Extending Partial Representations of Circle Graphs in Near-Linear Time

Guido Brückner
Karlsruhe Institute of Technology, Germany

Ignaz Rutter
University of Passau, Germany

Peter Stumpf
University of Passau, Germany

Abstract

The *partial representation extension problem* generalizes the recognition problem for geometric intersection graphs. The input consists of a graph G, a subgraph $H \subseteq G$ and a representation $\mathcal{H}$ of H. The question is whether G admits a representation $\mathcal{G}$ whose restriction to H is $\mathcal{H}$. We study this question for *circle graphs*, which are intersection graphs of chords of a circle. Their representations are called *chord diagrams*.

We show that for a graph with n vertices and m edges the partial representation extension problem can be solved in $O((n+m)\alpha(n+m))$ time, where α is the inverse Ackermann function. This improves over an $O(n^3)$-time algorithm by Chaplick, Fulek and Klavík [2019]. The main technical contributions are a canonical way of orienting chord diagrams and a novel compact representation of the set of all canonically oriented chord diagrams that represent a given circle graph G, which is of independent interest.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis

Keywords and phrases circle graphs, partial representation extension, split decomposition tree, recognition algorithm

Digital Object Identifier 10.4230/LIPIcs.MFCS.2022.25

Funding Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Ru 1903/3-1.

1 Introduction

Geometric intersection representations of graphs are an important concept that establishes a strong connection between geometry, combinatorics and graph theory. In an intersection representation of a graph $G = (V, E)$ each vertex $v \in V$ is represented by a geometric object $R(v)$ whose intersections encode the edges of G, i.e., $\{u, v\} \in E$ if and only if $R(u)$ and $R(v)$ intersect. Different classes of graphs can be obtained by restricting the types of geometric objects used for the representation. For example *interval graphs* are intersection representations of intervals of the real line, *string graphs* are intersection graphs of curves in the plane and *circle graphs* are intersection graphs of chords of a circle; see Figure 1.

For a fixed class $\mathcal{C}$ of intersection graphs a natural question is the *recognition problem*, which asks whether a given graph G belongs to $\mathcal{C}$. For circle graphs the recognition problem has been studied for a long time, and has culminated in an algorithm with running time $O((n + m)\alpha(n + m))$ [10, 13], where n and m denote the number of vertices and edges of the input graph, respectively, and α denotes the slowly growing inverse of the Ackermann function. There is also an $O(n^2)$ time algorithm which is faster for very dense graphs [24].

A generalization of the basic recognition problem has attracted considerable attention: the *partial representation extension problem* [19, 1, 23, 2, 9, 18, 17, 21]. In this problem, the input consists of a triplet $(G, H, \mathcal{R}')$, where G is a graph, $H \subseteq G$ is an induced subgraph

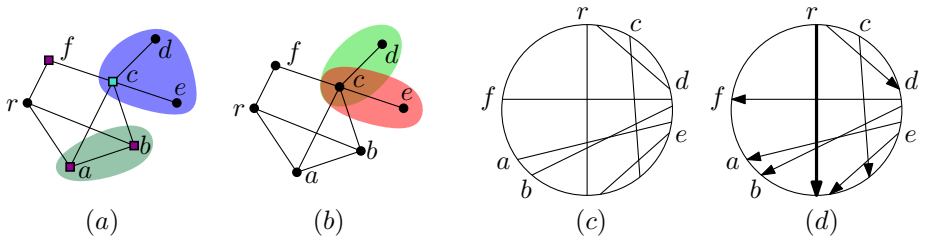

Figure 1 A graph G with good (a) and overlapping (b) splits. Only one side of each split is marked. The square vertices in (a) are the boundary of the blue split. (c) an undirected chord diagram of G (d) an oriented chord diagram of G with reference chord r.

of G, and $\mathcal{R}'$ is an intersection representation of H. The question is whether there exists a representation $\mathcal{R}$ of G that *extends* $\mathcal{R}'$, i.e., its restriction to H coincides with $\mathcal{R}'$. Following the notation of Chaplick, Fulek and Klavík [4], for a class $\mathcal{C}$ of intersection graphs, we denote the partial representation extension problem for $\mathcal{C}$ by RepExt($\mathcal{C}$).

Related Work

The recognition problem can be solved efficiently for a wide range of classes of intersection graphs. The partial representation extension problem was introduced by Klavík et al. [20], who gave an efficient algorithm for RepExt($\int$), where $\int$ denotes the class of interval graphs, which they improved to linear running time in their full version [19]. Angelini et al. [1] give a linear-time algorithm for planar topological graph drawings and Patrignani shows NP-hardness for planar straight-line drawings [23]. Recently, the problem has also been considered for simple topological and 1-planar drawings [2, 9]. In the meantime efficient algorithms are known for proper and unit interval graphs [18], permutation graphs and function graphs [17], as well as for trapezoid graphs [21]. Concerning the class Circle of circle graphs, Chaplick et al. [4] gave the first efficient algorithm for RepExt(Circle) with running time $O(n^3)$.

For other forms of representation, there are compact descriptions of all representations of a graph G, e.g., SPQR-trees [22, 8] for planar graphs and modular decomposition trees [11] for comparability graphs. Both descriptions express a representation of G by choosing representations for small graphs that are associated to the nodes of a tree in such a way that a bijection is obtained between the representations of G and the choices for the small graphs. For circle graphs, Cunningham and Edmonds introduced split-trees [7] that decompose a graph along its splits into smaller graphs from which G is assembled in a tree structure. Gioan and Paul [12] described split-trees using graph-labeled trees. Gioan et al. [13] observed that all possible chord diagrams of G can be obtained by choosing a chord diagram for each of the small graphs associated to the nodes of the split-tree of G and combining them suitably with each other.

Contribution and Outline

However, besides the choices of the chord diagrams, there are choices to be made when combining those diagrams, which Gioan et al. [13] express by basically directing the individual chords. These choices allow to obtain the same chord diagram starting with a different set of directed chord diagrams for the nodes of the split-tree; see Figure 2. We strengthen this connection by generalizing the concept of split-trees and introducing a canonical orientation of chord diagrams. In particular, for the canonical split-trees of Cunningham and Edmonds,

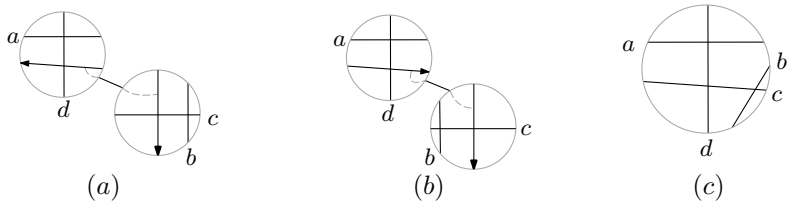

Figure 2 (a), (b) split-trees with different associated chord diagrams, only the relevant orientations are shown (c) the chord diagram resulting from the join for (a) and (b).

we establish a situation analogous to SPQR-trees or modular decomposition trees: the canonically oriented chord diagrams of G correspond bijectively to choices of canonically oriented chord diagrams for the nodes of the canonical split-tree.

