Helly Circular-Arc Graph Isomorphism Is in Logspace

Johannes Köbler, Sebastian Kuhnert[*], and Oleg Verbitsky[**]

Humboldt-Universität zu Berlin, Institut für Informatik

Abstract. We present logspace algorithms for the canonical labeling problem and the representation problem of Helly circular-arc (HCA) graphs. The first step is a reduction to canonical labeling and representation of interval intersection matrices. In a second step, the Δ trees employed in McConnell's linear time representation algorithm for interval matrices are adapted to the logspace setting and endowed with additional information to allow canonization. As a consequence, the isomorphism and recognition problems for HCA graphs turn out to be logspace complete.

1 Introduction

A graph G is *circular-arc* if each vertex $v \in V(G)$ can be assigned an arc $\rho(v)$ on a circle such that two vertices are adjacent if and only if their arcs intersect. We call any such assignment ρ a *circular-arc representation* of G and the arc system $\rho(G) = \{\rho(v) \mid v \in V(G)\}$ a *circular-arc model* of G. G is *Helly circular-arc* (HCA) if G has a representation ρ such that the arcs of the vertices in every clique C of G have non-empty intersection. We call such a ρ an *HCA representation* and $\rho(G)$ an *HCA model* of G. In this article, we solve the *canonical representation problem* for HCA graphs in logspace. That is, we give a logspace algorithm that computes for any given HCA graph G an HCA representation ρ_G such that isomorphic HCA graphs G and H receive identical HCA models $\rho_G(G) = \rho_H(H)$. If the input graph G is not HCA, the algorithm will detect this.

Previous results. HCA graphs were introduced by Gavril under the name of Θ circular-arc graphs [Gav74]. Gavril gave an $O(n^3)$ time representation algorithm for HCA graphs. Hsu improved this to $O(nm)$ [Hsu95]. Recently, Joeris et al. gave a linear time algorithm [JLM$^+$11]. The fastest known isomorphism algorithm for HCA graphs is due to Curtis et al. and works in linear time [CLM$^+$13]. Chen gave a parallel AC^2 algorithm [Che96]. Note that, though a logspace algorithm can take time bounded by a polynomial of high degree, the logspace solvability implies that the problem can be solved even in logarithmic time by a CRCW PRAM with polynomially many processors.

[*] Supported by DFG grant KO 1053/7–1.

[**] Supported by DFG grant VE 652/1–1. On leave from the Institute for Applied Problems of Mechanics and Mathematics, Lviv, Ukraine.

For the special case of interval graphs (which are easily seen to be HCA), the linear time algorithms by Booth and Lueker for recognition [BL76] and isomorphism [LB79] have been known for many decades. Recently, these have been supplemented with a logspace algorithm for canonical representation of interval graphs [KKL+11].

Generalizing these results to the class of all circular-arc graphs remains a challenging problem. While the representation problem for this class is solved in linear time by McConnell [McC03], no polynomial-time isomorphism test for circular-arc graphs is currently known (see the discussion in [CLM+13], where a counterexample to the correctness of Hsu's $O(nm)$ time isomorphism algorithm [Hsu95] is given). The history of the isomorphism problem for circular-arc graphs is surveyed in more detail by Uehara [Ueh13].

This motivates the persistent interest in isomorphism algorithms for subclasses of circular-arc graphs. Besides HCA graphs, mainly proper circular-arc graphs and concave-round graphs have been studied. The isomorphism problem for these two classes can be solved in linear time [LSS08; CLM+13] and in logspace [KKV12].

Overview of our results. Our logspace algorithm for canonical representation of HCA graphs proceeds in several steps; see Fig. 1.

Hsu observed that the structure of certain circular-arc graphs G allows to prescribe the intersection structure of each pair of arcs in a circular-arc representation of G as `di` (disjoint), `cd` (contained), `cs` (contains), `cc` (circle cover), and `ov` (overlap) [Hsu95]. We store this information in the *neighborhood matrix* λ_G of G (for more details see Section 2).

The motivation for switching to the matrix λ_G is that flipping the arc of a vertex (i.e., exchanging its two start and end points) can be mimicked in λ_G by substituting some of its entries (details are given in Section 3). We show how to identify a subset $X \subseteq V(G)$ such that flipping the arcs of all vertices in X results in a matrix $\lambda_G^{(X)}$ that can be realized by an interval system. We choose X as an inclusion-maximal clique of G that is the common neighborhood of two vertices, and prove that at least one such clique can be found in logspace.

In order to compute an HCA representation for G from the matrix $\lambda_G^{(X)}$ we give a logspace algorithm solving the representation problem of interval matrices. Let an *intersection matrix* be a quadratic matrix $\mu = (\mu_{i,j})_{i \neq j \in V}$ with entries `di`, `cs`, `cd`, `cc`, `ov`. We call the elements of V the *vertices* of μ and we assume

$$\begin{tikzcd}
\text{colored } \Delta \text{ tree} & \Delta(\lambda) \arrow[r, "\text{tree}", "\text{canonization}"'] & \widehat{\Delta(\lambda)} \arrow[d] & \text{canonical } \Delta \text{ tree} \\
\text{interval matrix} & \lambda \arrow[u] & \mathcal{I}_{\widehat{\lambda}} \arrow[d] & \text{interval model} \\
\text{HCA matrix} & \lambda_G \arrow[u] & \mathcal{A}_{\widehat{\lambda_G}} \arrow[d, equal] & \text{HCA model} \\
\text{HCA graph} & G \arrow[u] & \mathcal{A}_{\widehat{G}} & \text{HCA model}
\end{tikzcd}$$

Fig. 1. Overview of the canonical representation algorithm for HCA graphs

that V is linearly ordered. An intersection matrix μ is called *CA matrix* if it is possible to assign to each vertex $i \in V$ an arc $\rho(i)$ on a circle such that for any pair of vertices $i \neq j \in V$ the arcs $\rho(i)$ and $\rho(j)$ intersect in accordance with the entry $\mu_{i,j}$. A CA matrix μ is called an *HCA matrix* (*interval matrix*) if the arc system $\rho(V) = \{\rho(i) \mid i \in V\}$ is Helly (does not cover the whole circle, respectively).

Our logspace algorithm for computing a representation of a given interval matrix is described in Section 4. McConnell gave a linear time algorithm for this problem as part of his representation algorithm for circular-arc graphs [McC03]. He introduced the *Δ tree* of an interval matrix to capture all possible interval representations. Our key contribution here is to compute the Δ tree in logspace.

In Section 5, we show how to compute *canonical* representations of interval matrices. This is a significant extension of McConnell's algorithm, which only deals with representation. We implement this step as a reduction to colored tree canonization, which can be solved in logspace using Lindell's algorithm [Lin92].

2 Preliminaries

A *circular-arc system* $\mathcal{A}$ is a set of non-empty arcs on a circle. An *interval system* $\mathcal{I}$ is a set of non-empty intervals on a line. Equivalently, we can define an interval system as a circular-arc system $\mathcal{I}$ having the special property that there is at least one point on the circle that is not covered by any arc of $\mathcal{I}$. A set system $\mathcal{S}$ has the *Helly property* if every subsystem $\mathcal{S}' \subseteq \mathcal{S}$ with non-empty pairwise intersections has a non-empty overall intersection, i.e., $(\forall A, B \in \mathcal{S}' : A \cap B \neq \varnothing) \Rightarrow \bigcap_{A \in \mathcal{S}'} A \neq \varnothing$. It is easy to see that every interval system has the Helly property, but that there are non-Helly circular-arc systems; see Figure 2 (a) for an example. To keep notation concise, we use *CA* as a shorthand for circular-arc and *HCA* as an abbreviation of Helly circular-arc.

Two sets A and B *intersect* if $A \cap B \neq \varnothing$. They *overlap* (written $A \between B$) if additionally $A \setminus B \neq \varnothing$ and $B \setminus A \neq \varnothing$.

Given a set system $\mathcal{S}$, its *intersection graph* $\mathbb{I}(\mathcal{S})$ has one vertex for each set $A \in \mathcal{S}$, and two nodes $A, B \in \mathcal{S}$ are adjacent if and only if $A \cap B \neq \varnothing$. A graph G is a *CA graph* if there is a CA system $\mathcal{A}$ such that $G \cong \mathbb{I}(\mathcal{A})$. In this case, $\mathcal{A}$ is called a *CA model* of G, and an isomorphism $\rho \colon V(G) \to \mathcal{A}$ from G to $\mathbb{I}(\mathcal{A})$ is called a *CA representation* of G. HCA graphs and interval graphs are defined analogously, and so are their respective models and representations.

Given a graph G and a vertex $v \in V(G)$, let $N_G[v]$ denote the *closed neighborhood* of v, i.e., the set of vertices with distance at most 1 from v. The *common neighborhood* of two vertices $u, v \in V(G)$ is $N_G[u,v] = N_G[u] \cap N_G[v]$. If G is understood from the context, the index will be omitted. A vertex $v \in V(G)$ is *universal* if $N[v] = V(G)$. Two vertices $u, v \in V(G)$ are *twins* if $N[u] = N[v]$. A *twin class* is an inclusion-maximal set $U \subseteq V(G)$ such that all pairs of vertices in U are twins.

Let $\mu = (\mu_{i,j})_{i \neq j \in V}$ be a quadratic matrix. We call the elements of V the *vertices* of μ and we assume that V is linearly ordered. Another quadratic

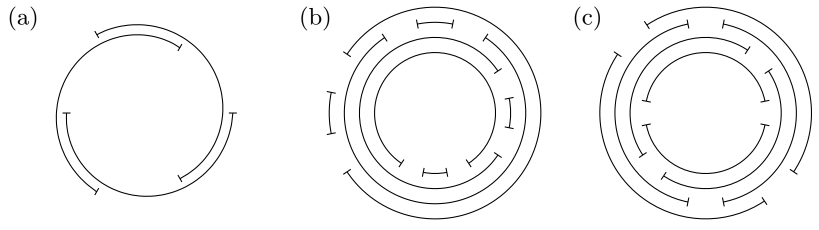

Fig. 2. (a) A non-HCA model of the HCA graph K_3. (b) Let G_n denote the split graph on $n + n$ vertices consisting of an n-clique C and a set S of n independent vertices, which are connected by the bipartite complement of a perfect matching between C and S. Every G_n is HCA; the figure shows an HCA model of G_4. Note that G_n has exactly $n + 1$ maxcliques, each of size n, and the maxclique C cannot be described as intersection or difference of less than n neighborhoods. (c) The complement graph H_n of n independent edges is CA. It has 2^n maxcliques C_i, each containing exactly one endpoint of each edge in $\overline{H_n}$. Since the common neighborhood of fewer than n vertices of H_n contains both endpoints of at least one edge in $\overline{H_n}$, no maxclique C_i can be described in this way. The figure shows a CA model of H_4.

matrix $\lambda = (\lambda_{i,j})_{i \neq j \in U}$ is *isomorphic* to μ (written $\lambda \cong \mu$) if there is a bijection $\sigma \colon U \to V$ such that $\lambda_{i,j} = \mu_{\sigma(i),\sigma(j)}$ for all $i \neq j \in U$. Note that two graphs are isomorphic if and only if their adjacency matrices are isomorphic.

An *intersection matrix* is a matrix $\mu = (\mu_{u,v})_{u \neq v \in V}$ with entries $\mu_{u,v} \in \{\mathtt{di}, \mathtt{cs}, \mathtt{cd}, \mathtt{cc}, \mathtt{ov}\}$ that satisfies (a) $\mu_{u,v} = \mathtt{cd} \Leftrightarrow \mu_{v,u} = \mathtt{cs}$ and (b) $\mu_{u,v} = \mu_{v,u}$ in all other cases. Our interest is in intersection matrices that describe the intersection types between the arcs of a CA system. The following notation was introduced in [LS09].

Definition 2.1. *Let $\mathcal{A}$ be a CA system such that no single arc $C \in \mathcal{A}$ covers the whole circle and the endpoints of all arcs $C \in \mathcal{A}$ are pairwise distinct. The* intersection matrix *$\mu_{\mathcal{A}} = (\mu_{A,B})_{A \neq B \in \mathcal{A}}$ of $\mathcal{A}$ is defined by the entries*

$$\mu_{A,B} := \begin{cases} \mathtt{di} & \text{if } A \cap B = \varnothing; \\ \mathtt{cd} & \text{if } A \subsetneq B; \\ \mathtt{cs} & \text{if } A \supsetneq B; \\ \mathtt{cc} & \text{if } A \between B \text{ and } A \text{ and } B \text{ jointly cover the circle}; \\ \mathtt{ov} & \text{if } A \between B \text{ but } A \text{ and } B \text{ do not jointly cover the circle.} \end{cases}$$

The intersection matrix $\mu_{\mathcal{I}}$ of an interval system $\mathcal{I}$ with pairwise distinct endpoints is defined similarly, using only the entries di*,* cd*,* cs *and* ov *(for $A \between B$). A matrix μ is a* CA matrix *if there is a CA system $\mathcal{A}$ such that $\mu \cong \mu_{\mathcal{A}}$.* HCA matrices *and* interval matrices *are defined analogously.*

Definition 2.2. *Given a graph G, its* neighborhood matrix $\lambda_G = (\lambda_{u,v})_{u \neq v \in V(G)}$ *is defined by the entries*

$$\lambda_{u,v} := \begin{cases} \texttt{di} & \textit{if } \{u,v\} \notin E(G); \\ \texttt{cd} & \textit{if } N[u] \subsetneq N[v]; \\ \texttt{cs} & \textit{if } N[u] \supsetneq N[v]; \\ \texttt{cc} & \textit{if } N[u] \between N[v], N[u] \cup N[v] = V, \\ & \textit{and } \forall w \in N[u] \setminus N[v] : N[w] \subset N[u], \\ & \textit{and } \forall w \in N[v] \setminus N[u] : N[w] \subset N[v]; \\ \texttt{ov} & \textit{otherwise.} \end{cases}$$

Note that λ_G can be viewed as an augmented adjacency matrix, as 0 entries correspond to `di` and 1 entries are subdivided into four different categories. The *underlying graph* of an intersection matrix $\mu = (\mu_{u,v})_{u \neq v \in V}$ is denoted by G_μ and consists of the vertices V and the edges $\big\{\{u,v\} \bigm| \mu_{u,v} \neq \texttt{di}\big\}$.

Following [Hsu95], we call a CA representation $\rho \colon V(G) \to \mathcal{A}$ of G *normalized* if ρ is an isomorphism between the neighborhood matrix λ_G and the CA matrix $\mu_{\mathcal{A}}$. Hsu provides an algorithm that transforms any CA representation of a CA graph with certain properties into a normalized representation, obtaining the following.

Lemma 2.3 ([Hsu95]). *Any CA graph G without twins and universal vertices has a normalized CA representation.*

All normalized CA representations have a property that is called *stable* by Joeris et al. who prove that every stable CA representation of an HCA graph G yields an HCA model [JLM+11, Theorem 4.1]. This implies the following.

Lemma 2.4. *Any normalized CA representation of an HCA graph G without twins and universal vertices provides an HCA model for G.*

As usual, L is the class of all languages decidable by Turing machines with a read-only input tape using only $O(\log N)$ space on the working tapes, where N is the input size. FL is the class of all functions computable by such machines that additionally have a write-only output tape. Note that FL is closed under composition: To compute $f(g(x))$ for $f, g \in \mathsf{FL}$, simulate the Turing machine for f and keep track of the position of its input head. Every time this simulation needs a character from f's input tape, simulate the Turing machine for g on input x until it outputs the required character. Note also that g can first output a copy of its input x and afterwards compute additional information to be used by f. This construction can be iterated a constant number of times, still preserving the logarithmic space bound. This closure property allows us to present logspace algorithms in a modular way.

Lemma 2.5. *There is a logspace reduction from the (canonical) HCA representation problem for HCA graphs G to the (canonical) CA representation problem of vertex-colored HCA matrices.*

Proof. First consider the case that G is twin-free and has no universal vertex. Compute the neighborhood matrix λ_G. By Lemma 2.3, λ_G admits a normalized CA representation ρ. Any such ρ is Helly by Lemma 2.4, and easily seen to be also a HCA representation of G. If ρ is canonical for (colored) matrices, it is also canonical for (colored) graphs, as $G \cong H$ is equivalent to $\lambda_G \cong \lambda_H$.

If the input graph G contains twins, apply the above algorithm to the graph G' that is obtained from G by removing all but one vertex from each twin class. Let ρ' be the computed representation of G'. Then define a representation ρ of G by $\rho(v) = \rho'(\hat{v})$, where $\hat{v}$ is the representative of the twin class of v that is present in G'. To preserve canonicity, color each vertex of G' with the number of twins it stands for; then $G \cong H$ is equivalent to $G' \cong H'$. It remains to observe that universal vertices can be likewise removed before computing the neighborhood matrix, with arcs added for them afterwards. ☐

3 Transforming HCA Matrices into Interval Matrices

In this section we describe a logspace reduction of the (canonical) CA representation problem for HCA matrices to the (canonical) representation problem of interval matrices. Note that it suffices for our purposes to obtain *any* (not necessarily Helly) representation: Lemma 2.4 implies that the representation is Helly if the HCA matrix is the neighborhood matrix of some HCA graph G.

Following McConnell, we can transform a CA system $\mathcal{A}$ into an interval system $\mathcal{A}^{(X)} = \{C \in \mathcal{A} \mid x \notin C\} \cup \{\tilde{C} \mid C \in \mathcal{A}, x \in C\}$ by choosing any point x on the circle that is different from all endpoints of $\mathcal{A}$ and flipping all the arcs in the set $X = \{C \in \mathcal{A} \mid x \in C\}$. *Flipping* an arc C just means that we replace it with the arc $\tilde{C}$ having the same endpoints as C but covering the opposite part of the circle. McConnell observed that flipping arcs of $\mathcal{A}$ corresponds to the replacements in the CA matrix $\mu_{\mathcal{A}}$ (cf. Definition 2.1) that are given in Table 1. Denote the result of flipping a subset X of the vertices of a CA matrix λ as $\lambda^{(X)}$. Note that $\lambda^{(X)}$ will become an interval matrix if exactly the arcs that contain a point x are flipped.

Table 1. The effect of flipping arcs of a CA system $\mathcal{A}$ on the entries of its CA matrix $\mu_{\mathcal{A}} = (\mu_{A,B})_{A \neq B \in \mathcal{A}}$

$\mu_{A,B}$	di	cd	cs	cc	ov
$\mu_{\tilde{A},B}$	cs	cc	di	cd	ov
$\mu_{A,\tilde{B}}$	cd	di	cc	cs	ov
$\mu_{\tilde{A},\tilde{B}}$	cc	cs	cd	di	ov

The rules described in Table 1 can be applied to any CA matrix $\mu = (\mu_{i,j})_{i \neq j \in V}$. To ensure that the resulting matrix $\mu^{(X)}$ is interval, a suitable vertex set $X \subseteq V$ has to be used. If we don't have a CA representation of μ, the set X has to be identified only from the structure of μ. If μ is HCA, the underlying graph G_μ is also HCA. The following fact implies that any inclusion-maximal clique (*maxclique* for short) C of G_μ can be used as X in this case.

Fact 3.1 *Let $\rho\colon V(G) \to \mathcal{A}$ be any HCA representation of a graph G and let C be any maxclique of G. Then there is a point x in the HCA model $\mathcal{A}$ such that no arc has x as its endpoint and $\{\rho(v) \mid v \in C\} = \{A \in \mathcal{A} \mid x \in A\}$.*

Proof. As C is a clique, the arcs in $\rho(C) = \{\rho(v) \mid v \in C\}$ intersect pairwise. As $\mathcal{A}$ is Helly, $\bigcap_{v\in C} \rho(v)$ is non-empty. By maximality of C, no further arc can contain any point x in this intersection. □

In an interval graph, all maxcliques can be characterized as the common neighborhood of two vertices. This property was used in [KKL+11] to reduce the canonical representation problem of interval graphs to that of interval hypergraphs. The same approach is not possible for HCA graphs, as they may contain maxcliques that cannot be characterized as the intersection or difference of constantly many neighborhoods; see Fig. 2 (b) for an example. However, at least one maxclique can be found in this way.

Theorem 3.2. *Let G be an HCA graph. Then there are $u, v \in V(G)$ (possibly $u = v$) such that $N[u, v]$ is a maxclique.*

We remark that general CA graphs do not necessarily have such a maxclique, see Fig. 2 (c) for an example.

Proof. Let $\lambda_G = (\lambda_{u,v})_{u\neq v\in V(G)}$ be the neighborhood matrix and $\rho\colon V(G) \to \mathcal{A}$ a normalized HCA representation of G. In order to find two vertices $u, v \in V(G)$ such that $N[u, v]$ is a maxclique, we start with an arbitrary vertex v such that there is no vertex w with $\lambda_{v,w} = \texttt{cs}$ (i.e., $\nexists w : N[w] \subsetneq N[v]$). Note that there cannot be a vertex w' with $\lambda_{v,w'} = \texttt{cc}$, since this would imply that there is a vertex $w \in N[v] \setminus N[w']$ (because $N[w'] \between N[v]$) with $N[w] \subsetneq N[v]$ (we can rule out equality because $w' \in N[v] \setminus N[u']$).

In case there is no vertex w with $\lambda_{v,w} = \texttt{ov}$, $N[v]$ is a maxclique. This follows since $\lambda_{v,w} = \texttt{cd}$ for all $w \in N(v)$ and hence, for all $w, w' \in N[v]$ it holds that $w \in N[v] \subseteq N[w']$.

Otherwise, we choose a vertex $u \in N[v]$ with $\lambda_{v,u} = \texttt{ov}$, such that $N[u, v]$ is minimal w.r.t. inclusion and claim that $N[u, v]$ is a maxclique. In order to derive a contradiction assume that there exist $w, w' \in N[u, v]$ such that $w \notin N[w']$. If $\lambda_{v,w} = \texttt{cd}$ (or $\lambda_{v,w'} = \texttt{cd}$) then it follows that $w' \in N[v] \subseteq N[w]$ (or $w \in N[v] \subseteq N[w']$), a contradiction.

If $\lambda_{v,w} = \lambda_{v,w'} = \texttt{ov}$ then $\rho(w) \cap \rho(w') = \varnothing$ and $\rho(w) \between \rho(v) \between \rho(w')$. Since $\rho(u) \between \rho(v)$ it follows that $\rho(u)$ overlaps $\rho(v)$ from the same side as one of $\rho(w)$ and $\rho(w')$, say $\rho(w)$. Because of $w' \in N[u, v] \setminus N[w]$ and the Helly property, it follows that $\rho(u) \cap \rho(v) \cap \rho(w') \neq \varnothing$ but $\rho(w) \cap \rho(v) \cap \rho(w') = \varnothing$, implying that $\rho(v) \cap \rho(w) \subseteq \rho(v) \cap \rho(u)$. Using again the Helly property, it now follows for any $x \in N[w, v]$ that $\rho(v) \cap \rho(w) \cap \rho(x) \neq \varnothing$ which in turn implies that $\rho(v) \cap \rho(u) \cap \rho(x) \neq \varnothing$. Hence, we get the inclusion $N[w, v] \subseteq N[u, v]$, contradicting the choice of u, since $w' \in N[u, v] \setminus N[w, v]$. □

Theorem 3.3. *The (canonical) CA representation problem for vertex-colored HCA matrices can be reduced in logspace to the (canonical) representation problem for vertex-colored interval matrices.*

Proof. Given an HCA matrix $\mu = (\mu_{u,v})_{u,v \in V}$, the algorithm works as follows.

1. Find all pairs $u, v \in V$ such that $N[u, v]$ is a maxclique in G_μ (allowing $u = v$). By Theorem 3.2 at least one such pair exists. Denote the set of all maxcliques that are found in this way by $\mathcal{M}$.
2. For each $M \in \mathcal{M}$: Compute the interval matrix $\mu^{(M)}$ and mark the flipped vertices with a new color. Compute a (canonical) interval representation of $\mu^{(M)}$ and flip back all colored arcs, obtaining a CA representation $\rho_{\mu,M}$ of μ.
3. Among the $\rho_{\mu,M}$ computed in the previous step, choose ρ_μ as one that results in a lexicographically least CA model $\rho_{\mu,M}(\mu)$. Output ρ_μ.

To see that ρ_μ is canonical if the interval representation of $\mu^{(M)}$ is canonical, consider any isomorphic copy $\mu' = \varphi(\mu)$. Clearly, $M = N_{G_\mu}[u, v]$ is a maxclique of G_μ if and only if $M' = N_{G_{\mu'}}[\varphi(u), \varphi(v)] = \varphi(M)$ is a maxclique of $G_{\mu'}$. Note that φ is also an isomorphism between $\mu^{(M)}$ and $\mu'^{(M')}$. Thus the latter two receive the same colored canonical interval model $\mathcal{I}$. As the CA models $\rho_{\mu,M}(\mu)$ and $\rho_{\mu',M'}(\mu')$ are both obtained from $\mathcal{I}$ by flipping back the colored arcs, it follows that $\rho_{\mu,M}(\mu) = \rho_{\mu',M'}(\mu')$. Hence, μ and μ' receive the same CA model $\rho_\mu(\mu) = \min\{\rho_{\mu,M}(\mu) \mid M \in \mathcal{M}\} = \min\{\rho_{\mu',\varphi(M)}(\mu') \mid M \in \mathcal{M}\} = \rho_{\mu'}(\mu')$. □

4 Finding Representations of Interval Matrices in Logspace

McConnell [McC03] showed how to find interval representations of interval matrices in linear time. In this section, we apply some of his techniques to solve this task in logspace.

Given an intersection matrix $\lambda = (\lambda_{u,v})_{u \neq v \in V}$, define $G_{\texttt{ov,di}}$ as the undirected graph on the vertex set V with edges $\{u, v\}$ for each pair with $\lambda_{u,v} \in \{\texttt{ov}, \texttt{di}\}$. Similarly, define $D_{\texttt{cd}}$ (resp. $D_{\texttt{cs}}$) as the directed graph on V with arrows (u, v) for each pair with $\lambda_{u,v} = \texttt{cd}$ (resp. $\lambda_{u,v} = \texttt{cs}$).

A *transitive orientation* of an undirected graph is an assignment of directions to all edges such that the resulting set of arrows is transitive. An *interval orientation* of an intersection matrix λ is a transitive orientation $D_{\texttt{ov,di}}$ of $G_{\texttt{ov,di}}$ that remains transitive when restricted to $G_{\texttt{di}}$ and that satisfies

$$\lambda_{u,v} = \texttt{di} \wedge \lambda_{u,w} = \lambda_{v,w} = \texttt{ov} \Rightarrow \text{either } (u, w) \in D_{\texttt{ov,di}} \text{ or } (v, w) \in D_{\texttt{ov,di}} \quad (1)$$

The last condition requires that if w stays in overlap relation with two disjoint vertices u, v, then w has to be arranged in between u and v. Any interval representation ρ of λ induces an interval orientation of λ: An edge $\{u, v\}$ of $G_{\texttt{ov,di}}$ is oriented as (u, v) if and only if $\rho(u) < \rho(v)$, i.e., if the interval $\rho(u)$ starts left of the interval $\rho(v)$. The following lemma shows the converse, implying that interval orientations are in 1-1 correspondence with interval representations (provided that we fix the set of endpoints as $\{0, \ldots, 2n - 1\}$).

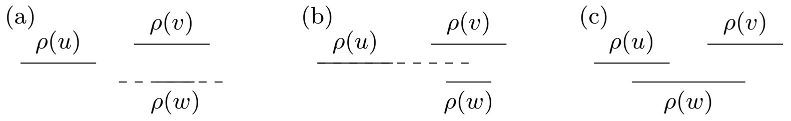

Fig. 3. In all three cases of Definition 4.2 there is no way to place $\rho(u)$ between $\rho(v)$ and $\rho(w)$.

Lemma 4.1. *Let λ be an interval matrix, and let $D_{\texttt{ov},\texttt{di}}$ be an interval orientation of λ. Then there exists an interval representation ρ of λ that induces the interval orientation $D_{\texttt{ov},\texttt{di}}$. Moreover, ρ is computable in logspace on input λ and $D_{\texttt{ov},\texttt{di}}$.*

Proof sketch. Let $\lambda = (\lambda_{u,v})_{u \neq v \in V}$. To obtain ρ, order the left endpoints according to $D_{\texttt{ov},\texttt{di}} \cup D_{\texttt{cs}}$ and the right endpoints according to $D_{\texttt{ov},\texttt{di}} \cup D_{\texttt{cd}}$. Interleave these two linear orders such that the relationships in λ are obeyed. □

By Lemma 4.1 it suffices to compute an interval orientation $D_{\texttt{ov},\texttt{di}}$ of a given interval matrix λ to get an interval representation of λ.

Definition 4.2 (cf. [McC03, Definition 6.3]). *Let λ be an intersection matrix, and let $\{u, v\}$ and $\{u, w\}$ be edges in $G_{\texttt{ov},\texttt{di}}$. The binary relation Δ contains the entries $(u, v)\Delta(u, w)$ and $(v, u)\Delta(w, u)$ if one of the following holds:*
(a) $\lambda_{u,v} = \lambda_{u,w} = \texttt{di}$ *and* $\lambda_{v,w} \neq \texttt{di}$
(b) $\lambda_{u,v}, \lambda_{u,w} \in \{\texttt{ov}, \texttt{di}\}$ *and* $\lambda_{v,w} \in \{\texttt{cd}, \texttt{cs}\}$
(c) $\lambda_{u,v} = \texttt{di}$ *and* $\lambda_{u,w} = \lambda_{v,w} = \texttt{ov}$

If any of these three condition holds true, then in any interval representation ρ of λ, the intervals $\rho(v)$ and $\rho(w)$ must be on the same side of $\rho(u)$; see Fig. 3. In other words, any interval orientation $D_{\texttt{ov},\texttt{di}}$ of λ must contain (u, v) if and only if it contains (u, w). This is the rationale for the following definition: *Δ implication classes* are the equivalence classes of the symmetric transitive closure of Δ. The union of a Δ implication class and its transpose is called *Δ color class* and can be viewed as a set of (undirected) edges in $G_{\texttt{ov},\texttt{di}}$.

Lemma 4.3 ([McC03, Theorem 6.4]). *Each interval orientation of λ contains exactly one Δ implication class from each Δ color class.*

This implies that (u, v) and (v, u) cannot be in the same Δ implication class. However, not any selection of one Δ implication class from each Δ color class yields an interval orientation of λ. To find a valid selection, we need to consider the *Δ tree* of λ.

A *module* of a matrix $\lambda = (\lambda_{u,v})_{u \neq v \in V}$ is a subset $U \subseteq V$ that is not distinguished by any vertex outside U, i.e., for any $u \neq v \in U$ and $w \in V \setminus U$ it holds $\lambda_{u,w} = \lambda_{v,w}$ and $\lambda_{w,u} = \lambda_{w,v}$. McConnell [McC03] calls a module U of an intersection matrix λ a *Δ module*, if it is a clique in the corresponding intersection graph (i.e., $\lambda_{u,v} \neq \texttt{di}$ for all $u \neq v \in U$) or if there is no $v \in V \setminus U$ such that $\lambda_{v,u} = \texttt{ov}$ for all $u \in U$. The Δ modules of an intersection matrix form a tree decomposable family [McC03, Definition 6.7 and Theorem 6.9]. The resulting decomposition tree, i.e., the transitive reduction of the containment relation among strong Δ modules U (i.e., U does not overlap any other Δ module), is called *Δ tree* of λ. The leaves of the Δ tree are trivial modules consisting of single vertices. An inner node in the Δ tree is called *degenerate* if taking the union of any of its children gives a Δ module, and *prime* otherwise. If U is an inner node in the Δ tree and $W_1, \ldots, W_k$ are its children, the *quotient of λ at U* is the submatrix $\lambda[U]$ of λ on the vertices $W = \{w_1, \ldots, w_k\}$ with $w_i \in W_i$. As the W_i are disjoint modules, $\lambda[U]$ does not depend on the actual choice of the w_i. In the quotient matrix of a degenerate node, its children are either in pairwise `ov`, in pairwise `di`, or in pairwise `cd`/`cs` relation [McC03, p. 110]. Hence, the inner nodes of the Δ tree can be classified as prime, disjoint, overlap or containment nodes.

The following results from [McC03] show that the Δ tree provides a compact representation of all possible interval orientations of λ.

Lemma 4.4 ([McC03, Lemma 6.14]). *The set of vertices spanned by a Δ color class in an interval intersection matrix λ is a Δ module of λ.*

Lemma 4.5 ([McC03, Theorem 6.15]). *A set of edges of $G_{\mathtt{ov},\mathtt{di}}$ is a Δ color class if and only if it is the set of edges of $G_{\mathtt{ov},\mathtt{di}}$ connecting all children of a prime node or a pair of children of a degenerate node in the Δ tree.*

Lemma 4.6 ([McC03, Theorem 6.19]). *Any acyclic union of Δ implication classes gives an interval orientation of λ.*

The next lemma reduces the problem of computing an interval orientation of λ to the problem of computing interval orientations of the quotient matrices of the inner nodes of the Δ tree.

Lemma 4.7. *Let λ be an intersection matrix, and let $U_1, \ldots, U_c$ be the inner nodes of its Δ tree. Any sequence of interval orientations $D_1, \ldots, D_c$ for the quotient matrices $\lambda[U_1], \ldots, \lambda[U_c]$ induces an interval orientation D of λ, which can be computed in logspace.*

Proof. By Lemma 4.5, there is a one-one correspondence between the Δ color classes and certain subsets of children of the nodes $U_1, \ldots, U_c$ (more precisely, any pair of children if U_a is degenerate and all children if U_a is prime). For each node U_a, the interval orientation D_a of $\lambda[U_a]$ picks exactly one of the two Δ implication classes in all the color classes associated with its children. Define D as the union of these implication classes; this construction is clearly possible in logspace. By Lemma 4.6 it suffices to show that D is acyclic. This can be seen as follows: By Lemma 4.1, the interval orientations D_a induce interval representations ρ_a of the quotient matrices. For each directed edge $(u, v) \in D$, let $U_{a(u,v)}$ denote the Δ tree node that has u and v in different children. Then $(u, v) \in D_{a(u,v)}$ and thus $\rho_{a(u,v)}(u) < \rho_{a(u,v)}(v)$. Hence, any edge (u, v) leads further right in the smallest Δ tree node that contains it. This implies that D is acyclic. ◻

As soon as we have the Δ tree, it's very easy to compute interval orientations for the quotient matrices corresponding to its inner nodes U. If U is prime, we can take any of the two implication classes of the color class connecting all its children. If U is degenerate of type overlap or disjoint, any linear ordering of its children provides an interval orientation for its quotient matrix. Finally, if U is of type containment, no edges have to be oriented.

Theorem 4.8. *The Δ implication classes, the Δ color classes, and the Δ tree of a given intersection matrix λ can be computed in logspace.*

Proof. The Δ implication classes (and thus the Δ color classes) can be found in logspace using Reingold's reachability algorithm on an auxiliary graph that has the pairs (u, v) with $\lambda_{u,v} \in \{\mathtt{ov}, \mathtt{di}\}$ as vertices and the symmetric closure of the Δ relation (Definition 4.2) as edges.

To compute the Δ tree, consider the overlap graph O having as its nodes the vertex sets spanned by the Δ color classes and the overlap relation on these nodes as its edge set. By Lemma 4.4, the nodes of O are Δ modules. Also, the connected components of O correspond to Δ modules, as these form a tree decomposable family.

The prime Δ modules and the degenerate Δ modules of type overlap and disjoint correspond to the connected components of O as they are not overlapped by another Δ module and hence form strong Δ modules. Note that the prime Δ modules and the degenerate Δ modules of type overlap and disjoint having only two children appear as isolated nodes in O. The leaf nodes of the Δ tree are just the vertices of λ.

It remains to compute the degenerate nodes of type containment. We first argue that every containment node is the union of certain non-containment nodes (which are already computed). Indeed, consider any containment node U in the Δ tree. The containment structure of U induces a linear order on its children $U_1, \ldots, U_c$, i.e., for all $u_a \in U_a$ and $u_b \in U_b$ with $a < b$ it holds that $\lambda_{u_a,u_b} = \mathtt{cd}$ (and $\lambda_{u_b,u_a} = \mathtt{cs}$). Notice that no U_a can be a containment node: Otherwise, let $U'_1 \ldots, U'_{c'}$ be the children of U_a, again ordered by containment. Consider the sequence $U_1, \ldots, U_{a-1}, U'_1, \ldots, U'_{c'}, U_{a+1}, \ldots, U_c$. It is not hard to see that the union of the sets in any consecutive subsequence is a Δ module, and some of these would overlap U_a, a contradiction.

To compute the containment nodes of the Δ tree, define an auxiliary graph C with all non-containment nodes as vertices, putting edges between two nodes U_1, U_2 if

$$\forall u_1 \in U_1, u_2 \in U_2, u_3 \notin U_1 \cup U_2 : \lambda_{u_1,u_2} \in \{\mathtt{cd}, \mathtt{cs}\} \wedge \lambda_{u_1,u_3} = \lambda_{u_2,u_3} \,.$$

This results in edges between children of containment nodes that are adjacent in the sequence mentioned above. The connected components of C (which are paths) correspond to the containment nodes, which can thus be computed in logspace.

Finally, compute the edges of the Δ tree as the transitive reduction of the containment relation among the nodes. ☐

By combining Theorem 4.8 with Lemma 4.1 we obtain the following result.

Corollary 4.9. *Given an intersection matrix λ, an interval representation for it can be computed in logspace.*

5 Finding Canonical Representations for Interval Matrices

In this section, we describe a logspace algorithm for computing a canonical representation of a given interval matrix λ. The main task is to choose between the different possible interval orientations of the quotient matrices corresponding to the inner nodes of the Δ tree. By providing the Δ tree with additional information we can reduce this task to (colored) tree canonization.

Lemma 5.1. *Given an interval matrix λ and its Δ tree T', for each inner node U of T' the following can be computed in logspace:*
- *The quotient $\lambda[U]$ of λ at U.*
- *All possible interval models of $\lambda[U]$ (either only one, or two that are the reverse of each other).*
- *For each interval model M_U of $\lambda[U]$, the possible correspondences of the children of U to the intervals in M_U. This can either be arbitrary, a fixed mapping or one of two fixed mappings.*

Proof. Computing $\lambda[U]$ is clearly possible in logspace.

If U is a prime node, the edges of $G_{\mathtt{ov},\mathtt{di}}$ between the children of U are in the same Δ color class by Lemma 4.5. Thus each of the two corresponding Δ implication classes, when restricted to edges present in $\lambda[U]$, defines an orientation of all $G_{\mathtt{ov},\mathtt{di}}$ edges in $\lambda[U]$. By Lemma 4.3, this must be an interval orientation if λ is interval. By Lemma 4.1, it can be converted to an interval representation of $\lambda[U]$. As the two Δ implication classes are the transpose of each other, the two resulting interval models of $\lambda[U]$ are the reverse of each other. If they are the same, U is called *symmetric* and two mappings of the children of U to the intervals in the model are possible; otherwise only one.

If U is a degenerate node, the model of $\lambda[U]$ is uniquely determined by the node type and the number of children. If U is of type overlap or of type disjoint, the mapping of the children of U to the intervals is arbitrary. If U is of type containment, the mapping from the children to the intervals is fixed. These constructions are clearly possible in logspace. □

Definition 5.2. *Given an intersection matrix λ, the* colored Δ tree $\mathbb{T}(\lambda)$ *has the same nodes as the Δ tree (i.e., the strong Δ modules of λ that are not overlapped by another Δ module), plus three additional nodes* $\mathrm{lo}_U, \mathrm{mi}_U, \mathrm{hi}_U$ *for each inner node U that admits exactly two assignments of its children to the interval model of its quotient matrix (cf. Lemma 5.1); these nodes are inserted between U and its children. Each Δ tree node U receives a tuple (p_U, M_U) as color, where M_U is the interval model of the quotient $\lambda[U]$ given by Lemma 5.1* *(if there are two different models, take the smaller one), and p_U is the* position *of U among the children of its parent: If U is the root or if the parent of U admits an arbitrary mapping of its children to its quotient intervals, let $p_U = 0$. If the parent of U has a fixed assignment of its children to its intervals, let p_U be the position of the interval corresponding to U among the other intervals. If the parent of U allows two assignments of its children, let $p_{U,1}$ and $p_{U,2}$ be the positions of U under the two assignments, respectively. If $p_{U,1} < p_{U,2}$, make U a child of lo_U and define $p_U = (p_{U,1}, p_{U,2})$; if $p_{U,1} = p_{U,2}$, make U a child of mi_U and define $p_U = (p_{U,1}, p_{U,2})$; if $p_{U,1} > p_{U,2}$, make U a child of hi_U and define $p_U = (p_{U,2}, p_{U,1})$. Finally, color all lo_U and hi_U nodes with 0 and all mi_U nodes with 1.*

By Theorem 4.8 and Lemma 5.1, $\mathbb{T}(\lambda)$ can be computed in logspace.

Lemma 5.3. *If λ and λ' are isomorphic interval matrices, then $\mathbb{T}(\lambda) \cong \mathbb{T}(\lambda')$.*

Proof. Let φ be an isomorphism between λ and λ', i.e., $\lambda_{u,v} = \lambda'_{\varphi(u),\varphi(v)}$ for all u and v. Then φ induces a bijection between the Δ modules of λ and the Δ modules of λ' that gives an isomorphism between the Δ trees. Additionally, for each node U in the Δ tree of λ, the quotient matrices are isomorphic, i.e., $\lambda[U] \cong \lambda'[\varphi(U)]$, via the appropriate restriction of φ. This implies that the models chosen as colors are equal, i.e., $M_U = M_{\varphi(U)}$. It also implies equal positions $p_U = p_{\varphi(U)}$: This is immediate if U is the root, or if its parent is a degenerate node or a prime node with two different interval models. If the parent of U is a prime node with a symmetric interval model, the two linear orders on its children might be exchanged. Depending on whether this is the case, φ can either be extended with $\mathrm{lo}_U \mapsto \mathrm{lo}_{\varphi(U)}, \mathrm{mi}_U \mapsto \mathrm{mi}_{\varphi(U)}, \mathrm{hi}_U \mapsto \mathrm{hi}_{\varphi(U)}$ or with $\mathrm{lo}_U \mapsto \mathrm{hi}_{\varphi(U)}, \mathrm{mi}_U \mapsto \mathrm{mi}_{\varphi(U)}, \mathrm{hi}_U \mapsto \mathrm{lo}_{\varphi(U)}$ to obtain an isomorphism between $\mathbb{T}(\lambda)$ and $\mathbb{T}(\lambda')$. □

Lemma 5.4. *Let λ be an interval matrix. Given an isomorphic copy T' of $\mathbb{T}(\lambda)$, an isomorphic copy λ' of λ (that depends only on T') can be computed in logspace. When also given an isomorphism $\ell\colon \mathbb{T}(\lambda) \to T'$, an isomorphism $\varphi\colon \lambda \to \lambda'$ can be computed within the same space bound.*

Proof. Let V' be the leaves of T'; take V' as the vertices of λ' in the order they appear in the given representation of T'. As the leaves of $\mathbb{T}(\lambda)$ correspond to the vertices of λ, the tree isomorphism ℓ defines a mapping of the vertices V of λ to V'; take this mapping as φ.

Let U' be an inner node of T' that does not have color 0 or 1, i.e., that is not the image of a lo, mi or hi node. Let $U = \ell^{-1}(U')$ be the node of $\mathbb{T}(\lambda)$ that gets mapped to U'. The color of U' specifies an interval model M_U of the quotient of λ at U. Let $\lambda'[U']$ be the interval intersection matrix of M_U; it can easily be computed in logspace. By construction, $\lambda'[U']$ is isomorphic to the quotient $\lambda[U]$ of λ at U. It suffices to compute an assignment $a_{U'}$ of the children of U' to the intervals in M_U (and thus to the vertices of $\lambda'[U']$), such that for all children U_1, U_2 of U it holds that $\lambda[U]_{U_1,U_2} = \lambda'[U']_{a_{U'}(\ell(U_1)),a_{U'}(\ell(U_2))}$. Then, for any two leaves $u', v' \in V'$, let $U'(u', v')$ denote the least common ancestor of u' and v' in T'; the matrix entry $\lambda'_{u',v'}$ can be computed as the entry in $\lambda'[U'(u', v')]$ of the children of $U'(u', v')$ which are the ancestors of u' and v', respectively.

To compute the mapping $a_{U'}$, look at the positions of the children of U', which are available from their colors:

– If the children of U' all have position 0, then U is an overlap or disjoint node, and $\lambda'[U']$ contains pairwise `ov` or `di` entries, respectively. This implies that the assignment $a_{U'}$ of the children of U' to the intervals in M_U can be arbitrary. Define $a_{U'}$ so that it preserves the order of the children, i.e., $a_{U'}(U'_1) < a_{U'}(U'_2)$ whenever $U'_1 < U'_2$ in the given representation of T'.
– If the children of U' have pairwise different positions, then U is either a containment node or an asymmetric prime node. In either case, the positions allow to reconstruct the unique assignment: Choose $a_{U'}$ as the function that satisfies $a_{U'}(U'_1) < a_{U'}(U'_2)$ whenever the positions p_1 and p_2 of U'_1 and U'_2 satisfy $p_1 < p_2$.
– If U' has three children, of which two have color 0 and one has color 1, then let $\mathrm{mi}_{U'}$ be the child with color 1 and let $\mathrm{lo}_{U'}$ and $\mathrm{hi}_{U'}$ be the children with color 0 such that $\mathrm{lo}_{U'} < \mathrm{hi}_{U'}$ in the given representation of T'. Note that $\ell(\mathrm{mi}_U) = \mathrm{mi}_{U'}$ and $\ell(\{\mathrm{lo}_U, \mathrm{hi}_U\}) = \{\mathrm{lo}_{U'}, \mathrm{hi}_{U'}\}$. For the children of $\mathrm{lo}_{U'}$ and $\mathrm{mi}_{U'}$, use the first entry in the position tuple and for the children of $\mathrm{hi}_{U'}$ use the second entry in the position tuple and proceed as in the previous case. This results in one of the two possible assignments. $\square$

Theorem 5.5. *The canonical representation problem for interval matrices can be solved in logspace.*

Proof. The algorithm works as follows:

1. Compute the Δ tree of λ (see Theorem 4.8).
2. Compute interval models of the quotient matrices at the nodes of the Δ tree to obtain the colored Δ tree $\mathbb{T}(\lambda)$ (see Lemma 5.1 and Definition 5.2).
3. Compute a canonical labeling of $\mathbb{T}(\lambda)$ and use the algorithm of Lemma 5.4 to compute a canonical copy λ' of λ and a canonical labeling φ of λ.
4. Compute the Δ tree of λ' and interval orderings for the quotient matrices at its inner nodes (in fact, the information from $\mathbb{T}(\lambda)$ can be reused; only the assignment of children needs to be revisited). Combine these orientations into one for the whole matrix (see Lemma 4.7) and convert it into an interval representation ρ' of λ' (see Lemma 4.1). Combined with the canonical labeling φ of λ, this results in an interval representation $\rho = \rho' \circ \varphi$ of λ.

Note that λ' depends only on the canon of $\mathbb{T}(\lambda)$, so $\lambda_1 \cong \lambda_2$ implies $\lambda'_1 = \lambda'_2$. As ρ' depends only on λ', the resulting interval model $\rho(\lambda) = \rho'(\lambda')$ is canonical. $\square$

6 Conclusion

Our algorithms also allow recognition of HCA graphs: If the input graph does not belong to this class, either one of the steps will fail (e.g. finding a suitable maxclique M), or the resulting arcs will not be a representation of G (which can easily be checked), or the resulting arcs are not Helly. The latter can be checked in logspace using [JLM+11, Theorem 3.1].

We remark that by combining Theorem 3.3 and Corollary 4.9 we already get a logspace algorithm that computes for any given HCA graph G an HCA representation of G. Since any HCA representation of G allows to compute *all* maxcliques in logspace, we can reduce the canonical representation problem of HCA graphs to that of CA hypergraphs $\mathcal{H}_G$: the vertex set of $\mathcal{H}_G$ consists of all maxcliques of G and for each vertex $v \in V(G)$, $\mathcal{H}_G$ contains a hyperedge consisting of all maxcliques that contain v. It is known that a graph G is HCA if and only if $\mathcal{H}_G$ is a CA hypergraph [Gav74]. Moreover, the hypergraph $\mathcal{H}_G$ provides a canonical HCA model for G, if we order its maxcliques by a canonical circular ordering. Hence an alternative canonical representation algorithm for HCA graphs can be obtained by using the algorithm for computing a canonical CA model of $\mathcal{H}_G$ given in [KKV12]. However, we believe that finding canonical representations of interval matrices is of independent interest, as these allow additional constraints on the structure of the intervals compared to interval graphs. For a different kind of constraint, namely prescribing the lengths of pairwise intersections (and optionally interval lengths), both logspace and $O(nm)$ time (resp. linear time) algorithms are known [KKW12].

Bibliography

[BL76] K. S. Booth and G. S. Lueker. ‘Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms’. In: *J. Comput. Syst. Sci.* 13.3 (1976), pp. 335–379.

[Che96] L. Chen. ‘Graph isomorphism and identification matrices: Parallel algorithms’. In: *Trans. Paral. Distrib. Syst.* 7.3 (1996), pp. 308–319.

[CLM+13] A. R. