Extending partial representations of circle graphs

Steven Chaplick1* | Radoslav Fulek2 | Pavel Klavík3

1Lehrstuhl für Informatik I, Universität Würzburg, Würzburg, Germany
2Institute of Science and Technology Austria, Klosterneuburg, Austria
3Department of Applied Mathematics, Charles University, Prague, Czech Republic

Correspondence
Pavel Klavík, Department of Applied Mathematics, Charles University, Prague, Czech Republic 118 00.
Email: klavik@kam.mff.cuni.cz

Funding information
Grantová Agentura České Republiky, Grant/Award Number: P202/12/G061; Grantová Agentura, Univerzita Karlova, Grant/Award Number: 1334217; Seventh Framework Programme, Grant/Award Number: FP7/2007-2013; European Union’s Seventh Framework Programme; REA, Grant/Award Number: 291734

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 $R$ giving some predrawn chords that represent an induced subgraph of $G$. The question is whether one can extend $R$ to a representation $R'$ of the entire graph $G$, that is, whether one can draw the remaining chords into a partially predrawn representation to obtain a representation of $G$. Our main result is an $O(n^3)$ time algorithm for partial representation extension of circle graphs, where $n$ is the number of vertices. To show this, we describe the structure of all representations of a circle graph using split decomposition. This can be of independent interest.

KEYWORDS
algorithm, circle graphs, partial representation extension, recognition, split decomposition

1 | INTRODUCTION

Geometric graph representations are important topics of graph theory and computer science. A frequently studied type of representations is the 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, that is, two vertices are adjacent if and only if the corresponding objects intersect. Classes of intersection graphs are obtained by restricting these...
objects; for example, interval graphs are intersection graphs of intervals of the real line, string graphs are intersection graphs of curves in the plane, and so on. These representations are well studied (see, eg, [40]).

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 [29]. 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 Figure 1 for an illustration. We 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 [19] in the early 1970s in study of stack sorting techniques. Other motivations are due to their relations to Gauss words [18] (see Figure 2) and matroid representations [8,17]. Circle graphs are also important regarding rank width [35].

Let $\chi(G)$ denote the chromatic number of $G$, and let $\omega(G)$ denote the clique number of $G$. Trivially we have $\omega(G) \leq \chi(G)$ and the graphs for which every induced subgraph satisfies equality are the well-known perfect graphs [12]. In general, the difference between these two numbers can be arbitrarily high, for example, there is a triangle-free graph with an arbitrary high chromatic number. Circle graphs are known to be almost perfect which means that $\chi(G) \leq f(\omega(G))$ for some function $f$. The best known result for circle graphs [31] states that $f(k)$ is $\Omega(k \log k)$ and $O(2^k)$.

Some NP-hard problems, such as maximum weighted clique and independent set [22], become tractable on circle graphs. Contrarily, problems such as vertex colorability [21] and Hamiltonicity [16] remain NP-complete even for circle graphs.

The complexity of recognition of circle graphs was a long-standing open problem; see [40] for an overview. The first results, for example, [19], 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 [7,20,33] (in time $O(n^7)$ and similar). Later, a more efficient algorithm [39] based on split decomposition was given, and the current state-of-the-art recognition algorithm [23] 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. Contrarily, Fáry’s Theorem claims that every planar graph has a straight-line embedding, but extension of such an embedding is NP-hard [36].

In the context of intersection-defined classes, this problem was first considered in [29] for interval graphs. Currently, the best known results are linear-time algorithms for interval graphs [5, 28] and proper interval graphs [26], a quadratic-time algorithm for unit interval graphs [26, 37, 38], and polynomial-time algorithms for permutation and function graphs [25], proper circular-arc graphs [3], and trapezoid graphs [32]. For chordal graphs (as subtree-in-a-tree graphs) several versions of the problems were considered [27] and all of them are NP-complete, and similarly for different contact representations of planar graphs [9]. In [30], minimal forbidden configurations making a partial interval representation nonextendible are characterized. Extending partial visibility representations is studied in [11].

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 $R'$. Then we efficiently test whether some representation $R$ satisfies these constraints. If none satisfies them, then $R'$ is not extendible. And if some $R$ satisfies them, then it extends $R'$.

It is well known that the split decomposition [14] captures the structure of all representations of circle graphs. The standard recognition algorithms produce a special type of representation 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 $R$. In Section 3, we give a simple recursive description of all possible representations based on splits. Our result can be interpreted as “describing a structure like PQ-trees” for circle graphs.

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

\[^1\text{See [6] for further information on PQ-trees.}\]
graphs.” It is possible that the proof techniques from other papers on circle graphs such as [13,23] would give a similar description. However, these techniques are more involved than our approach which turns 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 representation problem\(^2\), shortly Sim, consists of graphs \( G_1, \ldots, G_k \) with some vertices common for all the graphs. The problem asks whether there exist representations \( R_1, \ldots, R_k \) representing the common vertices in the same way. This problem is polynomially solvable for permutation and comparability graphs [24]. 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 \( k = 2 \) [5] 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 [29], and partial representations of interval graphs can be extended in linear time by reducing it to corresponding simultaneous representation problem [5].

The bounded representation problem [26] 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 convert an instance of the partial representation extension problem into an instance of the bounded representation problem by replacing every occurrence of a vertex by a small circular arc (such that no two circular arcs so introduced intersect) and prescribing a circular arc that covers the whole circle for every endpoint that does not appear in the instance. It is known to be polynomially solvable for interval and proper interval representations of interval graphs [2], and surprisingly it is NP-complete for unit interval representations [26,37,38]. The complexity for other classes is not known.

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

**Problem:** Partial representation extension—\( \text{REPEXT(CIRCLE)} \)

**Input:** A circle graph \( G \) and a partial representation \( R' \).

**Output:** Is there a representation \( R \) of \( G \) extending \( R' \)?

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 \( \text{REPEXT(CIRCLE)} \) can be solved in time \( O(n^3) \) where \( n \) is the number of vertices.

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

\(^2\)Here, we will focus on what is sometimes referred to as the sunflower version in the literature, see [4].
Theorem 2. If \( k \) is a part of the input, the problem \( \text{Sim}(\text{CIRCLE}) \) of \( k \) circle graphs is NP-complete.

Finally, we show that Theorem 1 implies the following.

Corollary 1. The problem \( \text{Sim}(\text{CIRCLE}) \) is \( \text{FPT} \) in the size of the common subgraph.

2 | DEFINITIONS AND PRELIMINARIES

Circle representations. A circle representation \( R \) of a graph \( G \) is a collection \( \{C_u | u \in V(G)\} \) 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 \( \text{CIRCLE} \).

Notice that a representation of a circle graph is completely determined by the 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.

Interval overlap graphs. Suppose that we cut the circle in a point which is not an endpoint of a chord and straighten it into a segment (see Figure 3). From this straightening of the circle, each chord can now be seen as an arc above the resulting segment. Notice that two chords \( C_u \) and \( C_v \) cross if and only if their endpoints appear in the order \( uvuv \) or \( vuvu \) from left to right. Alternatively, circle graphs are called interval overlap graphs. Their vertices can be represented by intervals and two vertices 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 \( \tau \) over an alphabet of symbols \( \Sigma \) is a word. A circular word represents the 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 \( \tau \) and a symbol \( u \), we write \( u \in \tau \), if \( u \) appears at least once in \( \tau \). Thus, \( \tau \) is also used to denote the set of symbols occurring in \( \tau \). A word \( \tau \) is a subword of \( \sigma \), if \( \tau \) appears consecutively in \( \sigma \). A word \( \tau \) is a subsequence of \( \sigma \), if the word \( \tau \) can be obtained from \( \sigma \) by deleting some symbols. We say that \( u \) alternates with \( v \) in \( \tau \), if \( uvuv \) or \( vuvu \) is a subsequence of \( \tau \). The corresponding definitions also apply to circular words. If \( \sigma \) and \( \tau \) are two words, we denote their concatenation by \( \sigma \tau \).

The above interpretation of circle graphs as interval overlap graphs allows us to associate each representation \( R \) of \( G \) with a unique circular word \( \tau \) over \( V \). The word \( \tau \) 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 \( \tau \) if and only if \( uv \in E(G) \). For example, \( \mathcal{R} \) in Figure 1 corresponds to the circular word \( \tau = susxxtutwvuw \). Notice that each vertex appears exactly twice in \( \tau \). A circular subsequence \( \tau' \) of \( \tau \) is induced by \( V' \subseteq V(G) \) if \( \tau' \) is obtained from \( \tau \) by deleting symbols in \( V(G) \setminus V' \).

**Partial representations.** Partial representations are defined in [29] and other papers as representations of induced subgraphs. In this paper, we consider the following more general definition. A partial representation \( \mathcal{R}' \) of a circle graph \( G \) is given by a circular word \( \tau' \) consisting of symbols of \( V(G) \) such that each \( u \in V(G) \) appears at most twice in \( \tau' \). A representation \( \mathcal{R} \) of \( G \) corresponding to a circular word \( \tau \) extends \( \mathcal{R}' \) if and only if \( \tau' \) is a subsequence of \( \tau \). The endpoints in \( \tau' \) and the corresponding vertices are called predrawn. If a predrawn vertex \( u \) has both occurrences in \( \tau' \), the chord \( C_u \) is predrawn.

## 3  STRUCTURE OF REPRESENTATIONS OF MAXIMAL SPLITS

Let \( G \) be a connected graph. A split of \( G \) is a partition of the set of vertices of \( G \) into four parts \( A \), \( B \), \( s(A) \) and \( s(B) \), such that

- We have \( A \neq \emptyset \) and \( B \neq \emptyset \), but possibly \( s(A) = \emptyset \) or \( s(B) = \emptyset \).
- For every \( a \in A \) and \( b \in B \), we have \( ab \in E(G) \).
- There is no edge between \( s(A) \) and \( B \cup s(B) \), and between \( s(B) \) and \( A \cup s(A) \).

Figure 4 shows two possible representations of a split. As usual by \( G[V'] \), for \( V' \subseteq V \), we denote the subgraph of \( G \) induced by \( V' \). We use \( G \setminus V' \) as a shorthand notation for \( G[V \setminus V'] \). Notice that a split is uniquely determined just by the sets \( A \) and \( B \), since \( s(A) \) consists of the vertices in the connected components of \( G' \setminus (A \cup B) \), attached to \( A \), and \( s(B) \) of those of the connected components of \( G' \setminus (A \cup B) \) attached to \( B \). We refer to this split as the split between \( A \) and \( B \). Alternatively, a split between \( A \) and \( B \) is a cut in \( G \) between \( A \) and \( B \) which is a complete bipartite graph.

The standard assumption is that a split is nontrivial, meaning that both sides of the split have at least two vertices: \( |A \cup s(A)| \geq 2 \) and \( |B \cup s(B)| \geq 2 \). The reason is that trivial splits are not very interesting: in every graph \( G \), the choice \( A = \{u\} \) and \( B = N(u) \) for \( u \in V(G) \) forms a trivial split. The goal of split decomposition is to divide a graph into smaller graphs and trivial splits are not helpful.

One of the novelties of this paper is that we study maximal splits. A split of \( G \) between \( A \) and \( B \) is maximal if there exists no split of \( G \) between \( A' \) and \( B' \) such that \( A \subseteq A' \), \( B \subseteq B' \) and \( |A| < |A'| \) or \( |B| < |B'| \). Both splits between \( A \) and \( B \) and between \( A' \) and \( B' \) are allowed to be trivial. Maximal splits satisfy the following property:

**Lemma 1.** A split between \( A \) and \( B \) is maximal, if and only if there exists no connected component \( C \) in \( G[s(A)] \) such that each vertex of \( C \) is either adjacent to all vertices of \( A \), or to none of them, and similarly for \( s(B) \) and \( B \).

**Proof.** Suppose that such a component \( C \) in \( G[s(A)] \) exists and let \( C' \subseteq V(C) \) consist of those vertices which are adjacent to all vertices of \( A \). The split between \( A \) and \( B \) is not
maximal since $A$ together with $B \cup C'$ forms a split for which $V(C) \setminus C' \subseteq s(B \cup C')$. Similarly, if such a component $C$ in $G[s(B)]$ exists, the split between $A$ and $B$ is not maximal.

Contrarily, suppose that a split between $A$ and $B$ is not maximal, so there exists a split between $A'$ and $B'$ such that, without loss of generality, $A \subseteq A'$ and $B \subseteq B'$. Since every vertex of $B' \setminus B$ is adjacent to all vertices in $A$, we have $B' \setminus B \subseteq s(A)$.

Choose an arbitrary $c \in B' \setminus B$ and let $C$ be the connected component of $G[s(A)]$ containing $c$. It holds that $V(C) \cap A' = \emptyset$. Indeed, $V(C) \subseteq s(A)$, and each vertex in $A'$ is adjacent to each vertex in $B'$. It follows by the connectivity of $C$ that also $V(C) \cap s(A') = \emptyset$.

As argued, all vertices of $V(C) \cap B'$ are adjacent to all vertices of $A$. Since $V(C) \cap B' \neq \emptyset$ and $V(C) \cap (A' \cup s(A')) = \emptyset$, the remaining vertices $V(C) \setminus B' \subseteq s(B')$. Therefore, they are nonadjacent to all vertices $A$, and $C$ satisfies the properties from the statement of this lemma.

We always start with a nontrivial split between $A$ and $B$, and modify it using Lemma 1 into a maximal split which may become trivial. But such a trivial maximal split has a special structure, described below.

**Lemma 2.** Let $A$ and $B$ form a nontrivial split and let $A'$ and $B'$ form a trivial maximal split such that $A \subseteq A'$, $B \subseteq B'$, $A' = \{a\}$, and $s(A') = \emptyset$. Then $a$ is an articulation in $G$, that is, $G \setminus a$ is disconnected.

**Proof.** Since $A \neq \emptyset$, we have $A = \{a\}$. Since the split between $A$ and $B$ is nontrivial, we have $s(A) \neq \emptyset$. Therefore, $a$ is an articulation in $G$ which separates $s(A)$ from $B \cup s(B)$. 

In the rest of this section, we examine the recursive structure of every possible representation of $G$ based on maximal splits. In Section 3.1, we analyze the structure of a representation of a maximal split. In Section 3.2, we use it to describe the structure of all circle representations. The described results still apply to trivial maximal splits, but are not very helpful. Therefore, in Section 3.3, we give a different description of all representations based on trivial maximal splits.
3.1 Structure of a representation of a maximal split

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

We first investigate for each \( \gamma_i \) which symbols it contains.

**Lemma 3.** For the subwords \( \gamma_1, \ldots, \gamma_k \) the following holds:

(a) Each \( \gamma_i \) contains each symbol at most once.

(b) The value of \( k \) is even and the opposite words \( \gamma_i \) and \( \gamma_{i+k} \) contain 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) For every \( a \in A \) and \( b \in B \), the fact \( ab \in E(G) \) implies that \( a \) and \( b \) alternate in the circular word \( \gamma \). So if some \( \gamma_i \) contains both occurrences of, say, \( a \), then \( a \) and \( b \) would not alternate in \( \gamma \).

(b) Let \( \gamma_i \) be, say, an \( A \)-word. We first prove that all the other occurrences of the symbols from \( \gamma_i \) are contained in one word \( \gamma_j \); so we get a matching between the words. Suppose that this is not true and there is \( x \in \gamma_i \), \( y \in \gamma_{j} \) for distinct \( i \), \( j \), and \( j' \). There is at least one \( B \)-word \( \gamma_{z} \) placed in between \( \gamma_j \) and \( \gamma_{j'} \) (in the part of the circle not containing \( \gamma_i \)). It is not possible for \( z \in \gamma_{z} \) to alternate with both \( x \) and \( y \), which contradicts \( xz, yz \in E(G) \).

Now, let \( \gamma_i \) and \( \gamma_j \) be two matched \( A \)-words. Then every pair of matched \( B \)-words must occur on opposite sides of the circle with respect to \( \gamma_i \) and \( \gamma_j \). Therefore the same number of \( B \)-words occurs on both sides of \( \gamma_i \) and \( \gamma_j \), and thus \( j = i + k \). In particular, \( \gamma_{k+1} \) is an \( A \)-word and therefore \( k \) is even.

(c) This is implied by (a) and (b) since the occurrences of \( x \) and \( y \) alternate in \( \gamma \). \( \square \)

Below, we prove that the structure of a maximal split between \( A \) and \( B \) greatly restricts possible representation of the vertices of \( s(A) \cup s(B) \):

**Lemma 4.** Let \( \tau \) and \( \gamma_1, \ldots, \gamma_{2k} \) be defined as above. There exists a unique mapping \( f: s(A) \cup s(B) \to \{1, \ldots, 2k\} \) satisfying the following properties:

(a) For \( c \in s(A) \cup s(B) \), let \( ct_{c}, ct'_{c} \) be the subsequence of \( \tau \) induced by \( A \cup B \cup \{c\} \). Then either \( \tau_{c} \) or \( \tau'_{c} \) is a subword of \( \gamma_{f(c)} \). For \( c \in s(A) \), the word \( \gamma_{f(c)} \) is an \( A \)-word, while for \( c \in s(B) \), it is a \( B \)-word.

(b) For each connected component \( C \) of \( s(A) \cup s(B) \), the mapping \( f|_{C} \) is constant, that is, for all \( c, c' \in C \), we have \( f(c) = f(c') \), and we denote the image by \( f(C) \).
Proof. Without loss of generality, we assume that \( c \in s(A) \); a symmetric argument works for \( c \in s(B) \). We first prove the existence and uniqueness of \( \gamma_f(c) \) when \( c \) is adjacent to some vertex in \( a \in A \). Since \( c \) alternates with \( a \), both \( \tau_c \) and \( \tau_c' \) are nonempty. In (b), we prove that \( f(c) = f(c') \) when \( cc' \in E(G) \), so by induction the existence and uniqueness follows for all vertices of \( C \).

(a) Since \( c \) alternates with \( a \in A \), if such \( \gamma_f(c) \) exists, then it is an \( A \)-word. Since \( A \)-words and \( B \)-words alternate in \( \gamma = \gamma_1 \cdots \gamma_{2k} \), we get that \( \tau_c \) is a subword of some \( A \)-word \( \gamma \) if and only if it contains no symbol from \( B \). Since at most one of \( \tau_c \) and \( \tau_c' \) contains no symbol from \( B \), it is easy to see that such \( \gamma \) is unique. It remains to prove that it always exists.

Let \( C \) be the connected component of \( s(A) \) containing \( c \). For contradiction, suppose that the property (a) fails for \( c \). If property (a) fails for \( c \), we have \( b \in \tau_c \) and \( b' \in \tau_c' \) such that \( b, b' \in \tau_c \). Since \( bc, b'c \notin E(G) \), we have \( b \neq b' \) and \( b'b' \notin E(G) \) (see Figure 5A).

For each \( a \in A \), we have \( ab, ab' \in E(G) \), so \( A \cup \{b, b', c\} \) induces in \( \tau \) the subsequence \( xbabcb' \alpha \beta \) such that both \( \alpha \) and \( \alpha' \) consist only of symbols from \( A \). For each \( a \in A \), we have \( a \in \alpha \) and \( a \in \alpha' \), so \( c \) is adjacent to all vertices of \( A \). Suppose \( x, y \in s(A) \) such that \( xy \in E(G) \). If \( A \cup \{x, b, b'\} \) induces the subsequence \( xbabcb' \alpha \beta \) or \( xxbabcb' \alpha \beta \) in \( \tau \), then since both \( x \) and \( y \) are nonadjacent to both \( b \) and \( b' \), we have that \( A \cup \{y, b, b'\} \) induces the subsequence \( ybabcb' \alpha \beta \) or \( yxbabcb' \alpha \beta \) or \( bababcb' \alpha \beta \) in \( \tau \).

Therefore, every vertex in \( C \) is connected by a path to \( c \), we have that \( A \cup \{b, b'\} \cup V(C) \) induces in \( \tau \) the subsequence \( xbabcb' \alpha \beta \), where both \( \sigma \) and \( \sigma' \) consist of symbols of \( V(C) \) (but \( \sigma \) and \( \sigma' \) may not necessarily consist of the same symbols). Therefore every \( c' \in V(C) \) is either adjacent to all vertices of \( A \), or to none of them. By Lemma 1, the split between \( A \) and \( B \) is not maximal.

(b) Let \( c, c' \in s(A) \) such that \( cc' \in E(C) \) and \( f(c) \) is already determined. We want to prove that \( f(c) = f(c') \). As depicted in Figure 5B, let \( \gamma_c, c_c \) be the subsequence of \( \tau \) induced by \( A \cup B \cup \{c, c'\} \), and suppose that \( \tau_c \) is a subword of \( \gamma_f(c) \). Both \( \tau_c \) and \( \tau_c' \) cannot contain symbols from \( B \), otherwise \( c ' \) alternates with each \( a \in A \), and the argument in (a) applies, telling us that there exists an \( A \)-word that contains \( \tau_c \) or \( \tau_c' \), contradicting the fact that both \( \tau_c \) and \( \tau_c' \) contain symbols from \( B \). Therefore, at least one of

---

**Figure 5** A, On the left, a connected component \( C \) of \( s(A) \) attached to \( A \). On the right, the circular subsequences of \( \tau \) induced by \( A \cup \{b, b', c\} \) and \( A \cup \{b, b'\} \cup C \). B, The circular subsequence \( c\tau_c c' \tau_c' \) induced by \( A \cup B \cup \{c, c'\} \), with the word \( \gamma_f(c) \) depicted. Exactly one of \( \tau_c \) and \( \tau_c' \) contains the symbols \( b \in B \) [Color figure can be viewed at wileyonlinelibrary.com]
\[ \tau_c \] and \( \tilde{\tau}_c \) does not contain symbols from \( B \), implying that either \( \tilde{\tau}_c \tau_c \) or \( \tilde{\tau}_c \tilde{\tau}_c \) is a subword of \( \gamma_f(c) \), so \( f(c) = f(c') \). We note that either \( \tilde{\tau}_c \tau_c \) or \( \tilde{\tau}_c \tilde{\tau}_c \) might be empty, so \( f(c') \) could be chosen arbitrarily to satisfy (a). We then set \( f(c') = f(c) \) to also satisfy (b).

Let \( \tau \) denote the subsequence of \( \tau \) formed by \( \gamma \), and of the symbols of \( \bigcup_{c \in f(C)} V(C) \) over all connected components \( C \) of \( s(A) \cup s(B) \). By Lemma 4, and the fact that a vertex in \( s(A) \) cannot alternate with a vertex in \( s(B) \), it can be concluded that the only difference between \( \gamma \) and \( \tau \) is that each subword \( \gamma_i \) is replaced by the subword \( \tau_i \) which additionally contains all occurrences of the vertices in some connected components of \( s(A) \) or \( s(B) \). Thus, \( \tau = \tau_1 \tau_2 \cdots \tau_k \).

Lemma 4 explains the following naming convention used for maximal splits between \( A \) and \( B \) in this paper (see Figure 4). We call the vertices of \( A \) and \( B \) as long vertices with respect to the maximal split between \( A \) and \( B \) since each is represented by “long chords” between \( \tau_i \) and \( \tilde{\tau}_i+k \). The vertices \( s(A) \) and \( s(B) \) are called short vertices with respect to the maximal split between \( A \) and \( B \), because each is represented by “short chords” inside some \( \tau_i \). In the sequel, if the maximal split is clear from the context, we will just call some vertices long and some vertices short.

**Lemma 5.** If two long vertices \( x, y \in A \cup B \) are connected by a path of length at least two having the internal vertices in \( s(A) \cup s(B) \), then \( x \) and \( y \) belong to the same pair \( \gamma_i \) and \( \gamma_{i+k} \) in every representation.

**Proof.** Let \( C \) be the connected component of \( s(A) \cup s(B) \) having the internal vertices of this path between \( x \) and \( y \). By Lemma 4, all vertices of \( C \) have both symbols in \( \gamma_f(c) \). Therefore, \( x, y \in \tau_f(c) \). So \( x, y \in \gamma_f(c) \), and by Lemma 3(b) also \( x, y \in \gamma_f(c)+k \).

Also, we prove the following simple lemma.

**Lemma 6.** Let \( x, y, z, \) and \( w \) be distinct vertices inducing a clique in \( G \), and let \( P \) be a path from \( x \) to \( y \) of length at least 2. If \( xyzwxyzw \) is a subsequence of the circular word \( \tau \) of a circle representation of \( G \), then some internal vertex of \( P \) is adjacent to \( z \) or \( w \).

**Proof.** Let \( v_1, \ldots, v_k \) be the internal vertices of \( P \) such that \( v_1x \in E(G) \). We prove by induction that no \( v_i \) having \( v_1z \in E(G) \) or \( v_1w \in E(G) \) implies that \( v_ky \notin E(G) \). If \( v_1 \) is not such a vertex, then we get that \( v_1xv_1zwxyzw \) is a subsequence of \( \tau \) since \( v_1x \in E(G) \). For the induction hypothesis, suppose that \( v_i v_{i+1}zwxyzw \) is a subsequence of \( \tau \). Since \( v_i v_{i+1} \in E(G) \), if \( v_{i+1} \) is not adjacent to \( z \) and \( w \), we get that \( v_{i+1}v_i+1zwxyzw \) is a subsequence of \( \tau \). Therefore, \( v_k \) does not alternate with \( y \), contradicting that \( v_ky \in E(G) \).

### 3.2 Conditions forced by a maximal split

Now, we want to investigate the opposite relation. Namely, what can one say about a representation from the structure of a maximal 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 \( \gamma \) of \( \gamma \) containing both \( x \) and \( y \).

Inspired by Naji [33], we define a symmetric relation \( \sim \) on \( A \cup B \) where \( x \sim y \) means that \( x \) and \( y \) must occur in the same subword \( \gamma_i \) of \( \gamma \). This relation is given by two conditions:
Lemma 3(c) states that if $xy \notin E(G)$, then $x \sim y$, that is, if $x$ and $y$ are placed in different subwords, then $C_x$ intersects $C_y$. In particular, $x \sim x$.

Lemma 5 gives $x \sim y$ when $x$ and $y$ are connected by a nontrivial path with all the inner vertices in $s(A) \cup s(B)$.

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 $B$. Figure 4 on right shows schematically a situation in which the relation $\sim$ has four equivalence classes $A_1$, $A_2$, $B_1$, and $B_2$.

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

**Recognition algorithms based on splits.** Split decompositions 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 in a graph in linear time [15]. In fact, the entire split decomposition tree (ie, 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 created from $G$ by contracting the vertices of $B \cup s(B)$ into a new vertex $v_A$, and $G_B$ by contracting $A \cup s(A)$ into a new vertex $v_B$. So $v_A$ is adjacent to all vertices in $A$ and to no vertices in $s(A)$, and similarly for $v_B$. Then we apply the algorithm recursively on $G_A$ and $G_B$ and construct their representations $\mathcal{R}_A$ and $\mathcal{R}_B$ (see Figure 6). It remains to join the representations $\mathcal{R}_A$ and $\mathcal{R}_B$ to construct $\mathcal{R}$.

To this end we take $\mathcal{R}_A$ and replace $C_{v_A}$ by the representation of $B \cup s(B)$ in $\mathcal{R}_B$. More precisely, let the circular ordering of the endpoints of chords defined by $\mathcal{R}_A$ be $v_A \tau_A v_A \check{\tau}_A$ and let the circular ordering defined by $\mathcal{R}_B$ be $v_B \tau_B v_B \check{\tau}_B$. The constructed $\mathcal{R}$ has the corresponding circular ordering $\tau_A \tau_B v_A \check{\tau}_A \check{\tau}_B$. 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 Figure 4 on the right cannot be constructed in this way using the split between $A$ and $B$. In what follows we describe the structure of all the representations of a circle graph $G$ based on the different circular orderings of the equivalence classes of $\sim$. While the described structure of all the representations depends
on the maximal split that we chose, the relation ~ defined with respect to this maximal split can be used to generate all the representations of G.

We choose an arbitrary circular ordering Φ₁,...,Φₚ of the classes of ~. Let Gᵢ be a graph constructed from G by contracting the vertices V(G) \ (Φᵢ ∪ s(Φᵢ)) into one vertex vᵢ, that is, Gᵢ is defined similarly to Gỵ and Gᵦ above. Let ℛᵰ,...,ℛᵦ be arbitrary representations of Gᵰ,...,Gᵦ. We join these representations as follows. Let vᵢτ₁vᵢτᵦ be the circular ordering of ℛᵰ. We construct ℛ as the circular ordering

$$\tau₁τ₂...\tauᵦ⁻¹\tauᵦ ̂\tauᵦ⁻¹...\tauᵦ.$$  

(1)

In Figure 4, we obtain the representation on the left by the circular ordering A₁A₂B₁B₂ of the classes of ~ and the representation on the right by A₁B₁A₂B₂.

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

**Lemma 7.** Every circular ordering 1 constructed as above defines a circle representation of G.

**Proof.** Let u, v ∈ V(G). We shall prove that u and v are adjacent in G if and only if they alternate in ℛ. Suppose that u, v ∈ V(Gᵢ) \ {vᵢ}. Since the cyclic subsequence τᵦ appears in both ℛᵰ and ℛ, two vertices in V(Gᵢ) \ {vᵢ} alternate in ℛ if and only if they are adjacent in Gᵢ, which is if and only if they are adjacent in G.

Otherwise, let u ∈ V(Gᵢ) \ {vᵢ} and v ∈ V(Gᵢ) \ {vᵢ} for i ≠ j. Then uv ∈ E(G) if and only if they are both long vertices. Each long vertex of Φᵢ appears once in both τᵦ and ̂τᵦ, but each short vertex s(Φᵢ) has both its occurrences either in τᵦ or in ̂τᵦ. We conclude that u and v alternate in ℛ if and only if they are both long vertices, that is, if and only if they are adjacent in G since u and v do not satisfy (C1).

Next, we analyze every representation ℛ of G.

**Lemma 8.** Let τ be the circular word corresponding to a representation ℛ of G. Then the symbols of Φᵢ ∪ s(Φᵢ) form exactly two subwords τᵦ and ̂τᵦ of τ such that for each u ∈ Φᵢ, we have u ∈ τᵦ and u ∈ ̂τᵦ, while each v ∈ s(Φᵢ) has both endpoints either in τᵦ or in ̂τᵦ.

**Proof.** Let ℛ be a representation of G and consider how it represents A U B. We get the subwords γᵢ,...,γ₂k of the endpoints of A U B, as described in Section 3.1.

Let x ∈ Φᵢ such that x ∈ γᵢ. We claim that Φᵢ is a subset of γᵢ. Since Φᵢ is an equivalence class of ~, let y ∈ Φᵢ such that one of the conditions (C1) or (C2) applies to x and y. Since ~ is the transitive closure of conditions (C1) and (C2), to prove the claim, it is sufficient to show that y ∈ γᵢ. If (C1) applies, then y ∈ γᵢ by Lemma 3(c). If (C2) applies, then y ∈ γᵢ by Lemma 5. By Lemma 3(a), each vertex of Φᵢ appears exactly once in γᵢ and once in γᵢ₊k.

Furthermore, we claim that the vertices of Φᵢ form subwords of γᵢ and γᵢ₊k. Let z ∈ γᵢ be placed between x ∈ Φᵢ and y ∈ Φᵢ. First, we assume that (C1) or (C2) applies to x and y.

- If (C1) applies to x and y, then xy ∉ E(G). As x and y do not alternate, it is not possible for z to alternate with both x and y. Thus z ~ x or z ~ y, which in turn implies that z ∈ Φᵢ.
- Suppose that (C2) applies to x and y. If x z ∉ E(G) or y z ∉ E(G), we get that z ∈ Φᵢ by (C1). Otherwise, we claim that a path P from x to y having all the internal vertices in s(Φᵢ) has at least one internal vertex adjacent to z. For every w ∈ γᵢ₊₁ and w ∈ γᵢ₊₁₊k,
we have $xw, yw, zw \in E(G)$, but none of the inner vertices of $P$ are adjacent to $w$, as otherwise $w$ would be in $\Phi_i$ and therefore in $\gamma_j$. Since $\{x, y, z, w\}$ induce the subsequence $xwywyzw$ in $\tau$, by Lemma 6 some inner vertex of $P$ has to alternate with $z$. Therefore, $z \sim x$ and $z \sim y$ by (C2), so $z \in \Phi_i$.

If $x \sim y$ and neither of (C1) and (C2) applies, we easily proceed by an inductive argument on the number of applications of (C1) and (C2). If $x \sim y \sim y$ and a vertex $z \in \gamma_j$ is placed between $x$ and $y$ in $\gamma_j$, then $z$ is also placed in $\gamma_j$ between $x$ and $y'$ or between $y'$ and $y$.

By the above argument, each class $\Phi_i$ forms two subwords of $\gamma$. Hence, it follows by Lemma 4, that by adding the short vertices $s(\Phi_i)$, we obtain two subwords of $\tau$ for each class $\Phi_i$.

Now, we are ready to prove the main structural proposition.

**Proposition 1.** Let $A$ and $B$ form a maximal split of $G$ and let $\sim$ be the equivalence relation defined by (C1) and (C2) on $A \cup B$. Then every representation $R$ of $G$ corresponds to some circular ordering $\Phi_1, \ldots, \Phi_e$ and to some representations $R_1, \ldots, R_e$ of $G_1, \ldots, G_e$. More precisely, $R$ can be constructed by arranging $R_1, \ldots, R_e$ as in $1$: $\tau_i, \tau_i, \tau_i, \ldots, \tau_i$.

**Proof.** By Lemma 7, every representation constructed by 1 is correct. Contrarily, let $R$ be a representation of $G$ with the corresponding circular word $\tau$. According to Lemma 8, we know that $\Phi_i \cup s(\Phi_i)$ forms two subwords $\tau_i$ and $\hat{\tau}_i$ of $\tau$. For $i \neq j$, the edges between $\Phi_i$ and $\Phi_j$ form a complete bipartite graph. The subwords $\tau_i, \hat{\tau}_i, \tau_j$, and $\hat{\tau}_j$ alternate, that is, appear as $\tau_i, \tau_i, \tau_j, \tau_j$ or $\tau_i, \tau_i, \tau_i, \tau_j$ in $\tau$. Thus, if we start from some point along the circle, the order of $\tau_i$'s gives a circular ordering $\Phi_1, \ldots, \Phi_e$ of the classes. The representation $R_i$ has the circular word $v_i \tau_i v_i \hat{\tau}_i$.

### 3.3 The structure of all representations of trivial maximal splits

Let $A$ and $B$ form a trivial maximal split with $A = \{a\}$ and $s(A) = \emptyset$, created from a nontrivial split. The results described in Sections 3.1 and 3.2 still apply to this split, but they are not very helpful. By Lemma 2, $a$ is an articulation in $G$. So, $G[B]$ consists of at least two connected components and $\sim$ has two equivalence classes $\Phi_1 = A$ and $\Phi_2 = B$. Since $G_2 \cong G$, Proposition 1 is not very useful as it describes all representations of $G$ in terms of all representations of $G$.

In this section, we show that all possible representations can be easily described in a different way, based on all different representations of connected components of $G \setminus a$. The following lemma states that connected components do not alternate in any circle representation.

**Lemma 9.** Let $C$ and $C'$ be two distinct connected component of a circle graph $G$. No representation has a subword $uvw$ where $u, v \in V(C)$ and $x, y \in V(C')$.

**Proof.** Let $\sigma$ be the subsequence induced by $V(C) \cup V(C')$. We know that $\sigma = \sigma_1 \ldots \sigma_{2k}$ such that each $\sigma_i$ is a maximal subword consisting only of symbols from $V(C)$ if $i$ is odd, and only of symbols from $V(C')$ if $i$ is even. We want to prove that $k = 1$. For contradiction, suppose that $k > 1$. Since $C$ is connected, there exists $u \in C$ such that $u \in \sigma_1$ and $u \in \sigma_i$ for $i > 1$. Since $C'$ is connected, there exists $x \in C'$ such that
\( x \in \sigma_2 \sigma_3 \cdots \sigma_{i-1} \) and \( x \in \sigma_{i+1} \cdots \sigma_{2k} \). Since \( u \) and \( x \) alternate, we have \( u x \in E(G) \) which is a contradiction.

We choose an arbitrary ordering of the connected components of \( G \setminus a \) as \( C_1, \ldots, C_\ell \). Let \( G_i \) be the subgraph of \( G \) induced by \( \{a\} \cup V(C_i) \). Let \( \mathcal{R}_i \) be an arbitrary representation of \( G_i \) having the circular word \( a \tilde{\tau}_i a \tilde{\tau} \). We construct the joined representation \( \mathcal{R} \) of \( G \) by the circular word

\[
\tilde{\tau}_1 \tilde{\tau}_2 \cdots \tilde{\tau}_{\ell-1} \tau_\ell a \tilde{\tau}_\ell \tilde{\tau}_{\ell-1} \cdots \tilde{\tau}_2 \tilde{\tau}_1
\]

(2)

(see Figure 7). First, we prove that every such constructed representation of \( G \) is correct.

**Lemma 10.** Every circular ordering 2 constructed as above defines a circle representation of \( G \).

**Proof.** Let \( \tau \) be the circular ordering constructed using 2. Since \( V(G_i) \) induces the subsequence \( a \tilde{\tau}_i a \tilde{\tau} \), each \( G_i \) is represented correctly in \( \mathcal{R}_i \). For \( i < j \), the vertices of \( V(C_i) \cup V(C_j) \) induce in \( \tau \) the subsequence \( \tau_i \tau_j \tilde{\tau}_j \tilde{\tau} \), so no two vertices \( u \in V(C_i) \) and \( v \in V(C_j) \) alternate and the nonedges between \( C_i \) and \( C_j \) are represented correctly. \( \square \)

Next, we analyze every representation \( \mathcal{R} \) of \( G \).

**Lemma 11.** Let \( \tau \) be the circular word corresponding to a representation \( \mathcal{R} \) of \( G \). Then the symbols of \( V(C_i) \) form exactly two subwords \( \tau_i \) and \( \tilde{\tau}_i \) of \( \tau \) such that \( a \tau_i a \tilde{\tau}_i \) is a subsequence of \( \tau \).

**Proof.** Since \( V(C_j) \cap B \neq \emptyset \), there exists some \( b \in V(C_j) \) which alternates with \( a \), so the symbols of \( V(C_j) \) form at least two subwords alternating with \( a \). If \( V(C_j) \) would form more than two subwords, then \( \tau \) has a subsequence \( au \cdots va \), where \( u, v \in V(C_j) \) and \( x \in V(C_j) \) for \( j \neq i \). Since some \( y \in V(C_j) \) alternates with \( a \), it follows that \( \tau \) has the subsequence \( au \cdots vy \), so we get \( uuvy \) which is not possible by Lemma 9. \( \square \)

Now, we are ready to prove the following structural proposition.

**Proposition 2.** Let \( A = \{a\} \) and \( B \) form a trivial maximal split of \( G \) created from a nontrivial split. Then every representation \( \mathcal{R} \) of \( G \) corresponds to some ordering \( C_1, \ldots, C_\ell \) of connected components of \( G \setminus a \) and to some representations \( \mathcal{R}_1, \ldots, \mathcal{R}_\ell \) of \( G_1, \ldots, G_\ell \) where \( G_i \) is

**FIGURE 7** If \( a \) is an articulation, then every circle representation \( \mathcal{R} \) of \( G \) consists of some ordering of connected components \( C_1, \ldots, C_\ell \) of \( G \setminus a \) and it corresponds to the depicted circular word \( \tau \) in which \( a \tau_i a \tilde{\tau}_i \) is some representation of the subgraph of \( G \) induced by \( V(C_i) \cup \{a\} \)
the subgraph of $G$ induced by $V(C_i) \cup \{a\}$. More precisely, $\mathcal{R}$ can be constructed by arranging $\mathcal{R}_1,...,\mathcal{R}_\ell$ as in 2: $a\hat{\tau}_1...\hat{\tau}_\ell a\hat{\tau}_\ell...\hat{\tau}_1$.

Proof. By Lemma 10, every representation constructed by 2 is correct. Contrarily, let $\mathcal{R}$ be a representation of $G$ corresponding to a circular word $\tau$. Suppose that $G \setminus a$ has $\ell$ connected components. The circular word $\tau$ defines an ordering $C_1,...,C_\ell$ of the connected components of $G \setminus a$ in the following way. By Lemma 11, $\tau = a\tilde{\tau}_1...\tilde{\tau}_\ell a\sigma_\ell a\sigma_{\ell-1}...\sigma_1$, where $\tilde{\tau}_i$ and precisely one $\sigma_j$ are two maximal subwords of $\tau$ containing all symbols from $V(C_i)$. Since the connected components $C_1,...,C_\ell$ cannot alternate by Lemma 9, we get that $\sigma_i$ consists of symbols of $C_i$, that is, $\sigma_i = \hat{\tau}_i$. Each $a\tau_i a\hat{\tau}_i$ gives some representation $\mathcal{R}_i$ of $G_i$. □

4 | ALGORITHM

In this section, we describe an $O(n^3)$ algorithm for the partial representation extension problem of circle graphs. It is based on the structure of all representations described in Section 3. Recall that a partial representation $\mathcal{R}'$ gives a circular word $\tau'$ such that each vertex $a \in V(G)$ appears at most twice in $\tau'$. We want to decide whether there exists a representation $\mathcal{R}$ corresponding to a circular word $\tau$ such that $\tau'$ is a subsequence of $\tau$.

Dealing with disconnected graphs. To apply the structural properties of Section 3, we need to work with connected graphs. In general, 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, eg, [26,27]). 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 disconnected inputs easily. By Lemma 9, we know that $\tau'$ cannot contain a subsequence $uvvy$ where $u, v$ belong to one 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 $\tau'$ containing all occurrences of the vertices of $C$. So from now on we assume that the input graph $G$ is connected.

Overview. Our algorithm proceeds recursively via split decomposition. For each encountered graph $G$ with a partial representation $\mathcal{R}'$ corresponding to the circular word $\tau'$, it proceeds with the following steps:

- If $G$ is prime, we have two possible representations (one is reversal of the other) and we test whether one of them extends $\tau'$. We return the result.
- Otherwise, we find a nontrivial split and modify it into a maximal split between $A$ and $B$, using Lemma 1. Next, we proceed with one of the following steps.
- In case I, the maximal split between $A$ and $B$ is nontrivial. We compute the relation $\sim$. We try to determine an ordering $\Phi_1,...,\Phi_\ell$ of the equivalence classes of $\sim$ along the circle as in 1 which is compatible with the partial representation $\mathcal{R}'$. This order is partially prescribed by predrawn endpoints of short and long vertices and we recurse on testing whether partial representations
of different equivalence classes $\Phi \cup s(\Phi)$ can be extended. If no ordering is compatible, we stop and output “no.”

- In case II, the maximal split between $A$ and $B$ is trivial with $A = \{a\}$ and $s(A) = \emptyset$. We try to determine an ordering $C_1, \ldots, C_k$ of the connected components of $G \setminus a$ along the circle as in 2 which is compatible with the partial representation $R'$. This order is partially prescribed by predrawn endpoints of chords and we recurse on testing whether partial representations of different components $C$ can be extended. If no ordering is compatible, we stop and output “no.”

For a more detailed overview of the main steps, see Algorithm 1. Now we describe everything in detail.

**Algorithm 1** The $O(n^2)$ algorithm for REPEXT(CIRCLE)

| Input: | A circle graph $G$ and a partial representation $R'$ corresponding to a circular word $\tau'$. |
| Output: | ACCEPT if $R'$ is extendable, REJECT otherwise. |

1. If $R'$ is incorrect then REJECT.
2. If $G$ is a prime graph then
   3. Construct the unique representations $\tau$ and (its reverse) $\tau_R$ of $G$.
   4. If $\tau'$ is a subsequence of $\tau$ or $\tau_R$ then ACCEPT else REJECT.
   5. Else ($G$ is not a prime graph)
      6. Find a non-trivial split between $A'$ and $B'$.
      7. Modify it into a maximal split between $A$ and $B$ such that $A' \subseteq A$ and $B' \subseteq B$.
      8. \textbf{Case I:} If the maximal split between $A$ and $B$ is non-trivial then
         9. Compute the equivalence relation $\sim$.
         10. Let $\tau' = \tau'_1 \cdots \tau'_x$ be the maximal subwords of extended classes $\Psi$.
         11. \textbf{Case I.1:} If some extended class corresponds to two maximal subwords in $\tau'$ then
             12. Compute a circular ordering $\Psi_1, \ldots, \Psi_l$ compatible with $\tau'$.
             13. Construct the partial representations $R'_i$ of $G_i$.
             14. If all $R'_i$ are extendible then ACCEPT else REJECT.
         15. \textbf{Case I.2:} Else (each extended class corresponds to one maximal subword in $\tau'$)
             16. Construct the partial representations $R'_i$ and $\bar{R}'_i$ of $G_i$.
             17. Proceed as in the subroutine of Algorithm 2.
      18. \textbf{Case II:} Else (the maximal split between $A$ and $B$ is trivial with $A = \{a\}$ and $s(A) = \emptyset$)
         19. Compute the connected components of $G \setminus a$.
         20. \textbf{Case II.1:} If both endpoints of $a$ appear in $\tau'$ then
             21. Compute a linear ordering $C_1, \ldots, C_k$ compatible with $\tau'$.
             22. Construct the partial representations $R'_i$ of $G_i$.
             23. If all $R'_i$ are extendible then ACCEPT else REJECT.
         24. \textbf{Case II.2:} Else if single endpoint of $a$ appears in $\tau'$ then
             25. Decompose the problem into two subproblems.
             26. One is solved using Case II.1, the other as in Case I.2.
             27. If both succeed then ACCEPT else REJECT.
         28. \textbf{Case II.3:} Else (no endpoint of $a$ appears in $\tau'$)
             29. \textbf{Case II.3a:} If some component has two maximal subwords in $\tau'$ then
                 30. Decompose the problem into three subproblems.
                 31. Two are solved using Case II.2, the last one using Case II.1.
                 32. If all succeed then ACCEPT else REJECT.
             33. Else (no component has two maximal subwords in $\tau'$)
                 34. Proceed as in the subroutine of Algorithm 3.

**Testing correctness of $R'$.** In the beginning, the algorithm tests correctness of the input partial representation. If $u, v \in V(G)$ have both occurrences in $\tau'$, we check that these occurrences alternate if and only if $uv \in E(G)$, and if some pair is represented incorrectly, we stop the algorithm and output “no.” If only a single endpoint of $u \in V(G)$ appears in $\tau'$, no checking is done. This checking can be done trivially in time $O(n^2)$.
Prime 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 $R$ and $\hat{R}$ [15] where one is the reversal of the other. We just need to test whether one of them extends $R'$. We can construct one of these representations in quasi-linear time [23].

Finding a maximal split between $A$ and $B$. If the graph $G$ is not prime, then we can find a nontrivial split between $A'$ and $B'$ in linear time [15]. Using Lemma 1, we modify it into a maximal split between $A$ and $B$ such that $A' \subseteq A$ and $B' \subseteq B$ in linear time.

4.1 Case I: a nontrivial maximal split between $A$ and $B$

We start by computing the equivalence relation $\sim$ which can be done in time $O(n^2)$. Next, we want to find an ordering of its equivalence classes. For a class $\Phi$ of $\sim$, we define the extended class $\Psi$ of $\sim$ as $\Phi \cup s(\Phi)$. If some extended class has no vertex predrawn, we may choose an arbitrary representation and place it in an arbitrary order, so we can ignore such classes for the rest of case I. Let $\sim$ have $\ell$ equivalence classes, all of them appearing in $\tau'$.

The circular word $\tau'$ is composed of $k$ maximal subwords $\tau' = \tau'_1 \tau'_2 \cdots \tau'_k$ such that each $\tau'_i$ contains only symbols of one extended class $\Psi$. According to Proposition 1, each extended class $\Psi$ corresponds to at most two different maximal subwords. Also, if two extended classes $\Psi$ and $\hat{\Psi}$ each correspond to two different maximal subwords, then occurrences of these subwords alternate in $\tau'$. Otherwise we reject the input.

Case I.1: an extended class corresponds to two maximal subwords. We denote this class by $\Psi_1$ and put this class as first in the ordering. By renumbering, we may assume that $\Psi_1$ corresponds to $\tau'_1$ and $\tau'_2$. Then one circular order of the classes can be determined by the following linear ordering $< \text{starting with } \Psi_1$. Let $\Psi_i$ and $\Psi_j$ be two distinct classes. If $\Psi_i$ corresponds to $\tau'_a$ and $\Psi_j$ corresponds to $\tau'_b$ such that either $a < b < t$ or $t < a < b$, we put $\Psi_i < \Psi_j$. We obtain the ordering of the classes as any linear extension of $<$. Since subwords of all extended classes with two subwords in $\tau'$ alternate, we get that $<$ is acyclic and a linear extension always exists. Figure 8A shows an example.

We have ordered the extended classes $\Psi_1, \ldots, \Psi_\ell$ and the corresponding classes $\Phi_1, \ldots, \Phi_\ell$. We construct each $G_i$ with the vertices $\Psi_i \cup \{v_i\}$ as in Section 3.2, so $v_i$ is adjacent to $\Phi_i$ and nonadjacent to $s(\Phi_i)$. The partial representation $R'_i$ of $G_i$ is either the word $v_i \tau'_s v_i$ (if $\Psi_i$ corresponds to the single maximal subword $\tau'_i$ in $\tau'$) or the word $v_i \tau'_s v_i \tau'_t$ (if $\Psi_i$ corresponds to two maximal subwords $\tau'_s$ and $\tau'_t$ in $\tau'$). We test recursively, whether each representation $R'_i$ of $G_i$ is extendible to a representation of $R_i$. If yes, we join $R_1, \ldots, R_\ell$ as in Proposition 1. Otherwise, the algorithm outputs “no.”

Lemma 12. In case I.1, the representation $R'$ is extendible if and only if the representations $R'_1, \ldots, R'_\ell$ of the graphs $G_1, \ldots, G_\ell$ are extendible.

Proof. Suppose that $R$ extends $R'$. According to Proposition 1, the representations of $\Psi_1, \ldots, \Psi_\ell$ are ordered along the circle, and so we obtain representations $R_1, \ldots, R_\ell$ extending $R'_1, \ldots, R'_\ell$. For the other implication, we just take $R_1, \ldots, R_\ell$ and combine them to form $R$ as in 1. This works since the ordering $<$ was constructed so that $R$ extends $R'$.
Case I.2: no extended class corresponds to two maximal subwords. We number the classes according to their appearance in $\tau'$, that is, $\Psi_i$ corresponds to the subword $\tau_i'$. By Proposition 1, we know that in any representation $\mathcal{R}$ of $G$ the class $\Psi_i$ corresponds to two subwords $\tau_i$ and $\hat{\tau}_i$. The difficulty here arises from the potential for $\tau_i'$ to be a subsequence of $\tau_i\hat{\tau}_i$, but neither $\tau_i$ nor $\hat{\tau}_i$.

Figure 8B shows two potential extending representations.

We solve this as follows: instead of constructing just one partial representation $\mathcal{R}'_i$ of $G_i$ corresponding to the circular word $\tau_i v_i \hat{\tau}_i$, we construct an additional partial representation $\check{\mathcal{R}}_i'$ corresponding to the circular word $\tau_i v_i$, that is, $v_i$ has only one endpoint predrawn. Figure 9 shows that $\check{\mathcal{R}}_i'$ is less restrictive: if $\mathcal{R}'_i$ is extendible, then $\check{\mathcal{R}}_i'$ is also extendible, but it might not be the other way. For instance, every long chord in $\Phi_i$ alternates with $v_i$, so if some long chord has both endpoints predrawn in $\tau_i'$, $\mathcal{R}'_i$ is necessarily nonextendible, but $\check{\mathcal{R}}_i'$ might be extendible.

The following lemma is the main trick of the algorithm and is essential to prove that it has cubic running time. It states that, if $\tau'$ is extendible, at most one class can be forced to use $\check{\mathcal{R}}_i'$.

**Lemma 13.** In case I.2, the representation $\mathcal{R}'$ is extendible if and only if $\check{\mathcal{R}}_i'$ is extendible for some $i$ and $\mathcal{R}'_j$ is extendible for all $j \neq i$.

**Proof.** When $\mathcal{R}_j$ corresponding to a word $v_j \tau_j v_j \hat{\tau}_j$ is an extension of $\mathcal{R}'_j$ for $j \neq i$, then $\tau_j'$ is a subsequence of, say, $\tau_j$. Contrarily, when $\mathcal{R}_i$ corresponding to a word $v_i \tau_i v_i \hat{\tau}_i$ is an
extension of \( \hat{\mathcal{R}} \), then \( \tau'_i \) is a subsequence of \( \tau \hat{i} \), but might not be of \( \tau_i \) or \( \hat{i} \). We use the circular ordering \( \Psi_i, \Psi_j \) of the classes and we construct the representation \( \mathcal{R} \) as in 1:

\[
\tau_{i+1} \cdots \tau_i \hat{i} \cdots \tau_i \hat{i} \tau_{i+1} \cdots \hat{i} \hat{i}_1 \cdots \hat{i}_{i-1} \hat{i},
\]

where all predrawn endpoints of \( \tau' \) appear in those words written in bold. It is easy to see that \( \mathcal{R} \) extends \( \mathcal{R}' \) since \( \tau' \) has no predrawn endpoints in \( \hat{i} \cdots \hat{i} \hat{i}_1 \cdots \hat{i}_{i-1} \).

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 1, the representation \( \mathcal{R} \) gives a representation \( \mathcal{R}_i \) corresponding to \( \tau \tau \hat{i} \tau \hat{i} \) of \( G_i \) and \( \mathcal{R}_j \) corresponding to \( \tau \tau \tau \tau \hat{i} \) of \( G_j \). Since \( \mathcal{R}_i \) and \( \mathcal{R}_j \) are nonextendible, we have that \( \tau'_i \) is neither a subsequence of \( \tau_i \), nor \( \hat{i} \), and similarly \( \tau'_j \) is neither of \( \tau_j \), nor \( \hat{i} \). Therefore, either \( \tau \tau \tau \tau \hat{i} \), or \( \tau \tau \tau \tau \hat{i} \) is a subsequence of \( \tau \), and we get that two maximal subwords in \( \tau' \) correspond to both \( \Psi_i \) and \( \Psi_j \) which is a contradiction. \( \square \)

---

**Algorithm 2** The subroutine for case I.2

1. Test whether each of \( \mathcal{R}_{2}, \ldots, \mathcal{R}_{\ell} \) is extendible.
2. If two of \( \mathcal{R}_{2}, \ldots, \mathcal{R}_{\ell} \) are not extendible then **REJECT**.
3. If exactly one of \( \mathcal{R}_{2}, \ldots, \mathcal{R}_{\ell} \), denoted by \( \mathcal{R}_i \), is not extendible then
4. If \( \hat{\mathcal{R}}_i \) and \( \mathcal{R}_i \) are extendible then **ACCEPT**, else **REJECT**.
5. Else (all of \( \mathcal{R}_{2}, \ldots, \mathcal{R}_{\ell} \) are extendible)
6. If \( \hat{\mathcal{R}}_i \) is extendible then **ACCEPT**, else **REJECT**.

---

Let \( n = |V(G)| \) and let \( \Psi_1 \) be the largest class, so \( |\Psi_i| \leq n/2 \) for \( i > 1 \). If we want to recursively test for each \( \Psi_i \) whether both \( \mathcal{R}_i \) and \( \hat{\mathcal{R}}_i \) are extendible, the running time might be exponential since we might have \( |\Psi_i| \approx n \). Fortunately, using Lemma 13, it is sufficient to test only one of \( \mathcal{R}_i \) and \( \hat{\mathcal{R}}_i \). We recursively test whether \( \mathcal{R}_{2}, \ldots, \mathcal{R}_{\ell} \) are extendible; see the pseudocode of Algorithm 2:

- **Two or more of** \( \mathcal{R}_{2}, \ldots, \mathcal{R}_{\ell} \) **are not extendible**. By Lemma 13, \( \mathcal{R}' \) is nonextendible, the algorithm stops and outputs “no.”
- **Exactly one of** \( \mathcal{R}_{2}, \ldots, \mathcal{R}_{\ell} \) **is not extendible**. Let \( \mathcal{R}_i \) be the nonextendible representation. We test whether \( \hat{\mathcal{R}}_i \) and \( \mathcal{R}_i \) are extendible. If at least one is nonextendible, the algorithm stops and outputs “no.” If both are extendible, we similarly join in \( \mathcal{R} \) the representations \( \mathcal{R}_{2}, \ldots, \mathcal{R}_{\ell} \) according to 1 as described in the proof of Lemma 13.
- **All representations** \( \mathcal{R}_{2}, \ldots, \mathcal{R}_{\ell} \) **are extendible**. We have representations \( \mathcal{R}_{2}, \ldots, \mathcal{R}_{\ell} \) where \( \mathcal{R}_i \) extends \( \mathcal{R}_i \). We test whether the partial representation \( \hat{\mathcal{R}}_i \) is extendible. If not, the algorithm stops and outputs “no.” If it extends, we get a representation \( \mathcal{R}_i \) of \( G \). We construct the representation \( \mathcal{R} \) using 1 as described in the proof of Lemma 13.
Lemma 14. In case I.2, the representation \( R' \) is extendible if and only if the algorithm constructs it.

Proof. We know that \( \hat{R}' \) is extendible when \( R' \) is extendible. Lemma 13 states that \( R' \) is extendible if and only if at most one of \( R_i' \) is nonextendible while \( \hat{R}' \) is extendible. The algorithm tests this in case I.2, while postponing \( \Psi_i \) until it knows which of \( R_i' \) and \( \hat{R}' \) needs to be tested.

4.2 Case II: a trivial maximal split between \( A \) and \( B \)

Let \( A = \{a\} \) and \( s(A) = \varnothing \). In Section 3.3 we characterized all possible representations \( R \) in terms of representations of connected components \( C_i \) of \( G \setminus a \). We just need to test whether one of them is compatible with the partial representation \( R' \) corresponding to the circular word \( \tau' \).

Similarly as in Section 4.1, we may assume that every connected component \( C \) has at least one endpoint in \( \tau' \); otherwise, we can deal with it trivially.

Case II.1: both endpoints of \( a \) appear in \( \tau' \). The circular word \( \tau' \) is composed of \( k \) and \( k' \) maximal subwords \( \tau' = a\hat{\tau}'_2 \cdots \hat{\tau}'_k \hat{\tau}'_k a\hat{\tau}'_k' \cdots \hat{\tau}'_1' \) such that each \( \hat{\tau}'_i \) contains only symbols of one connected component \( C_i \) and similarly for each \( \hat{\tau}'_j \). According to Proposition 2, each connected component \( C_i \) corresponds to at most two different maximal subwords. If a connected component \( C_i \) corresponds to two subwords \( \hat{\tau}'_i \) and \( \hat{\tau}'_j \), then \( a\hat{\tau}'_i a\hat{\tau}'_j \) is a subsequence of \( \tau' \). Also, if two components \( C_i \) and \( \hat{C}_j \) each correspond to two different maximal subwords, then occurrences of these subwords do not alternate in \( \tau' \). Otherwise we reject the input.

Next, we find a linear ordering of \( \ell \) connected components as follows. We order \( C < C' \) if \( C \) corresponds to a subword \( \hat{\tau}'_s \) and \( C' \) to a subword \( \hat{\tau}'_t \) for \( s < t \), or \( C \) to \( \hat{\tau}'_s \) and \( C' \) to \( \hat{\tau}'_t \) for \( s < t \). We obtain a linear ordering \( C_1, \ldots, C_\ell \) as any linear extension. Since subwords of all connected components with two subwords in \( \tau' \) do not alternate, we get that \( < \) is acyclic and a linear extension always exists. Suppose that we renumber the maximal subwords of \( \tau' \) in such a way that \( C_i \) corresponds to \( \hat{\tau}'_i \) and \( \hat{\tau}'_i \) (one of them possibly empty). Let \( G_i \) be the subgraph of \( G \) induced by \( V(C_i) \cup \{a\} \). Let \( R_i' \) be the partial representation of \( G_i \) corresponding to the circular word \( a\hat{\tau}'_1 a\hat{\tau}'_1 \), so \( \tau' = a\hat{\tau}'_1 \cdots a\hat{\tau}'_1 \). Figure 10A shows an example.

Lemma 15. In case II.1, the representation \( R' \) is extendible if and only if the representations \( R_i', \ldots, R_\ell' \) of the graphs \( G_1, \ldots, G_\ell \) are extendible.

Proof. Suppose that \( R \) extends \( R' \). According to Proposition 2, the representations of \( C_1, \ldots, C_\ell \) are ordered along the circle, and so we obtain representations \( R_i, \ldots, R_\ell \) extending \( R_i, \ldots, R_\ell \).

For the other implication, we just take \( R_i, \ldots, R_\ell \) and combine them to form \( R \) as in 2. This works since the ordering \( < \) was constructed so that \( R \) extends \( R' \).

Case II.2: a single endpoint of \( a \) appears in \( \tau' \). The circular word \( \tau' \) is composed of \( k \) maximal subwords \( \tau' = a\hat{\tau}'_2 \cdots \hat{\tau}'_k \) such that each \( \hat{\tau}'_i \) contains only symbols of one connected component \( C_i \). According to Proposition 2, each connected component \( C_i \) corresponds to at most two different maximal subwords. Also, if two components \( C_i \) and \( \hat{C}_j \) each correspond to two
different maximal subwords, then occurrences of these subwords do not alternate in $\tau'$. Otherwise we reject the input.

Suppose there is a component $C$ corresponding to two maximal subwords $\tau'_s$ and $\tau'_t$ for $s < t$. Further let $C$ be such a component that maximizes the value $s$. In every extending representation, we have the subsequence $a\tau'_s a\tau'_t$, so we can assume that the second endpoint of $a$ is predrawn in between $\tau'_s$ and $\tau'_t$. We divide testing whether $R'$ is extendible into two subproblems. We deal with the connected components of the circular word $a\tau'_1\tau'_2\cdots\tau'_s a\tau'_t\tau'_{t+1}\cdots\tau'_k$ exactly as in case II.1. It remains to decide whether $a\tau'_{s+1}\cdots\tau'_{t-1}$ is extendible where each connected component corresponds to precisely one maximal subword (note that when no such component $C$ exists, we have precisely this situation). Figure 10B shows an example.

Suppose that we rename $\tau' = a\tau'_1\cdots\tau'_k$ such that $\tau'_1$ corresponds to the connected component $C_i$. Similar to case I.2, the difficulty comes from the fact that some $\tau'_j$ might be a subsequence of $\tau\tau\tau\tau\hat{t}$ of 2 in an extending representation, but not of $\tau$ or $\hat{t}$. We consider two partial representations for each $G_i$: the partial representation $R'_i$ corresponding to $a\tau'_i a$ and $\hat{R}'_i$ corresponding to $a\tau'_i$. Again, if $R'_i$ is extendible, then $\hat{R}'_i$ is also extendible.

**Lemma 16.** In case II.2 with no connected component corresponding to two maximal subwords of $\tau'$, the representation $R'$ is extendible if and only if $\hat{R}'_i$ is extendible for some $i$ and $R'_j$ is extendible for all $j \neq i$.

**Proof.** When $R'_j$ corresponding to a word $a\tau'_j a\hat{t}_j$ is an extension of $R'_j$, for $j \neq i$, then $\tau'_j$ is a subsequence of, say, $\tau_j$ for $j < i$ and of $\hat{t}_j$ for $j > i$. Contrarily, when $R'_i$ corresponding to a word $a\tau'_i a\hat{t}_i$ is an extension of $\hat{R}'_i$, then $\tau'_i$ is a subsequence of $\tau\tau\tau\tau\hat{t}$, but might not be of $\tau$ or $\hat{t}$. We use the linear ordering $C_1, \ldots, C_{i-1}, C_i, C_{i-1}, \ldots, C_1$ of the connected components and we construct the representation $R$ as in 2:

$$a\tau_1\cdots\tau_{i-1}\tau_{i+1}\tau_i a\tau_{i+1} \cdots \hat{t}_i \hat{t}_{i-1} \cdots \hat{t}_i.$$
where all predrawn endpoints of $\tau'$ appear in those words written in bold. It is easy to see that $R$ extends $R'$ since there are no predrawn endpoints in $\tau_i', \ldots, \tau_{i+1}$ and in $\hat{\tau}_{i-1}, \ldots, \hat{\tau}_i$.

For the other implication, suppose that $R$ corresponding to $\tau$ extends $R'$, and we add into $\tau'$ the position of the other endpoint of $a$. It splits at most one maximal word $\tau'_i$, so for every $j, j \neq i$, $a\tau'_ja$ is a subsequence of $\tau$ and $R'_j$ is extendible. Since $a\tau'_i$ is a subsequence of $\tau$, we get that $\hat{R}_i$ is extendible.

The rest of this case proceeds exactly as case I.2.

**Lemma 17.** In case II.2, the representation $R'$ is extendible if and only if the algorithm constructs it.

**Proof:** The proof is similar to Lemma 14.

**Case II.3:** no endpoint of $a$ appears in $\tau'$. As in case II.2, the circular word $\tau'$ is composed of $k$ maximal subwords $\tau' = \tau_1', \ldots, \tau_k'$. If two components $C$ and $\hat{C}$ each correspond to two different maximal subwords, then occurrences of these subwords do not alternate in $\tau'$. Otherwise we reject the input. Also, if some connected component $C$ corresponds to two subwords $\tau'_i$ and $\tau'_j$, then $a\tau'_ja$ is a subsequence of every extending representation. Therefore, existence of such a component restricts the possible positions of endpoints of $a$, so we divide this case into two subcases.

**Case II.3a:** some component has two maximal subwords in $\tau'$. By a suitable renaming of the subwords, let $C$ be the connected component corresponding to $\tau'_p$ and $\tau'_q$ such that $p < q$, $p$ is minimal, and $\tau'_{p+1}, \ldots, \tau'_p, \tau'_1, \ldots, \tau'_{p-1}$ correspond to connected components having only one maximal subword in $\tau'$. Similarly, let $C'$ be the connected component corresponding to $\tau'_s$ and $\tau'_t$ such that $s < t$ and all $\tau'_{s+1}, \ldots, \tau'_{t-1}$ correspond to connected components having only one maximal subword in $\tau'$, and possibly $C = C'$. If $R'$ is extendible, we get that every connected component corresponding to two maximal subwords $\tau'_x$ and $\tau'_y$ has $p \leq x \leq t \leq y \leq q$; otherwise we reject the input. Figure 11 shows an example.

It follows that every extending representation has $a\tau'_p, \tau'_{p+1}, \ldots, \tau'_s a\tau'_i, \tau'_{i+1}, \ldots, \tau'_q$ as a subsequence. Similarly as case II.2, we can divide testing whether $R'$ is extendible into three subproblems:

- Testing using case II.2 whether the partial representation $\tau'_{q+1}, \ldots, \tau'_s a\tau'_i, \tau'_{i+1}, \ldots, \tau'_p$ is extendible.
- Testing using case II.1 whether the partial representation $a\tau'_p, \ldots, \tau'_s a\tau'_i, \tau'_q$ is extendible.
- Testing using case II.2 whether the partial representation $a\tau'_{s+1}, \ldots, \tau'_{t-1}$ is extendible.

**Lemma 18.** In case II.3a, the representation $R'$ is extendible if and only if the algorithm constructs it.

**Proof:** This is implied by Lemmas 15 and 17.

**Case II.3b:** no component has two maximal subwords in $\tau'$. Let $\tau'_i$ correspond to the connected component $C_i$, and define $R'_i$ and $\hat{R}_i$ exactly as in case II.2. Similarly to case II.2, the difficulty comes from the fact that some $\tau'_i$ might correspond to both $\tau_i$ and $\hat{\tau}_i$ of 2 in an
Lemma 19. The representation $R'$ is extendible if and only if $R'_i$ and $R'_j$ are extendible for some $i$ and $j$, and $R'_k$ is extendible for all $k \neq i, j$.

Proof: Let $i < j$. When $R_k$ corresponding to a word $a_t \alpha_k$ is an extension of $R'_k$ for $k \neq i, j$, then $\tau'_k$ is a subsequence of, say, $\tau_k$ when $i < k < j$, and of $\tilde{\tau}_k$ when $k < i$ or $k > j$. Contrarily, when $R_i$ corresponding to a word $a_t \alpha_i$ is an extension of $R'_i$, then $\tau'_i$ is a subsequence of $\tau_i \tilde{\tau}$, but might not be of $\tau_i$ or $\tilde{\tau}$, and similarly for $R_j$. We use the linear ordering $C_1, C_{i-1}, \ldots, C_1, C_{i+1}, \ldots, C_{j-1}, C_{j+1}, C_{j}$ of the connected components and we construct the representation $R$ as in 2:

$$a \tau_i \tilde{\tau}_{i-1} \cdots \tilde{\tau}_i \tau_{i+1} \cdots \tau_{j-1} \tau_{j+1} + 1 \tau_j a \tilde{\tau}_j \cdots \tilde{\tau}_{j-1} \tilde{\tau}_{j+1} \tau_i \cdots \tau_{i-1} \tilde{\tau}_i,$$

where all predrawn endpoints of $\tau'$ appear in those words written in bold. It is easy to see that $R$ extends $R'$ since there are no predrawn endpoints in $\tilde{\tau}_{i-1} \cdots \tilde{\tau}_i$, in $\tau_{j+1} \cdots \tau_{j-1}$, and in $\tilde{\tau}_{j-1} \cdots \tilde{\tau}_{j+1}$.

For the other implication, suppose that $R$ corresponding to $\tau$ extends $R'$, and we add into $\tau'$ the positions of the endpoints of $a$. It is not possible that both endpoints split the same maximal word $\tau'_i$, otherwise the remaining components $C_k$ would alternate with $C_i$, as they alternate with $a$. It is additionally not possible that two maximal words are split by the same endpoint. So at most two maximal words $\tau'_i$ and $\tau'_j$ are split by the endpoints of $a$. Therefore, for every $k \neq i, j$, we have $a \tau'_k a$ as a subsequence of $\tau$, so $R'_k$ is extendible. Since $a \tau'_i$ and $a \tau'_j$ are subsequences of $\tau$, we get that $R'_i$ and $R'_j$ are also extendible.

Let $n = |V(G)|$ and let $C_i$ be the largest component, so $|V(C_i)| \leq n/2$ for $i > 1$. The algorithm works similarly to case I.2; see Algorithm 3 for a pseudocode. So we test the extendibility of only one of $R'_i$ and $R'_j$ while testing both types of representations for at most two other graphs $G_i$ and $G_j$. 

**FIGURE 11** An example of case II.3a. On the left, we have two connected components corresponding to two maximal subwords $\tau'_3$ and $\tau'_{13}$, and $\tau'_5$ and $\tau'_{10}$. We put $p = 3$, $q = 13$, $s = 5$, and $t = 10$. We divide testing whether $R'$ is extendible into three subproblems depicted on the right.
Algorithm 3: The subroutine for case II.3b

1. Test whether each of \( R_{2}, \ldots, R_{\ell} \) is extendible.
2. If three of \( R_{2}, \ldots, R_{\ell} \) are not extendible then REJECT.
3. If exactly two of \( R_{2}, \ldots, R_{\ell} \), denoted \( R_{i} \) and \( R_{j} \), are not extendible then
4. If \( R_{i} \) and \( R_{j} \) are extendible then ACCEPT else REJECT.
5. If exactly one of \( R_{2}, \ldots, R_{\ell} \), denoted \( R_{i} \), is not extendible then
6. If \( R_{i} \) and \( R_{1} \) are extendible then ACCEPT else REJECT.
7. Else (all of \( R_{2}, \ldots, R_{\ell} \) are extendible)
8. If \( R_{1} \) is extendible then ACCEPT else REJECT.

Lemma 20. In case II.3b, the representation \( R' \) is extendible if and only if the algorithm constructs it.

Proof. We use Lemma 19 similarly as in the proof of Lemmas 14 and 17. □

4.3 Analysis of the algorithm

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

Lemma 21. The described algorithm correctly decides whether the partial representation \( R' \) of \( G \) is extendible.

Proof. If the input graph \( G \) is prime, we just test both representations whether they extend \( \tau' \). If the input graph \( G \) contains a nontrivial split, we modify it into a maximal split between \( A \) and \( B \) using Lemma 1. Next, we proceed by case I or case II, depending whether the maximal split is trivial or not. For case I, the algorithm is correct by Lemmas 12 and 14. For case II, the algorithm is correct by Lemmas 15, 17, 18, and 20. □

Lemma 22. The running time of the algorithm is \( O(n^3) \) where \( n \) is the number of vertices.

Proof. Let \( T(n) \) be the worst-case time complexity of the algorithm if the input graph has \( n \) vertices. We show that \( T(n) = O(n^3) \) by establishing and solving the following recurrence:

\[
T(n) \leq 2T(n_2 + 1) + \sum_{j=1}^{\ell} T(n_j + 1) + O(n^2),
\]

where \( \sum_{j=1}^{\ell} n_j = n \), and \( n - 2 \geq n_1 \geq n_2 \geq \cdots \geq n_\ell \).

First, we establish 3 in the case of prime graphs, where only the \( O(n^2) \)-term comes into play. As described, we can test whether the graph \( G \) is prime and construct a unique representation \( \tau \) in quasi-linear time using [23], but for the purpose of our analysis \( O(n^2) \) is sufficient (namely, we could instead use the slightly older \( O(n^2) \) algorithm [39] here).
Since each symbol appears twice in $\tau$, we can easily test in linear time whether $\tau'$ is a subsequence of $\tau$ or its reversal. Second, if $G$ is not prime, we find a nontrivial split between $A'$ and $B'$ using [15] and modify it using Lemma 1 into a maximal split between $A$ and $B$ such that $A' \subseteq A$ and $B' \subseteq B$. Both can be achieved in linear time.

**Case I.** We compute the $\sim$ relation in time $O(n^2)$.
- In case I.1, we divide the problem into $\ell$ smaller disjoint subproblems each of size $n_i + 1$, where $\sum_i n_i = n$ and $n_i \leq n - 2$ for every $n_i$. Indeed, a split between $A$ and $B$ that we consider in this case is nontrivial. Hence, this step clearly satisfies 3.
- In case I.2, we test both representations $\mathcal{R}_i'$ and $\bar{\mathcal{R}}_i'$ for at most one extended class of size at most $|\Psi| \leq n/2$, while we test exactly one of these representations of all remaining extended classes. We get the following recursion:

$$T(n) \leq T(|\Psi| + 1) + \sum_j T(|\Psi| + 1) + O(n^2),$$

where the right-hand side is clearly dominated by the right-hand side of 3 by the nontriviality of the split between $A$ and $B$.

**Case II.** We find connected components of $G \setminus a$ in linear time. We recall that $a$ is an articulation by Lemma 2.
- In case II.1, the analysis is similar as in case I.1.
- In case II.2, we divide the input into two disjoint subproblems, one is solved as in case II.1, the other as in case I.2. Thus, 3 is established also in this case.
- In case II.3a, we divide the input into three disjoint subproblems solved using cases II.1 and II.2, which establishes 3.
- In case II.3b, we test both representations $\mathcal{R}_i'$ and $\bar{\mathcal{R}}_i'$ for at most two extended classes of sizes $|\Psi|, |\Psi| \leq n/2$, while we test exactly one of the representations of all remaining extended classes. We get the following recursion:

$$T(n) \leq T(|\Psi| + 1) + T(|\Psi| + 1) + \sum_j T(|\Psi| + 1) + O(n^2),$$

where the right-hand side corresponds to the right-hand side of 3.

It remains to show that 3 implies the claimed cubic running time $cn^3$, for some sufficiently large fixed $c > 0$. We proceed by induction on $n$, where the base case, in which $n \leq 1$, is trivial. In the inductive step we distinguish two cases $n_i \geq n/2$ and $n_i < n/2$.

In both cases, the convexity of the function $f(x) = cx^3$ over positive reals allows us to consider only the situations in which the number of subproblems is as small as possible.

Thus, in the case when $n_i \geq n/2$, we get by applying the induction hypothesis that

$$T(n) \leq cn_i^3 + 3c(n - n_i)^3 + 3cn^2 + cn^2,$$

for some fixed $c' > 0$, where the $c'n^2$-term comes from $O(n^2)$, and the $3cn^2$-term is obtained by upper bounding the lower order terms coming from $2c(n_2 + 1)^3 + \sum_i c(n_i + 1)^3$. By
derivating according to $x$ the function $x^3 + 3c(n - x)^3$, we obtain that the right-hand side is upper bounded by

$$\max \left\{ c(n - 2)^3 + 24c + 3cn^2 + c'n^2, \frac{1}{2}cn^3 + 3cn^2 + c'n^2 \right\},$$

where the two values over which we maximize are obtained for the extreme values of $n_1$, namely, $n/2$ and $n - 2$. By taking $c$ sufficiently large in terms of $c'$ we can upper bound the right-hand side by $cn^3$, for every $n \geq 2$, and we are done in this case.

The case when $n_1 < n/2$ is handled analogously. We apply the induction hypothesis and consider the number of subproblems as small as possible, which is 3. We can assume that $n_1 = \lceil (n/2) - 1 \rceil$, $n_2 = \lceil (n/2) - 1 \rceil$ and $n_3 = 2$. Indeed, otherwise by putting $n_1 := n_1$, $n_2 := n_2 + \varepsilon$, $n_3 := n_3 - \varepsilon$, for some small $\varepsilon > 0$, the upper bound on $T(n)$ can only increase in the following expression:

$$T(n) \leq 3c \left[ \frac{n}{2} \right]^3 + 2c3^3 + c'n^2.$$

Thus, we are done by the same token as in the previous case, which concludes the proof. □

The proof of the main result in this paper now follows easily.

**Proof of Theorem 1.** The result is implied by Lemmas 21 and 22. □

### 5 | SIMULTANEOUS REPRESENTATIONS OF CIRCLE GRAPHS

In this section, we give two results concerning the simultaneous representation problem for 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:

**Problem:** Simultaneous representation for circle graphs—\textsc{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 $R_1, \ldots, R_k$ of $G_1, \ldots, G_k$ which use the same representation of the vertices of $I$?

**Proof of Theorem 2.** To show that \textsc{Sim(circle)} is NP-complete, we reduce it from the total ordering problem:

**Problem:** The total ordering problem—\textsc{TotalOrdering}

**Input:** A finite set $S$ and a finite set $T$ of triples from $S$.

**Output:** Does there exist a total ordering $< \circ S$ such that for all $(x, y, z) \in T$ either $x < y < z$, or $z < y < x$?

Opatrny [34] proved this problem is NP-complete.

Given an instance $(S, T)$ of \textsc{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 \textsc{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 \( \in \text{xyz} \). 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 Figure 12 for an example of this construction.

The clique in \( G_0 \) defines a split where each class of \( ~ \) is a singleton. According to Proposition 1, every representation \( p_0 \) of \( G_0 \) places the elements of \( I \) in some circular ordering \( \cdots w \) which corresponds to the total ordering \( \cdots s \). Now the representations \( R_0, \ldots, R_t \) of \( G_0, \ldots, G_t \) can be constructed if and only if all the total ordering constraints are satisfied. This implies that there exists a solution \( R_0, \ldots, R_t \) of \( G_0, \ldots, G_t \) if and only if the instance \( (S, T) \) of \text{TOTALORDERING} is solvable.

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

**Proof of Corollary.** 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 \( R' \). We just solve \( k \) instances of \text{RepExt(CIRCLE)}, one for each \( G_i \), and the partial representation \( R' \) of \( I \), which can be done in polynomial time according to Theorem 1.

\[ \square \]

6 | CONCLUSIONS

The structural results described in Section 3, namely, Propositions 1 and 2, 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 1 also seems to be useful in attacking the following open problems:

---

**FIGURE 12** 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 \text{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 \( 5 < 1 < 2 < 4 < 3 \).
Question 1. 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 for some chords two circular arcs and we want to construct a representation which places endpoints into these circular arcs.

Question 2. What is the complexity of the bounded representation problem for circle graphs?

**Permutation graphs.** Permutation graphs are intersection graphs of segments between two parallel lines. So every permutation representation of \( G \) consists of two words \( \tau \) and \( \hat{\tau} \), each containing each vertex \( V(G) \) exactly once, and \( uv \in E(G) \) if and only if their order in \( \tau \) and \( \hat{\tau} \) differs. We denote the class by \( \text{PERM} \).

Let \( \hat{\tau}_R \) be the reversal of \( \hat{\tau} \). Since \( \tau\hat{\tau}_R \) is a circle representation of \( G \), it follows that every permutation graph is a circle graph. More strongly, a graph \( G \) is a permutation graph if and only if \( G \) constructed from \( G \) by adding a universal vertex \( u \) is a circle graph, since \( u\tau\hat{\tau}_R \) is a circle representation of \( \tilde{G} \).

The partial representation problem for permutation graphs is studied in [25] and solved in time \( O(n^3) \). The following result gives an alternative algorithm running in time \( O(n^3) \) as well.

**Proposition 3.** The problem \( \text{REPEXT(\text{PERM})} \) reduces in time \( O(n + m) \) to \( \text{REPEXT(CIRCLE)} \).

**Proof.** Let \( G \) be a permutation graph with a partial representation \( \mathcal{R}' \) corresponding to two words \( \tau' \) and \( \hat{\tau}' \). The problem \( \text{REPEXT(\text{PERM})} \) asks whether there exists words \( \tau \) and \( \hat{\tau} \) representing \( \mathcal{R} \) such that \( \tau' \) and \( \hat{\tau}' \) are subsequences of \( \tau \) and \( \hat{\tau} \), respectively. The reduction constructs the circle graph \( \tilde{G} \) by adding a universal vertex \( u \) to \( G \) and the partial representation \( \tilde{\mathcal{R}}' \) given by the circular word \( u\tau\hat{\tau}_R \). The reduction clearly works in linear time. It is correct since the partial representation \( \mathcal{R}' \) of \( G \) is extendible if and only if \( \tilde{\mathcal{R}}' \) of \( \tilde{G} \) is extendible. □

**Minimal split decomposition and split trees.** A split decomposition of \( G \) works as follows. Consider a split between \( A \) and \( B \). We replace \( G \) by the graphs \( G_A \) and \( G_B \) defined in Section 3.2. Then we apply the decomposition recursively on \( G_A \) and \( G_B \), and we stop on prime graphs containing no splits. We note that by different orders of splits, different decompositions of \( G \) may be constructed. A split decomposition can be computed in linear time [15].

A split decomposition is called **minimal** if it is constructed by the least number of splits. Suppose that we also stop on **degenerate graphs** which are complete graphs \( K_n \) and stars \( S_n = K_{1,n} \). Cunningham [14] proved that the minimal split decomposition of a connected graph stopping on prime and degenerate graphs is unique.

The unique **split tree** \( S \) representing a graph \( G \) encodes the minimal split decomposition [23]. A split tree is a graph with two types of vertices (normal and marker vertices) and two types of edges (normal and tree edges). We initially put \( S = G \) and modify it according to the minimal split decomposition. If the minimal decomposition contains a split between \( A \) and \( B \) in \( G \), then we replace \( G \) in \( S \) by the graphs \( G_A \) and \( G_B \), and connect the marker vertices \( m_A \) and \( m_B \) by a **tree edge** (see Figure 13A). We repeat this recursively on \( G_A \) and \( G_B \) (see Figure 13B). Each prime and degenerate graph is a **node** of the split tree. A node that is incident with exactly one tree edge is called a **leaf node**.
The minimal split decompositions and the split trees can be computed in quasi-linear time [23]. Similarly as in Propositions 1 and 2, it should be possible to derive every circle representation of a connected graph $G$ from the split tree $S$, but the precise statement is unclear. It is a natural question whether split trees can be used to solve the partial representation extension problem:

**Question 3.** Is it possible to use split trees to solve $\text{REPEXT(CIRCLE)}$? Can it be done faster that in time $O(n^3)$?

**ACKNOWLEDGMENTS**

We thank an anonymous reviewer for pointing out that the results described in Section 3 work only for maximal splits. The second author is supported by the People Programme (Marie Curie Actions) of the European Union’s Seventh Framework Programme (FP7/2007-2013) under REA grant agreement no (291734). The third author is supported by CE-ITI (P202/12/G061 of GAČR) and Charles University as GAUK 1334217.

**REFERENCES**

1. P. Angelini et al., *Testing planarity of partially embedded graphs*, ACM Trans. Algorithms 11 (2015), 32.
2. M. Balko, P. Klavík, and Y. Otachi, *Bounded representations of interval and proper interval graphs*, ISAAC, Vol. 8283, pp. 535–546; LNCS. Springer, Hong Kong, China, December 16-18 (2013).
3. J. Bang-Jensen, J. Huang, and X. Zhu, *Completing orientations of partially oriented graphs*, J. Graph Theory 87 (2015), no. 3, 285–304.
4. T. Bläsius, S. G. Kobourov, and I. Rutter, Simultaneous embedding of planar graphs R. Tamassia ed. *Handbook of Graph Drawing and Visualization*, CRC Press, Boca Raton, FL, 2013, pp. 349–383.
5. T. Bläsius, and I. Rutter, *Simultaneous PQ-ordering with applications to constrained embedding problems*, ACM Trans. Algorithms 12 (2015), 16.
6. K. S. Booth, and G. S. Lueker, *Testing for the consecutive ones property, interval graphs, and planarity using PQ-tree algorithms*, J. Comput. Syst. Sci. 13 (1976), 335–379.
7. A. Bouchet, *Reducing prime graphs and recognizing circle graphs*, Combinatorica 7 (1987), 243–254.
8. A. Bouchet, *Unimodularity and circle graphs*, Discrete Math. 66 (1987), 203–208.
9. S. Chaplick et al., *Contact representations of planar graphs: Extending a partial representation is hard*. WG’14, vol. 8747, pp. 139-151; LNCS. Springer, Athens, Greece, September 19-21 (2014).
10. S. Chaplick, R. Fulek, and P. Klavík, Extending partial representations of circle graphs, S. Wismath, and A. Wolff eds.*Graph drawing*, LNCS, Springer, Berlin, Heidelberg, 8242 2013, 131–142.
11. S. Chaplick et al., *The partial visibility representation extension problem*. Graph Drawing and Network Visualization: 24th International Symposium, GD 2016, vol. 9801, pp. 266-279; LNCS. Springer, International Publishing, Nouan-le-Fuzelier, France, June 25-27, (2016).
12. M. Chudnovsky et al., *The strong perfect graph theorem*, Ann. Math. 164 (2006), 51–229.
13. B. Courcelle, *Circle graphs and monadic second-order logic*, J. Appl. Logic 6 (2008), 416–442.
14. W. Cunningham, *Decomposition of directed graphs*, SIAM J. Algebr. Discrete Methods 3 (1982), 214–228.
15. E. Dahlhaus, *Parallel algorithms for hierarchical clustering and applications to split decomposition and parity graph recognition*, J. Algorithms 36 (1998), 205–240.
16. P. Damaschke, *The hamiltonian circuit problem for circle graphs is NP-complete*, Inform. Process. Lett. 32 (1989), 1–2.
17. H. de Fraysseix, *Local complementation and interlacement graphs*, Discrete Math. 33 (1981), 29–35.
18. H. de Fraysseix, and P. O. de Mendez, *On a characterization of gauss codes*, Discrete Comput. Geom. 22 (1999), 287–295.
19. S. Even, and A. Itai, *Queues, stacks, and graphs* Z. Kohavi, and A. Paz eds.
20. C. P. Gabor, K. J. Supowit, and W. Hsu, *Recognizing circle graphs in polynomial time*, J. ACM 36 (1989), no. 3, 435–473.
21. M. Garey et al., *The complexity of coloring circular arcs and chords*, SIAM J. Algebr. Discrete Methods 1 (1980), 216–227.
22. F. Gavril, *Maximum weight independent sets and cliques in intersection graphs of filaments*, Inform. Process. Lett. 73 (2000), 181–188.
23. E. Gioan et al., *Practical and efficient circle graph recognition*, Algorithmica 69 (2014), 759–788.
24. K. R. Jampani, and A. Lubiw, *The simultaneous representation problem for chordal, comparability and permutation graphs*, J. Graph Algorithms Appl. 16 (2012), 283–315.
25. P. Klavík et al., Extending partial representations of function graphs and permutation graphs Leah Epstein, and Paolo Ferragina eds. *Algorithms ESA 2012, LNCS*, Springer, Berlin, Heidelberg, 7501 2012, pp. 671–682.
26. P. Klavík et al., *Extending partial representations of proper and unit interval graphs*, Algorithmica 77 (2017), 1071–1104.
27. P. Klavík et al., *Extending partial representations of subclasses of chordal graphs*, Theor. Comput. Sci. 576 (2015), 85–101.
28. P. Klavík et al., *Extending partial representations of interval graphs*, Algorithmica 78 (2017), 945–967.
29. P. Klavík, J. Kratochvíl, and V. Vyskočil, *Extending partial representations of interval graphs*, M. Ogihara, and J. Tarui eds. *Theory and applications of models of computation, LNCS*, Springer, Berlin, Heidelberg, 6648 2011, pp. 276–285.
30. P. Klavík and M. Saumell, *Minimal obstructions for partial representations of interval graphs*, ISAAC, vol. 8889, pp. 401–413; LNCS. Springer, Berlin, Heidelberg and Jeonju, Korea, December 15-17 (2014).
31. A. Kostochka, and J. Kratochvíl, *Covering and coloring polygon-circle graphs*, Discrete Math. 163 (1997), 299–305.
32. T. Krawczyk and B. Walczak, *Extending partial representations of trapezoid graphs*, WG 2017; LNCS. Springer, Eindhoven, The Netherlands, June 21-23 (2017).
33. W. Naji, *Graphes de Cordes: Une Caracterisation et ses Applications*, Ph.D. Thesis, l’Université Scientifique et Médicale de Grenoble, 1985.
34. J. Opatrny, *Total ordering problem*, SIAM J. Comput. 8 (1979), 111–114.
35. S. Oum, *Rank-width and vertex-minors*, J. Comb. Theory, Ser. B 95 (2005), 79–100.
36. M. Patrignani, On extending a partial straight-line drawing Patrick Healy, and Nikola S. Nikolov eds. *Graph drawing, LNCS*, Springer, Berlin, Heidelberg, 3843 2006, pp. 380–385.
37. F. J. Soulignac, *Bounded, minimal, and short representations of unit interval and unit circular-arc graphs. Chapter I: theory*, J. Graph Algorithms Appl. 21 (2017), 455–489.
38. F. J. Soulignac, *Bounded, minimal, and short representations of unit interval and unit circular-arc graphs. Chapter II: Algorithms*, J. Graph Algorithms Appl. 21 (2017), 491–525.
39. J. P. Spinrad, *Recognition of circle graphs*, J. Algorithms 16 (1994), 264–282.
40. J. P. Spinrad, Efficient graph representations, Field Institute Monographs, American Mathematical Society, Providence, Rhode Island, 2003.

**How to cite this article:** Chaplick S, Fulek R, Klavík P. Extending partial representations of circle graphs. *J Graph Theory*. 2019;91:365–394.

https://doi.org/10.1002/jgt.22436