In the $O(n^3)$-time algorithm of Chaplick et al. [4] for RepExt(Circle), they characterize all viable chord diagrams and solve the problem recursively, creating a decomposition that appears to be similar to the decomposition in our representation. To illustrate the usefulness of our representation, we improve over this result by showing how to extend partial chord diagrams in near-linear time $O((n+m)\alpha(n+m))$. This answers open questions of Chaplick et al. [4] and Kalisz et al. [16] who specifically ask whether such a representation of all chord diagrams exists and whether it can be used to solve RepExt(Circle) faster. We note that, given the data structure computed by the fastest known recognition algorithm by Gioan et al. [13], our algorithm runs in linear time.

We introduce notation and preliminary definitions in Section 2. In Section 3 we develop the compact description of representations. Section 4 shows how these results can be used to obtain an almost-linear time algorithm for RepExt(Circle). Lemmas and theorems marked with ($\star$) are proven in the full version.

2 Preliminaries

In this section we introduce important concepts that we use throughout the paper. In particular, we recall the concepts of circle graphs and chord diagrams, and we introduce a way to canonically orient them. Moreover, we also recall the notion of splits and split decompositions, which are a classic tool in circle graph recognition algorithms.

Circle Graphs and Chord Diagrams

An *(undirected) chord diagram* D consists of a set C of *chords* of the unit circle, i.e., undirected straight-line segments that connect pairwise distinct points on the unit circle. A chord diagram D naturally defines an intersection graph $G(D) = (C, E)$ of its chords, where $\{c, c'\} \in E$ if and only if the chords c and c' intersect in D, i.e., if and only if their endpoints alternate around the unit circle. A graph G is a *circle graph* if it is an intersection graph of the chords of a chord diagram.

While chord diagrams are geometric objects, we are mostly interested in their combinatorial structure, i.e., we identify chord diagrams with the same order of chord ends on the circle. To break certain symmetries we usually consider *oriented* chord diagrams, which additionally have a chord end as a *starting point*. Then an oriented chord diagram D can be encoded as the word over the set of chords C obtained by starting at the starting point and walking around the unit circle in clockwise direction and recording the encountered chords. For $c \in C$, we refer to the first end of c that we encounter with $\mathring{c}$ and to the second end with

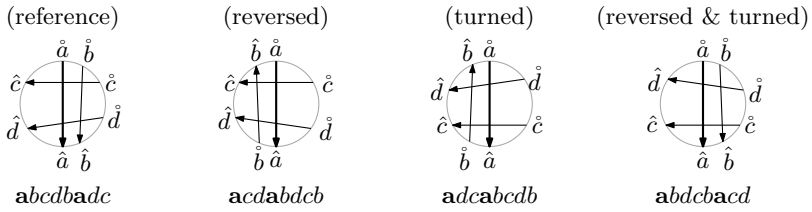

Figure 3 The turn and rev operation on a chord diagram with reference chord a.

$\hat{c}$. We consider the chord c as directed from $\mathring{c}$ to $\hat{c}$ in the geometric interpretation; see Figure 3 (reference). Unless mentioned otherwise, chord diagrams are always considered oriented and they are usually treated as their encodings.

We call the chord with the starting point the *reference chord* of D. We never change the reference chord. In the following, we consider changes that can be applied to every chord diagram without changing the represented graph or the reference chord. For a word w we write w^{rev} for the word obtained by reading w backwards. The *reverse* of a chord diagram $a\rho a\sigma$ is $\text{rev}(a\rho a\sigma) = a\sigma^{\text{rev}}a\rho^{\text{rev}}$. Geometrically, this corresponds to mirroring the chord diagram at the reference chord a; see Figure 3 (reversed). The *turn* of a chord diagram $a\rho a\sigma$ is $\text{turn}(a\rho a\sigma) = a\sigma a\rho$. Geometrically, this corresponds to choosing $\hat{a}$ as the new starting point. In a sense, this turns the chord diagram by 180 degrees; see Figure 3 (turned). Note that $\text{turn}(\text{turn}(a\rho a\sigma)) = a\rho a\sigma$, $\text{rev}(\text{rev}(a\rho a\sigma)) = a\rho a\sigma$ and $\text{turn}(\text{rev}(a\rho a\sigma)) = \text{turn}(a\sigma^{\text{rev}}a\rho^{\text{rev}})) = a\rho^{\text{rev}}a\sigma^{\text{rev}} = \text{rev}(a\sigma a\rho) = \text{rev}(\text{turn}(a\rho a\sigma))$, i.e., turn and rev are selfinverse and commute.

Splits and Split-Trees

Let $G = (V, E)$ be a graph. Consider a bipartition (X, Y) of V with $\emptyset \subsetneq X, Y \subsetneq V$ and let $B_X \subseteq X$ denote the subset of vertices of X that are adjacent to a vertex in Y and let $B_Y \subseteq Y$ denote the subset of vertices of Y that are adjacent to a vertex in X. The bipartition (X, Y) is called a *split* if all possible edges between B_X and B_Y exist in G, i.e., $\{\{x, y\} \mid x \in B_X, y \in B_Y\} \subseteq E$. Then (B_X, B_Y) is called the *split boundary*; see Figure 1. Observe that if X and Y are not connected, then B_X, B_Y are empty. A split (X, Y) is *trivial* if one of its sides consists of a single vertex. Following Courcelle [5], we call a split (X, Y) *good* if it does not overlap any other split, in the sense that for any other split (W, Z) at least one of $X \cap W$, $X \cap Z$, $Y \cap W$, $Y \cap Z$ is empty. A graph is *prime* if all its splits are trivial, and it is *degenerate* if every bipartition of the vertices yields a split. It is well known that the connected degenerate graphs are precisely cliques and stars [6].

Next, we define *split-trees*, introduced by Cunningham and Edmonds [7] as decomposition trees. We use the graph-labeled tree description of Gioan and Paul [12]. However we consider graph-labeled trees that correspond to general decompositions in the sense of Cunningham and Edmonds, while Gioan and Paul used the term split-tree for a unique structure which we later define as the canonical split-tree. To avoid confusion with the vertices and edges of our graphs, we refer to the vertices and edges of split-trees as nodes and arcs, respectively. A *split-tree* T is a tree where each inner node μ has a *skeleton graph* $\text{skel}(\mu)$ and a bijection corr_μ from $V(\text{skel}(\mu))$ to the nodes of T adjacent to μ; see Figure 4. Given an inner node μ and a neighbor ν of μ, we often need to refer to the vertex v of $\text{skel}(\mu)$ that represents ν. For convenience, we define $v_\mu(\nu)$ as the vertex v of $\text{skel}(\mu)$ with $\text{corr}_\mu(v) = \nu$.

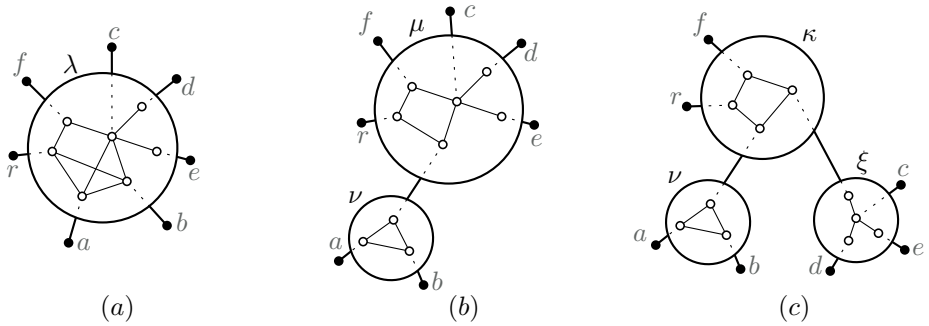

Figure 4 Split-trees of the graph from Figure 1a. (a) shows the base case, in (b) the node λ from (a) is decomposed into μ and ν along the split $(\{a,b\},\{c,d,e,r,f\})$. In (c) the node μ from (b) is further decomposed along the split that has $\{c,d,e\}$ on one side.

We define split-trees inductively. Let $G = (V, E)$ be a graph. In the base case; see Figure 4a, a split-tree T of G consists of one inner node λ and one leaf for each vertex $v \in V$ that is adjacent to λ. We identify the leaves of T with the vertices in V. Define the skeleton of λ as $\text{skel}(\lambda) = G$. For every vertex $v \in V(\text{skel}(\lambda))$, we define $\text{corr}_\lambda(v)$ as the leaf node v of T.

We can further decompose such a split-tree. Let λ be an inner node and let (X, Y) be a non-trivial split of $\text{skel}(\lambda)$. Let G_X denote the graph obtained from $\text{skel}(\lambda)$ by contracting Y into a single vertex y. Symmetrically, let G_Y denote the graph obtained from $\text{skel}(\lambda)$ by contracting X into a single vertex x. Then λ can be split into two nodes μ, ν connected by an arc $\{\mu, \nu\}$; see Figure 4b. Define $\text{skel}(\mu) = G_X$ and $\text{skel}(\nu) = G_Y$. Observe that y in G_X represents the graph G_Y and x in G_Y represents the graph G_X. We define $\text{corr}_\mu(y) = \nu$ and $\text{corr}_\nu(x) = \mu$. For any vertex $v \in V(\text{skel}(\lambda))$ we replace the arc $\{\lambda, \text{corr}_\lambda(v)\}$ with $\{\mu, \text{corr}_\lambda(v)\}$ if $v \in X$ or $\{\nu, \text{corr}_\lambda(v)\}$ if $v \in Y$. Finally, for each inner node κ adjacent to λ consider the unique vertex $v \in \text{skel}(\kappa)$ with $\text{corr}_\kappa(v) = \lambda$. We redefine $\text{corr}_\kappa(v) = \mu$ if $v_\lambda(\kappa) \in X$ and $\text{corr}_\kappa(v) = \nu$ if $v_\lambda(\kappa) \in Y$. Observe that the result is still a tree and corr_ξ is well defined for each inner node ξ. Hence, the inner nodes may be decomposed again. The split-trees of G are the split-trees that can be obtained in this way.

The inverse of a decomposition is a *join*. Let G_1, G_2 be two (vertex-disjoint) graphs with $v_2 \in G_1$, $v_1 \in G_2$. Then we define their *join at* v_2,v_1 as the graph obtained from $G_1 \cup G_2$ by connecting all neighbors of v_2 in G_1 with all neighbors of v_1 in G_2 and removing the vertices v_1, v_2. We denote the resulting graph by $G_1 \oplus_{v_2,v_1} G_2 = (V, E)$. For a split-tree T with an arc $\{\mu, \nu\}$ let $b = v_\mu(\nu)$, $a = v_\nu(\mu)$. The nodes μ, ν can be joined into a single node λ with $\text{skel}(\lambda) = \text{skel}(\mu) \oplus_{b,a} \text{skel}(\nu)$. Moreover, for any inner node κ adjacent to μ or ν and the unique vertex $v \in V(\text{skel}(\kappa))$ with $\text{corr}_\kappa(v) \in \{\mu, \nu\}$ we redefine $\text{corr}_\kappa(v) = \lambda$.

Let T denote a split-tree and let $\{\mu, \nu\}$ be an arc of T. Removing $\{\mu, \nu\}$ separates T into two trees T_μ and T_ν where T_μ contains the node μ and T_ν contains the node ν. Let $L(T_\mu) \subseteq V$ denote the set of leaves of T_μ. By construction of the split-tree $(L(T_\mu), L(T_\nu))$ is a split of G, which is induced by the arc $\{\mu, \nu\}$. For example, the blue split from Figure 1a is represented by the arc $\{\kappa, \xi\}$ in Figure 4c. Let (A, B) be a non-trivial split of $\text{skel}(\mu)$. This split induces a split (L_A, L_B) of G with $L_A = \bigcup_{v \in A} L(T_{\text{corr}_\mu(v)})$ and $L_B = \bigcup_{v \in B} L(T_{\text{corr}_\mu(v)})$. An example of this is the red split from Figure 1b that is represented by a non-trivial split inside node ξ of Figure 4c. We call an inner node of a split-tree *degenerate* and *prime*, if its skeleton graph is degenerate and prime, respectively. Observe that in the split-tree in Figure 4c, the node κ is prime, and ν, ξ are degenerate.

In a *good-split-tree* every inner arc $\{\mu, \nu\}$ of T induces a good split. For a connected graph G, a *canonical split-tree* is a good-split-tree such that no skeleton has a non-trivial good split, no inner arc induces a trivial split, and any two arcs induce different splits. A canonical split-tree is obtained by decomposing G (in arbitrary order) along all its good splits. This is possible since the good splits form a laminar set. The resulting canonical split-tree is denoted by $\mathrm{ST}(G)$. It was first defined by Cunningham and Edmonds [7]. We use the graph-labeled-tree description by Gioan and Paul [12]. It is unique, and nodes have connected skeletons that are either prime or degenerate. Moreover, if two adjacent nodes μ, ν are both degenerate, then either one of them is a star and one is a clique, or they are both stars and the vertices $v_\mu(\nu)$ and $v_\nu(\mu)$ are either both leaves or both the star centers of their respective skeletons. By the following useful property, $\mathrm{ST}(G)$ represents all splits of G.

▶ **Proposition 1** ([6],[14, Theorem 2.18]). *A partition (A, B) of the vertex set of a connected graph G is a split of G if and only if it is induced by an arc of $\mathrm{ST}(G)$ or by a non-trivial split of a (skeleton of a) degenerate node of $\mathrm{ST}(G)$.*

3 Compact Representation of all Chord Diagrams

Let $G = (V, E)$ be a circle graph. In Section 3.1 we establish a correspondence between the chord diagrams of G and assignments of chord diagrams to the inner nodes of a split-tree of G. For canonical split-trees, this correspondence turns out to be a bijection. We further show that both directions of this bijection can be computed in linear time. In Section 3.2 we describe the possible choices for the chord diagrams for nodes of a canonical split-tree of G. Altogether, this gives the claimed compact representation for connected circle graphs.

3.1 Configurations of Split-Trees

Let $G = (V, E)$ be a circle graph, let $r \in V$ be the reference chord of G and let T be a split-tree of G. We root T at r and direct all arcs away from the root. For each inner node ν of T, we define the reference chord r_ν as the chord $v_\nu(\mu)$ associated with the parent μ of ν. A *configuration* c of T is a mapping that assigns to each inner node μ of T a chord diagram $c(\mu)$ of $\mathrm{skel}(\mu)$ with reference chord r_μ; see Figure 5. We also refer to $c(\mu)$ as the *configuration of* μ. Recall that a split decomposition can be used to split a node λ of T into two nodes μ, ν connected by a (directed) arc (μ, ν). In the reverse direction, a join composition can be used to join two nodes μ, ν connected by an arc (μ, ν) into a single node λ. The join operation extends to configurations.

The configurations of μ and ν induce a unique configuration of λ as follows. Let $v = v_\mu(\nu)$ and let $c(\nu) = r_\nu \rho r_\nu \sigma$. The induced configuration $c(\lambda) = c(\mu) \oplus_{v, r_\nu} c(\nu)$ of λ is obtained from $c(\mu)$ by replacing $\hat{v}$ with ρ and $\hat{v}$ with σ. See Figure 5b. The order of multiple joins affects neither the resulting split-tree nor the resulting configuration. For a configuration c of a split-tree T we denote by $\mathcal{D}(c)$ the chord diagram obtained by joining the configurations of all inner nodes of T. For connected graphs, the two joined chord diagrams $c(\mu)$, $c(\nu)$ are fully determined by their join $c'(\lambda)$ since the four combined words ρ, σ, μ, ν are fully determined by $c'(\lambda)$. This implies that two different configurations yield two different chord diagrams.

If G is connected, a diagram D can be decomposed along a split (A, B) only if the endpoints of the chords in A and B appear suitably in D. We say that D *respects* the split (A, B) if D decomposes into two words over A and two words over B that alternate. Formally, this means we have $D = \rho_1\sigma_1\rho_2\sigma_2\rho_3$ where $\rho_2, (\rho_3\rho_1)$ and σ_1, σ_2 are words over A and B.

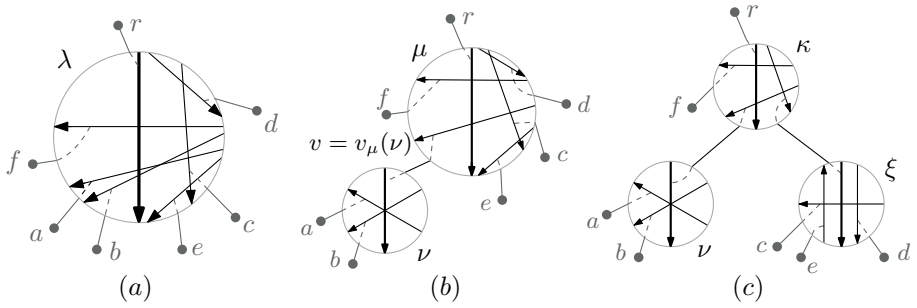

Figure 5 (a), (b), (c) show configurations of the corresponding split-trees in Figure 4 that yield the same chord diagram. We have $c'(\lambda) = rdcdf\mathbf{ba}ecer\mathbf{ba}f = rdcdf\mathbf{v}ecer\mathbf{v}f \oplus_{v,r_\nu} r_\nu bar_\nu ba = c(\mu) \oplus_{v,r_\nu} c(\nu)$, where $v = v_\mu(\nu)$.

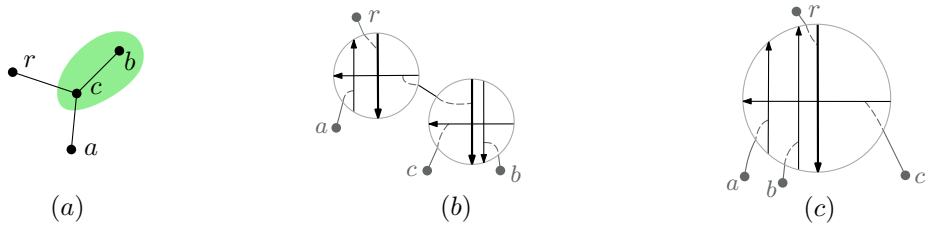

Figure 6 (a) A star G with a non-good split ({r,a},{b,c}). (b) A split-tree T of G with an arc corresponding to the bad split in (a). (c) A chord diagram of G that is not represented by T.

We show that a chord diagram can be recursively decomposed along several splits if it respects each of them. This yields the following lemma.

▶ **Lemma 2** (⋆). *Let G be a (not necessarily connected) circle graph and let T be a split-tree of G rooted at some reference chord r. Then the mapping $\mathcal{D}$ that maps configurations of T to chord diagrams of G is surjective on the set of chord diagrams of G with reference chord r that respect all splits induced by arcs of T.*

Note that, for any configuration c of T, the chord diagram $\mathcal{D}(c)$ respects all splits induced by arcs of T. Since it is known that a chord diagram D of a connected circle graph G respects all good splits of G [5, Proposition 9], we conclude the following.

▶ **Theorem 3.** *Let T be a good-split-tree of a connected circle graph G rooted at some reference chord $r \in V(G)$. Then the mapping $\mathcal{D}$ that maps configurations of T to chord diagrams of G with reference chord r is a bijection.*

We can translate between configuration and chord diagram in linear time, which allows us to use this result algorithmically. Note that in the following theorem the split-trees are not required to be good-split-trees, which means there may be chord diagrams for their graph that they do not represent; see Figure 6. This will be useful when dealing with RepExt(Circle).

▶ **Theorem 4.** *Let T be a split-tree of a connected circle graph G rooted at some reference chord $r \in V(G)$. Then the mapping $\mathcal{D}$ can be computed in linear time. Conversely, given a chord diagram D of G with reference chord r, it can be tested in linear time whether there exists a configuration c of T with $\mathcal{D}(c) = D$. If it exists, the configuration c can also be computed in linear time.*

Proof. We store chord diagrams as circular doubly linked lists of endpoints of chords in clockwise order. We assume that each chord endpoint is equipped with a pointer to the corresponding vertex and each vertex has pointers to the two endpoints of its chord.

Let c be a configuration of T. We first store for each chord u in a chord diagram which of its ends is $\mathring{u}$. We then process the tree T in bottom-up order. If c contains only a single inner node, then the configuration of this node is the desired diagram $\mathcal{D}(c)$. Otherwise let μ be an inner node of T whose children are all leaves and let ν be its parent. Let T' be the split tree obtained by replacing in T the node μ together with all its leaves by a single leaf v and let c' be the configuration of T' that coincides with c for all nodes of T'. Clearly, c' and T' can be computed from c and T in $O(1)$ time. We now recursively compute $\mathcal{D}(c')$ in time linear in the size of T'. To obtain $\mathcal{D}(c)$, we replace in $\mathcal{D}(c')$ the endpoints $\mathring{v}, \hat{v}$ of v by the sequences π, σ, respectively, where $c(\mu) = r\pi r\sigma$ and r is the reference chord of μ. Since we maintain the order of the endpoints in doubly linked lists, this can be done in $O(1)$ time. Therefore we only spend $O(1)$ time per node of T.

Conversely assume that D is a chord diagram of G with reference chord r. We again process the tree T in bottom-up order. As before, if T contains only a single inner node μ then $c(\mu) = D$ is the desired configuration. Otherwise let μ be an inner node whose children are all leaves. We denote the set of leaves of μ by L. Note that $(L, V \setminus L)$ is a split of G. Let T' be the tree obtained from T by replacing μ and its leaves by a single leaf v. Using simple flags, we can decide for an endpoint of a chord of D in constant time whether it belongs to a vertex of L and if so, whether it has already been processed. We now treat the endpoints of the vertices in L one by one. For the first endpoint e we obtain in this way, we scan to the left and right in the doubly linked list of D starting at e. In this way we determine a maximal sublist $[e_1, e_2]$ of D (containing all endpoints that lie clockwise between the first and last points e_1, e_2 of that list) with $e \in [e_1, e_2]$ such that all elements of $[e_1, e_2]$ belong to vertices of L. We mark all endpoints that we encounter in this way as processed. We then scan further the leaves of L. We do find another endpoint f that is not yet processed since G is connected. We similarly determine a sublist $[f_1, f_2]$ around f so that all endpoints that lie clockwise between f_1, f_2 belong to vertices of L and the predecessor of f_1 and the successor of f_2 do not. If there is an unprocessed endpoint left, this means D does not respect split $(L, V \setminus L)$ and we can reject. Hence, assume no unprocessed endpoint remains.

We thus have partitioned the endpoints of the chords of L into two disjoint sublists $[e_1, e_2]$ and $[f_1, f_2]$ of D. Then let D' be the diagram obtained from D by replacing the sublists $[e_1, e_2]$, $[f_1, f_2]$ each with v. We recursively compute a configuration c' of T' with $\mathcal{D}(c') = D'$ where for each chord u the end $\mathring{u}$ is stored. If this succeeds, we obtain the desired configuration c as follows. For each inner node $\nu \neq \mu$ we set $c(\nu) = c'(\nu)$. Since we know from each predecessor and successor u of $\mathring{v}$, $\hat{v}$ whether it is $\mathring{u}$, we can set $c(\mu) = r[e_1, e_2]r[f_1, f_2]$ or $c(\mu) = r[f_1, f_2]r[e_1, e_2]$ accordingly, where r is the reference chord of $\mathrm{skel}(\mu)$. By construction it is $\mathcal{D}(c) = D$. Note that any other choice of $c(\mu)$ or $\mathcal{D}(c_{|T-\mu})$ results in a chord diagram different from D, since $[e_1, e_2]$ and $[f_1, f_2]$ are non-empty and separated by chords not in L. We can then first iterate through the list of $[e_1, e_2]$, $[f_1, f_2]$ that replaces $\mathring{v}$ and then other one to store each endpoint $\mathring{u}$.

The time spent to compute T', D' as well to modify c' into c is proportional to $|L|$. Therefore the algorithm runs in linear time. ◀

3.2 Configurations of Canonical Split-Trees

In a chord diagram D of a degenerate circle graph G, i.e., of a clique or a star, one half of D determines the other half. More precisely, we can describe a chord diagram of a connected degenerate circle graph $G = (V, E)$ with reference chord $r \in V$ with a cyclic permutation of V

using the following mapping $\phi_{G,r}$ to chord diagrams of G with reference chord r. If G is a clique, we set $\phi_{G,r}(r\rho) = r\rho r\rho$. If G is a star with center x, we set $\phi_{G,r}(x\rho\sigma) = \rho^{\text{rev}} x\rho\sigma x\sigma^{\text{rev}}$ where ρ ends with r or is empty if $x = r$.

▶ **Lemma 5** (⋆). *For a clique or a star $G = (V, E)$ with $r \in V$ the map $\phi_{G,r}$ is a bijection.*

Bouchet [3] showed that the undirected chord diagram of a connected prime circle graph is unique up to reversal. We can additionally choose the orientation of the reference chord. As a shorthand, we set $\text{tr} = \{\text{id}, \text{turn}, \text{rev}, \text{turn}(\text{rev})\}$ and for any chord diagram D we set $\text{tr}(D) = \{D, \text{turn}(D), \text{rev}(D), \text{turn}(\text{rev}(D))\}$. Note that we can have $|\text{tr}(D)| < 4$, e.g., for cliques we have $\text{turn}(D) = D$.

▶ **Lemma 6** (⋆). *Let G be a connected prime circle graph, $r \in V(G)$, and D a chord diagram with reference chord r. Then $|\text{tr}(D)| = 4$ and $\text{tr}(D)$ is the set of chord diagrams of G with reference chord r.*

By combining Theorem 3 with Lemmas 5, 6 we obtain the claimed compact representation of all chord diagrams of a connected circle graph G.

▶ **Theorem 7.** *Let G be a connected circle graph and let T be the canonical split-tree of G with reference chord $r \in V(G)$. Let each prime node μ be equipped with a chord diagram of $\text{skel}(\mu)$ with reference chord r_μ. There is a bijection between the chord diagrams of G with reference chord r and the choices of (i) applying an operation $\tau_\mu \in \text{tr}$ to each prime node μ and (ii) choosing a cyclic permutation of $V(\text{skel}(\mu))$ for each degenerate node μ.*

4 Partial Representation Extension

In this section, we solve RepExt(Circle) for a given circle graph $G = (V, E)$ and a chord diagram D_H of an induced subgraph $H \subseteq G$ in near-linear time. The idea is the following. Assume G is connected and let T be the canonical split-tree of G. By Theorem 3, all chord diagrams of G are represented by T. We project T and its configurations in a sense on H and obtain an enriched split-tree T_H of H. We show that T_H describes exactly the chord diagrams of H that can be extended to G. This means we just need to check whether T_H describes the given chord diagram D_H.

If G is disconnected and there is a connected component C with two predrawn chord ends a, b, and a distinct connected component C' with two predrawn chord ends c, d in D_H such that a, b and c, d alternate in D_H, then there is no extension of D_H to G since C, C' need to cross each other (even though the crossing chords might not be predrawn). Chaplick et al. [4] argue that otherwise an extension exists if and only if each connected component of G admits an extension. Testing this requirement as well as combining representation extensions of the different components can be done in linear time. Hence, we assume in the following that G is connected. Note that H may still be disconnected.

We start with a canonical split-tree T of G rooted at a reference chord $r \in V(H)$, which by Theorem 3 represents all chord diagrams of G. For a chord diagram D of G let $D|_H$ denote the chord diagram for H induced by D (i.e., the chords of H are placed as in D with the same starting point). Let T_H be the subtree of T whose leaves are the vertices of H and whose inner nodes are the inner nodes of T that lie on a path from r to some leaf in $V(H)$. For each inner node of T_H, we define $\text{skel}_{T_H}(\mu)$ as the subgraph of $\text{skel}_T(\mu)$ induced by the vertices $v_\mu(V(T_H))$, i.e., we keep exactly those chords that represent nodes that lead to at least one vertex of H (see Figure 7). Finally, we suppress nodes with K_2 as skeleton in T_H

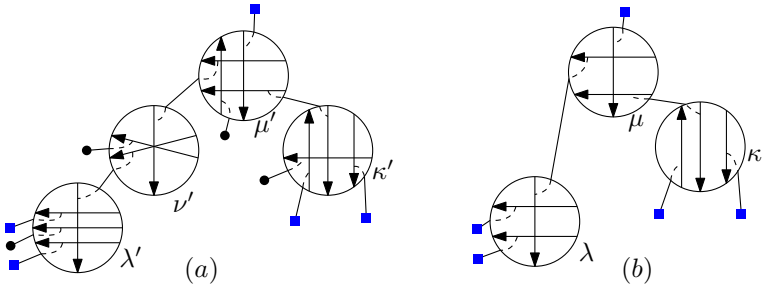

Figure 7 (a) Configuration of a canonical split-tree T. Leaves in $V(H)$ are blue squares. (b) split-tree T_H with the corresponding configuration. Node μ is degenerate in T_H while μ' is prime in T. skel(κ) is an isolated set while skel(κ') is a star.

by (iteratively) joining them with one of their neighbors. Note that T_H is a split-tree of H rooted at r, and each inner node μ of T_H stems from exactly one inner node μ' of T. We call T_H the *projection of* T *onto* H.

Let now c be a configuration of T. We define its *projection* c_H by setting $c_H(\mu) = c(\mu')|_{\text{skel}_{T_H}(\mu)}$ for each node μ of T_H, i.e., it is the restriction of the chord diagram $c(\mu')$ to $\text{skel}_{T_H}(\mu)$. Observe that the reference chord is not removed, and therefore $c_H(\mu)$ has the same reference chord as $c(\mu')$, i.e., c_H is a configuration of T_H. The following lemma shows that the projection $(T, c) \mapsto (T_H, c_H)$ commutes with all relevant operations.

▶ **Lemma 8.** *We have (i)* $\mathcal{D}(c)|_H = \mathcal{D}(c_H)$ *and (ii) for every inner node* μ *of* T_H, $\text{turn}(c_H(\mu)) = \text{turn}(c(\mu'))|_{\text{skel}_{T_H}(\mu)}$ *and* $\text{rev}(c_H(\mu)) = \text{rev}(c(\mu'))|_{\text{skel}_{T_H}(\mu)}$.

Proof. For Property (i) observe that it suffices to show that joining two diagrams commutes with the projection to a subgraph H. It then follows that $\mathcal{D}(c)|_H$, where the join is projected to H, is the same as $\mathcal{D}(c_H)$, where the skeletons are projected before the join, coincide.

More formally, let H_1, H_2 be two induced subgraphs of graphs G_1, G_2, respectively, and let $H = H_1 \oplus_{v_2,v_1} H_2$ and $G = G_1 \oplus_{v_2,v_1} G_2$. We show that for any chord diagrams D_1, D_2 of G_1, G_2, respectively, it is $(D_1 \oplus_{v_2,v_1} D_2)|_H = D_1|_{H_1} \oplus_{v_2,v_1} D_2|_{H_2}$. To see this, let $D_1 = \alpha v_2 \beta v_2 \gamma$ and $D_2 = v_1 \rho v_1 \sigma$ and let $\alpha', \beta', \gamma', \rho', \sigma'$ be the words obtained from $\alpha, \beta, \gamma, \rho, \sigma$ by removing all symbols for chords that are not in $V(H)$. Then we have $(D_1 \oplus_{v_2,v_1} D_2)|_H = (\alpha\rho\beta\sigma\gamma)|_H = \alpha'\rho'\beta'\sigma'\gamma'$. On the other hand, it is $D_1|_{H_1} \oplus_{v_2,v_1} D_2|_{H_2} = \alpha' v_2 \beta' v_2 \gamma' \oplus_{v_2,v_1} v_1\rho' v_1\sigma' = \alpha'\rho'\beta'\sigma'\gamma'$.

For Property (ii), let H be an induced subgraph of G. Let $D = r\rho r\sigma$ be a chord diagram for G with reference chord r and let ρ', σ' be the restrictions of ρ, σ to H, respectively. Then $\text{turn}(D)|_H = \text{turn}(r\rho r\sigma)|_H = (r\sigma r\rho)|_H = r\sigma' r\rho' = \text{turn}(r\rho' r\sigma') = \text{turn}((r\rho r\sigma)|_H) = \text{turn}(D|_H)$ and $\text{rev}(D)|_H = \text{rev}(r\rho r\sigma)|_H = (r\sigma^{\text{rev}} r\rho^{\text{rev}})|_H = r\sigma^{\text{rev}} r\rho^{\text{rev}}|_H = r\sigma'^{\text{rev}} r\rho'^{\text{rev}} = \text{rev}(r\rho' r\sigma') = \text{rev}(D|_H)$. ◀

Let D_H be a chord diagram of H. By Theorem 3 there exists a chord diagram of G that extends D_H if and only if there exists a configuration c of T with $\mathcal{D}(c)|_H = D_H$. By Lemma 8(i) this holds if and only if there exists a configuration c of T whose projection c_H satisfies $\mathcal{D}(c_H) = \mathcal{D}(c)|_H = D_H$. We aim to find such a configuration c. To do this, we make use of the property from Lemma 8(ii) as follows. Let c' be an arbitrary configuration of T. By Theorem 7, the configuration c is obtained from c' by (i) choosing a configuration for each degenerate node of T and (ii) by choosing for each prime node μ one of the diagrams $c(\mu) \in \text{tr}(c'(\mu))$. Note that induced subgraphs of cliques are themselves cliques and an induced subgraph of a star is either a star or an independent set. In the latter case

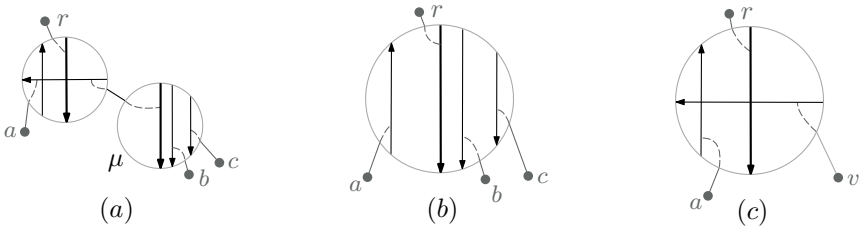

Figure 8 (a) a restricted split-tree for a graph H (b) a chord diagram D_H for H where the chord ends of the leaves of μ are contiguous. (c) the chord diagram we wish to create from D_H by replacing b,c by v. Note that the position of $\hat{v}$ is not neighboring the position of any leaf of μ in (b).

an induced chord diagram has the form $\rho^{\text{rev}}\rho\sigma\sigma^{\text{rev}}$ where the center of the original star has one end between ρ^{rev} and ρ and one end between σ and σ^{rev}. By Lemma 8(ii) it follows that c_H is obtained from c'_H by (i) arbitrarily choosing a configuration for each node of T_H that stems from a degenerate node of T and is connected, (ii) choosing a configuration of the form $\rho^{\text{rev}}\rho\sigma\sigma^{\text{rev}}$ for each node of T_H that stems from a degenerate node of T and is an independent set and (iii) by choosing for each node μ of T_H that stems from a prime node in T one of the diagrams $c_H(\mu) \in \text{tr}(c'_H(\mu))$.

We condense these rules as follows. We label the nodes of T_H as degenerate if they stem from a degenerate node in T and as prime if they stem from a prime node of T. We call two configuration c_H, c'_H of T_H *equivalent* if $c_H(\mu) \in \text{tr}(c'_H(\mu))$ for each prime-labeled node, and for each degenerate-labeled node ν where skel(ν) is an independent set, $c_H(\nu)$ is of the form $\rho^{\text{rev}}\rho\sigma\sigma^{\text{rev}}$. We call (T_H, c'_H) *the restricted split-tree of H with respect to G* and say that (T_H, c'_H) *represents* a diagram D_H of H if $D_H = \mathcal{D}(c_H)$ for some configuration c_H that is equivalent to c'_H. By the above observations, D_H can be extended to a diagram of G if and only if D_H is represented by (T_H, c'_H), where c'_H is the projection of a configuration of T.

If H is connected, this condition can be tested in linear time by computing the unique configuration c_H of T_H with $\mathcal{D}(c_H) = D_H$ using Theorem 4 and then checking whether it is equivalent to c'_H. If H is not connected, we use the following lemma to test whether (T_H, c'_H) represents D_H and to obtain a corresponding configuration c_H of T_H in that case.

▶ **Lemma 9 (⋆).** *Let (T_H, c'_H) be a restricted split-tree of a graph H with respect to a connected graph G and let D_H be a chord diagram of H with the root r of T_H as reference chord. It can be tested in linear time whether (T_H, c'_H) represents D_H. If so, a corresponding configuration c_H can also be computed in linear time.*

Sketch of proof. We use a bottom-up approach as in the proof of Theorem 4 to find c_H. Recall that in the proof of Theorem 4, we iteratively processed an inner node μ with only leaves as children and searched for subwords of D_H that correspond to a chord diagram D_μ of skel(μ) to check whether D_μ can be part of a configuration and then replace these subwords with a chord that represents μ as a leave for the remaining split-tree. To find these subwords we started at arbitrary leaves of μ to find contiguous sublists of such leaves in D_H. However, if H is not connected, it can happen that when processing a node μ, all its leaves are contiguous in D_H; see Figure 8. For example, this happens if μ is a leftmost leaf of a star node in T that lost its center in T_H (something similar can happen in prime-labeled nodes). In this case the first found sublist $[e_1, e_2]$ of leaves of μ already contains all leaves of μ and we do not find a second sublist $[f_1, f_2]$. This means that we have no pointer to

the correct place in the new chord diagram D' for the second end of the leaf chord v that replaces μ. We thus allow D_H to be a relaxed chord diagram where for some chords only one end is fixed. We then use the characterization of inner nodes of restricted split-trees to compute configurations for those nodes that match the given relaxed chord diagrams.

For example, if μ is a labeled degenerate and $\text{skel}(\mu)$ is a star with center x, we start at an end of x in D_H and traverse simultaneously in both directions until all ends are traversed (except possibly the second end of x). At each end of a chord v with no second end, insert that second end where the other traversal is at that moment. When a chord with two ends is reached, wait in front of that chord until the other traversal also reaches that chord and then skip that chord in both traversals. If the traversals wait at different chords we reject. In that case D_H induces the subword $xaabb$ (or some cyclic shifted version of that word) which cannot be realized by a star with center x. If x has a second end, the traversals meet and end there. Otherwise, add the second end of x where the traversals meet. Thereby, all chords for leaves intersect x and no other intersection occurs. By construction we obtain a chord diagram $c_H(\mu)$ that realizes D_H.

Note that both ends of v can be neighbors in $\mathcal{D}(c_H|_{T_H-\mu})$. Then, moving the end $\hat{r}$ of the reference chord in $c_H(\mu)$ does not change $\mathcal{D}(c_H)$ and c_H is no longer required to be unique. ◀

▶ **Theorem 10.** *Given $T = \text{ST}(G)$ and a chord diagram D of G, RepExt(Circle) can be solved in linear time. In the positive case a representation of G that extends the given diagram D_H of H can be computed in the same running time.*

Proof. From D we compute in linear time a configuration c' of T with $\mathcal{D}(c') = D$ using Theorem 4. From T and c', we compute in linear time the projection to the restricted split-tree (T_H, c'_H) of H. With Lemma 9, we test whether it represents the given diagram D_H of H and obtain a configuration c_H equivalent to c'_H with $\mathcal{D}(c_H) = D_H$. We now define a configuration c of T as follows. For each prime-labeled node μ of T_H, we define $c(\mu') = \tau(c'(\mu'))$ where $\tau \in \text{tr}$ such that $c_H(\mu) = \tau(c'_H(\mu))$. For each degenerate-labeled node μ of T_H, we choose a configuration as follows. If $\text{skel}_H(\mu)$ is connected we can extend $c'(\mu)$ arbitrarily. For a precise argument, we have by Lemma 5, $c_H(\mu) = \phi_{\text{skel}_H(\mu),r}(\sigma)$ for some cyclic permutation σ of $V(\text{skel}_{T_H}(\mu))$. We create a permutation σ' of $V(\text{skel}_T(\mu))$ by appending the elements of $V(\text{skel}_T(\mu')) \setminus V(\text{skel}_{T_H}(\mu))$ to σ in an arbitrary order. We then set $c(\mu') = \phi_{H,r}(\sigma')$. If $\text{skel}_{T_H}(\mu)$ is not connected, then $\text{skel}_T(\mu')$ is a star where the reference chord $r_{\mu'}$ is not the center x and $\text{skel}_{T_H}(\mu)$ does not contain x. In that case $c_H(\mu)$ is of the form $c_H(\mu) = \rho^{\text{rev}}\rho\sigma\sigma^{\text{rev}}$ and we have to insert the other chords in parallel, such that x can intersect all chords. Then set $c(\mu') = \rho^{\text{rev}}x\rho\sigma\alpha x\alpha^{\text{rev}}$, where α are the elements of $V(\text{skel}_T(\mu)) \setminus V(\text{skel}_{T_H}(\mu)) \setminus \{x\}$. Finally, for each node μ' of T that is not contained in T_H, we set $c(\mu') = c'(\mu')$. By construction, we have that c_H is the projection of c, and therefore $\mathcal{D}(c)|_H = \mathcal{D}(c_H) = D_H$, i.e., $\mathcal{D}(c)$ is the desired representation of G. Clearly the amount of work per skeleton is linear, and therefore the overall running time is linear. ◀

Theorem 10 assumes that $\text{ST}(G)$ and a chord diagram of G are available. If not, we can compute them in $O((n+m)\alpha(n+m))$ time [13, 14].

▶ **Corollary 11.** *RepExt(Circle) can be solved in $O((n+m)\alpha(n+m))$ time.*

5 Conclusion

We have developed a data structure that compactly represents all chord diagrams for a connected circle graph. As an application, we have shown how to solve the partial representation extension problem for circle graphs in almost linear time, improving over

the $O(n^3)$ algorithm of Chaplick et al. [4]. Using a reduction of Chaplick et al. this also solves the extension problem for permutation graphs in near linear time, improving over two different $O(n^3)$ algorithms [4], [17]. Our data structure may also be useful when seeking restricted chord diagrams that satisfy additional constraints. For example, we believe that it is possible to significantly simplify the circular-arc graph recognition of Hsu et al. [15].

References

1 Patrizio Angelini, Giuseppe Di Battista, Fabrizio Frati, Vít Jelínek, Jan Kratochvíl, Maurizio Patrignani, and Ignaz Rutter. Testing planarity of partially embedded graphs. *ACM Trans. Algorithms*, 11(4):32:1–32:42, 2015. doi:10.1145/2629341.

2 Alan Arroyo, Martin Derka, and Irene Parada. Extending simple drawings. In Daniel Archambault and Csaba D. Tóth, editors, *Proceedings of the 27th International Symposium on Graph Drawing and Network Visualization (GD'19)*, volume 11904 of *Lecture Notes in Computer Science*, pages 230–243. Springer, 2019. doi:10.1007/978-3-030-35802-0_18.

3 André Bouchet. Reducing prime graphs and recognizing circle graphs. *Combinatorica*, 7(3):243–254, 1987.

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

5 Bruno Courcelle. Circle graphs and monadic second-order logic. *Journal of Applied Logic*, 6(3):416–442, September 2008. doi:10.1016/j.jal.2007.05.001.

6 William H. Cunningham. Decomposition of directed graphs. *SIAM Journal on Algebraic Discrete Methods*, 3(2):214–228, June 1982. doi:10.1137/0603021.

7 William H Cunningham and Jack Edmonds. A combinatorial decomposition theory. *Canadian Journal of Mathematics*, 32(3):734–765, 1980.

8 Giuseppe Di Battista and Roberto Tamassia. On-line graph algorithms with SPQR-trees. In M.S. Paterson, editor, *Proceedings of the 17th International Colloquium on Automata, Languages, and Programming*, volume 443 of *Lecture Notes in Computer Science*, pages 598–611. Springer, 1990.

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

10 Csaba P Gabor, Kenneth J Supowit, and Wen-Lian Hsu. Recognizing circle graphs in polynomial time. *Journal of the ACM (JACM)*, 36(3):435–473, 1989.

11 Tibor Gallai. Transitiv orientierbare Graphen. *Acta Mathematica Academiae Scientiarum Hungarica*, 18:25–66, 1967.

12 Emeric Gioan and Christophe Paul. Split decomposition and graph-labelled trees: characterizations and fully dynamic algorithms for totally decomposable graphs. *Discrete Applied Mathematics*, 160(6):708–733, 2012.

13 Emeric Gioan, Christophe Paul, Marc Tedder, and Derek Corneil. Practical and efficient circle graph recognition. *Algorithmica*, 69(4):759–788, August 2014. doi:10.1007/s00453-013-9745-8.

14 Emeric Gioan, Christophe Paul, Mark Tedder, and Derek Corneil. Practical and efficient split decomposition via graph-labelled trees. *Algorithmica*, 69(4):789–843, 2014. doi:10.1007/s00453-013-9752-9.

15 Wen Lian Hsu. $o(m{\cdot}n)$ algorithms for the recognition and isomorphism problems on circular-arc graphs. *SIAM Journal on Computing*, 24(3):411–439, 1995. doi:10.1137/S0097539793260726.

16 Vít Kalisz, Pavel Klavík, and Peter Zeman. Circle graph isomorphism in almost linear time. *CoRR*, abs/1908.09151, 2019. arXiv:1908.09151.

17 Pavel Klavík, Jan Kratochvíl, Tomasz Krawczyk, and Bartosz Walczak. Extending partial representations of function graphs and permutation graphs. In Leah Epstein and Paolo Ferragina, editors, *Proceedings of the 20th Annual European Symposium on Algorithms (ESA'12)*, volume 7501 of *Lecture Notes in Computer Science*, pages 671–682. Springer, 2012. doi:10.1007/978-3-642-33090-2_58.

18 Pavel Klavík, Jan Kratochvíl, Yota Otachi, Ignaz Rutter, Toshiki Saitoh, Maria Saumell, and Tomáš Vyskočil. Extending partial representations of proper and unit interval graphs. In *Algorithm Theory–SWAT 2014*, pages 253–264. Springer, 2014.

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

20 Pavel Klavík, Jan Kratochvíl, and Tomáš Vyskočil. Extending partial representations of interval graphs. In Mitsunori Ogihara and Jun Tarui, editors, *Theory and Applications of Models of Computation: 8th Annual Conference, TAMC 2011, Tokyo. Proceedings*, pages 276–285. Springer, 2011. doi:10.1007/978-3-642-20877-5_28.

21 Tomasz Krawczyk and Bartosz Walczak. Extending partial representations of trapezoid graphs. In Hans L. Bodlaender and Gerhard J. Woeginger, editors, *Proceedings of the 43rd International Workshop on Graph-Theoretic Concepts in Computer Science*, volume 10520 of *Lecture Notes in Computer Science*, pages 358–371. Springer, 2017. doi:10.1007/978-3-319-68705-6_27.

22 Saunders Mac Lane. A structural characterization of planar combinatorial graphs. *Duke Mathematical Journal*, 3(3):460–472, 1937.

23 Maurizio Patrignani. On extending a partial straight-line drawing. *Int. J. Found. Comput. Sci.*, 17(5):1061–1070, 2006. doi:10.1142/S0129054106004261.

24 Jeremy Spinrad. Recognition of circle graphs. *Journal of Algorithms*, 16(2):264–282, 1994.