if and only if $\widetilde{\mathcal{R}}_i'$ is extendible for some i and $\mathcal{R}_j'$ is extendible for all $j \neq i$.*

Proof. Suppose that $\mathcal{R}_j$ corresponding to a word $v_j \tau_j v_j \hat{\tau}_j$ is an extension of $\mathcal{R}_j'$ for $j \neq i$. And let $\mathcal{R}_i$ corresponding to a word $w_i v_i w_i \tau_i v_i \hat{\tau}_i$ be an extension of $\widetilde{\mathcal{R}}_i'$. Then the representation $\mathcal{R}$ (after removing w_i) constructed as in (1) extends $\mathcal{R}'$.

For the other implication, suppose that $\mathcal{R}$ extends $\mathcal{R}'$. For contradiction, suppose that two distinct partial representations $\mathcal{R}'_i$ and $\mathcal{R}'_j$ are not extendible. According to Proposition 8, the representation $\mathcal{R}$ gives a representation $\mathcal{R}_i$ corresponding to $v_i\tau_i v_i\hat{\tau}_i$ of G_i and $\mathcal{R}_j$ corresponding to $v_j\tau_j v_j\hat{\tau}_j$ of G_j. But since both Ψ_i and Ψ_j correspond to single maximal words of τ', we have that τ'_i is a subsequence of τ_i or $\hat{\tau}_i$, or τ'_j is a subsequence of τ_j or $\hat{\tau}_j$, and so $\mathcal{R}'_i$ or $\mathcal{R}'_j$ is extendible. Contradiction. □

So we have two possible subcases for Case 2, and otherwise we output "no". For an overview, see the pseudocode of Algorithm 1.

- **Case 2a: All representations are extendible.** So we have representations $\mathcal{R}_2, \dots, \mathcal{R}_\ell$ extending the partial representations where $\mathcal{R}_i$ corresponds to $v_i\tau_i v_i\hat{\tau}_i$. We test whether the partial representations $\widetilde{\mathcal{R}}'_1$ is extendible. If no, the algorithm stops and outputs "no". If yes, we get a representation $\mathcal{R}_1$ of $\widetilde{G}_1$ corresponding to $w_1v_1w_1\tau_1v_1\hat{\tau}_1$. We construct the representation $\mathcal{R}$ as in (1).
- **Case 2b: Exactly one is not extendible.** Let $\mathcal{R}'_i$ be the non-extendible representation. Then we test whether $\widetilde{\mathcal{R}}'_i$ and $\mathcal{R}'_1$ are extendible. If at least one is non-extendible, the algorithm stops and outputs "no". If both are extendible, we similarly join in $\mathcal{R}$ the representations $\mathcal{R}_1, \dots, \mathcal{R}_\ell$ according to (1).

Analysis of the Algorithm. By using the established results, we show that the partial representation extension problem of circle graphs can be solved in polynomial time.

Lemma 11. *The described algorithm correctly decides whether the partial representation $\mathcal{R}'$ of G' is extendible.*

Proof. If the input graph G is prime, degenerate or trivial, we already argued correctness of the algorithm. If the input graph G contains non-trivial split between A and B, we proceed by Case 1, or by Case 2.

For Case 1, the algorithm is correct according to Lemma 9.

Algorithm 1 The subroutine for Case 2.

1. Let Ψ_1 be the largest class (i.e., $|\Psi_i| \le n/2$ for $i > 1$).
2. **If** $\mathcal{R}'_2, \dots, \mathcal{R}'_\ell$ are extendible **then**
3. **If** $\widetilde{\mathcal{R}}'_1$ is extendible **then** ACCEPT **else** REJECT.
4. **Else if** only $\mathcal{R}'_i$ is not extendible **then**
5. **If** $\widetilde{\mathcal{R}}'_i$ and $\mathcal{R}'_1$ are extendible **then** ACCEPT **else** REJECT.
6. **Else** REJECT.

For Case 2, we have Lemma 10 which states that Case 2a or Case 2b happens, otherwise the representation in not extendible. In both cases, we recursively construct the representation $\mathcal{R}$ if it exists. □

The next lemma states that the algorithm runs in polynomial time. A precise time analysis depends on algorithm used for split decomposition, and on the order in which we choose splits for recursion. We avoid this technical analysis and just note that the degree of the polynomial is reasonable small. Certainly, it would be easy to show the complexity of order $\mathcal{O}(nm)$.

Lemma 12. *The running time of the algorithm is polynomial.*

Proof. Clearly, we can test prime, degenerate and trivial graphs in polynomial time. First we have the following special case, if, let us say, for A we have $|A| = 2$ and $|\mathfrak{s}(A)| = 0$, and we have exactly two classes in $\sim$. But this step of recursion can be solved in time $\mathcal{O}(n + m)$ plus the time for recursion on $B \cup \mathfrak{s}(B)$, for both Case 1 and Case 2. In every other situation, the number of the vertices in n is decreased, so the depth of the recursion is linear in n.

For Case 1 and Case 2a, we just spent time $\mathcal{O}(n+m)$ on G and then apply separate recursion on $G_1, \dots, G_\ell$, so the complexity is polynomial subject to each step of recursion can be done in polynomial time. For Case 2b, we get the following recursion. Let $T(n)$ denotes the time complexity of the algorithm for at most n vertices in the worst case. Then we recurse twice on Ψ_i such that $|\Psi_i| \le \frac{n}{2}$ and once on all other classes. We get:

$$T(n) \le 2 \cdot T(|\Psi_i| + 2) + \sum_{j \ne i} T(|\Psi_j| + 2) + P(n),$$

where $P(n)$ is some fixed polynomial in n. (This polynomial time is a cost of finding a split, computing equivalence classes of $\sim$, and so on.) Assuming that T is at least linear, we get that T is a convex function. Then we get

$$T(n) \le 2 \cdot T(n/2 + 2) + T(n - 1) + P(n),$$

which means that $T(n)$ is clearly polynomial. □

Now, we are ready to prove the main result of this paper.

Proof (Theorem 1). The result is implied by Lemma 11 and Lemma 12. □

5 Simultaneous Representations of Circle Graphs

In this section, we give two results concerning simultaneous representations of circle graphs: We show that this problem is NP-complete and FPT in the size of the common intersection. Formally, we deal with the following decision problem:

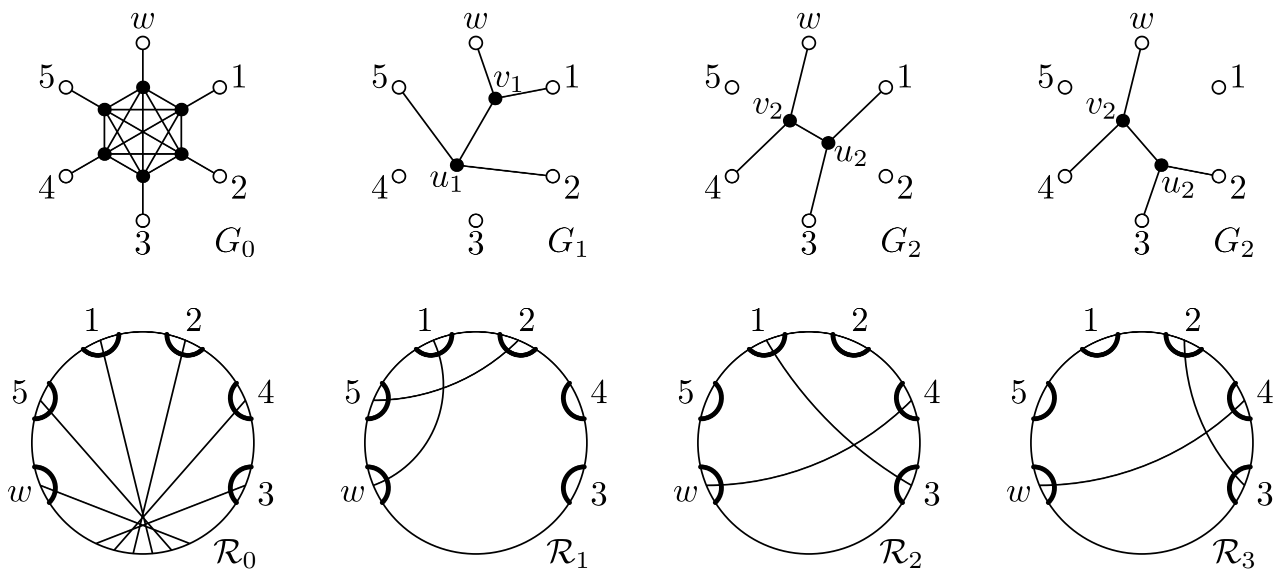

Fig. 7. Let $S = \{1, 2, 3, 4, 5\}$ and T consisting of three triples $(5, 1, 2)$, $(1, 4, 3)$ and $(2, 4, 3)$ be the instance of TotalOrdering. We construct graphs $G_0, \ldots, G_3$ depicted in the top, with the common vertices I depicted in white. Possible simultaneous representations are depicted in the bottom, giving the total ordering $5 < 1 < 2 < 4 < 3$.

Problem:	Simultaneous Representations of Circle Graphs – Sim(CIRCLE)
Input:	Graphs $G_1, \ldots, G_k$ such that $G_i \cap G_j = I$ for all $i \neq j$.
Output:	Do there exist representations $\mathcal{R}_1, \ldots, \mathcal{R}_k$ of $G_1, \ldots, G_k$ which use the same representation of the vertices of I?

Proof (Proposition 2). To show that Sim(CIRCLE) is NP-complete, we reduce it from the *total ordering problem*:

Problem:	The total ordering problem - TotalOrdering
Input:	A finite sets S and a finite set T of triples from S.
Output:	Does there exist a total ordering $<$ of S such that for all $(x, y, z) \in T$ either $x < y < z$, or $z < y < x$?

Opatrny [26] proved this problem is NP-complete.

Given an instance (S, T) of TotalOrdering and let $s = |S|$ and $t = |T|$. We construct a set of $t + 1$ graphs $G_0, G_1, \ldots, G_t$ as follows, so the number k from Sim(CIRCLE) is equal $t + 1$. The intersection of $G_0, G_1, \ldots, G_t$ is an independent set $I = S \cup \{w\}$ where w is a special vertex. The graph G_0 consists of a clique K_{s+1}, and to each vertex of this clique we attach exactly one vertex of I as a leaf. The graph G_i corresponds to the i-th constraint $(x_i, y_i, z_i) \in T$. In addition to I, each G_i contains two vertices u_i and v_i of degree three, such that u_i is adjacent to v_i, x_i and z_i, and v_i is further adjacent to y_i and the special vertex w. See Fig. 7 for an example of this construction.

The clique in G_0 defines a split where each class of $\sim$ is a singleton. According to Proposition 8, every representation $\mathcal{R}_0$ of G_0 places the elements of I in some circular ordering $wws_1s_1s_2s_2\cdots s_ss_s$ which corresponds to the total ordering $s_1 < s_2 < \cdots < s_s$. Now the representations $\mathcal{R}_1,\dots,\mathcal{R}_t$ of $G_1,\dots,G_t$ can be constructed if and only if all the total ordering constraints are satisfied. This implies that there exist simultaneous representations $\mathcal{R}_0,\dots,\mathcal{R}_t$ of $G_0,\dots,G_t$ if and only if the instance (S,T) of TotalOrdering is solvable. □

Further, we show that the problem is FPT in size of the common subgraph I.

Proof (Corollary 3). We just consider all possible representations of the common subgraph I which are all words of length $2|V(I)|$. Each word gives some partial representation $\mathcal{R}'$. We just solve k instance of RepExt(CIRCLE) for each G_i and the partial representation $\mathcal{R}'$ of I, which can be done in polynomial time according to Theorem 1. □

6 Conclusions

The structural results described in Section 3, namely Proposition 8, are the main new tools developed in this paper. Using it, one can easily work with the structure of all representations which is a key component of the algorithm of Section 4 that solves the partial representation extension problem for circle graphs. The algorithm works with the recursive structure of all representations and matches the partial representation on it. Proposition 8 also seems to be useful in attacking the following open problems:

Question 13. What is the complexity of Sim(CIRCLE) for a fixed number k of graphs? In particular, what is it for $k = 2$?

Recall that in the bounded representation problem, we give two circular arcs A_v and A'_v for each chord v, and we want to construct a representation which places one endpoint of v in A_v and the other endpoint in A'_v.

Question 14. What is the complexity of the bounded representation problem for circle graphs?

References

1. P. Angelini, G. D. Battista, F. Frati, V. Jelínek, J. Kratochvíl, M. Patrignani, and I. Rutter. Testing planarity of partially embedded graphs. In *SODA'10*, pages 202–221, 2010.
2. M. Balko, P. Klavík, and Y. Otachi. Bounded representations of interval and proper interval graphs. *To appear in ISAAC*, 2013.
3. T. Bläsius and I. Rutter. Simultaneous PQ-ordering with applications to constrained embedding problems. In *SODA'13*, pages 1030–1043, 2013.
4. A. Bouchet. Reducing prime graphs and recognizing circle graphs. *Combinatorica*, 7(3):243–254, 1987.
5. A. Bouchet. Unimodularity and circle graphs. *Discrete Mathematics*, 66(1-2):203–208, 1987.
6. M. Chudnovsky, N. Robertson, P. Seymour, and R. Thomas. The strong perfect graph theorem. *Annals of Mathematics*, 164:51–229, 2006.
7. B. Courcelle. Circle graphs and monadic second-order logic. *J. Applied Logic*, 6(3):416–442, 2008.
8. W. Cunningham. Decomposition of directed graphs. *SIAM J. Alg. and Disc. Methods*, 3:214–228, 1982.
9. E. Dahlhaus. Parallel algorithms for hierarchical clustering and applications to split decomposition and parity graph recognition. *Journal of Algorithms*, 36(2):205–240, 1998.
10. P. Damaschke. The hamiltonian circuit problem for circle graphs is NP-complete. *Information Processing Letters*, 32(1):1–2, 1989.
11. H. de Fraysseix. Local complementation and interlacement graphs. *Discrete Mathematics*, 33(1):29–35, 1981.
12. H. de Fraysseix and P. O. de Mendez. On a characterization of gauss codes. *Discrete & Computational Geometry*, 22(2):287–295, 1999.
13. S. Even and A. Itai. Queues, stacks, and graphs. *Theory of Machines and Computation (Z. Kohavi and A. Paz, Eds.)*, pages 71–76, 1971.
14. C. P. Gabor, K. J. Supowit, and W. Hsu. Recognizing circle graphs in polynomial time. *J. ACM*, 36(3):435–473, 1989.
15. M. Garey, D. Johnson, G. Miller, and C. Papadimitriou. The complexity of coloring circular arcs and chords. *SIAM Journal on Algebraic Discrete Methods*, 1(2):216–227, 1980.
16. F. Gavril. Maximum weight independent sets and cliques in intersection graphs of filaments. *Information Processing Letters*, 73(5-6):181–188, 2000.
17. E. Gioan, C. Paul, M. Tedder, and D. Corneil. Practical and efficient circle graph recognition. *Algorithmica*, pages 1–30, 2013.
18. K. R. Jampani and A. Lubiw. The simultaneous representation problem for chordal, comparability and permutation graphs. *Journal of Graph Algortihms and Applications*, 16(2):283–315, 2012.
19. P. Klavík, J. Kratochvíl, T. Krawczyk, and B. Walczak. Extending partial representations of function graphs and permutation graphs. In Leah Epstein and Paolo Ferragina, editors, *Algorithms ESA 2012*, volume 7501 of *LNCS*, pages 671–682. 2012.
20. P. Klavík, J. Kratochvíl, Y. Otachi, I. Rutter, T. Saitoh, M. Saumell, and T. Vyskočil. Extending partial representations of proper and unit interval graphs. *In preparation.*, 2013.
21. P. Klavík, J. Kratochvíl, Y. Otachi, and T. Saitoh. Extending partial representations of subclasses of chordal graphs. In Kun-Mao Chao, Tsan-sheng Hsu, and Der-Tsai Lee, editors, *Algorithms and Computation*, volume 7676 of *LNCS*, pages 444–454. 2012.
22. P. Klavík, J. Kratochvíl, Y. Otachi, T. Saitoh, and T. Vyskočil. Linear-time algorithm for partial representation extension of interval graphs. *In preparation.*, 2013.

23. P. Klavík, J. Kratochvíl, and T. Vyskočil. Extending partial representations of interval graphs. In Mitsunori Ogihara and Jun Tarui, editors, *Theory and Applications of Models of Computation*, volume 6648 of *LNCS*, pages 276–285. 2011.
24. A. Kostochka and J. Kratochvíl. Covering and coloring polygon-circle graphs. *Discrete Mathematics*, 163(1-3):299–305, 1997.
25. W. Naji. *Graphes de Cordes: Une Caracterisation et ses Applications.* PhD thesis, l'Université Scientifique et Médicale de Grenoble, 1985.
26. J. Opatrny. Total ordering problem. *SIAM J. on Computing*, 8(1):111–114, 1979.
27. S. Oum. Rank-width and vertex-minors. *J. Comb. Theory, Ser. B*, 95(1):79–100, 2005.
28. M. Patrignani. On extending a partial straight-line drawing. In Patrick Healy and Nikola S. Nikolov, editors, *Graph Drawing*, volume 3843 of *LNCS*, pages 380–385. 2006.
29. J. P. Spinrad. Recognition of circle graphs. *J. of Algorithms*, 16(2):264–282, 1994.
30. J. P. Spinrad. *Efficient Graph Representations.* Field Institute Monographs, 2003.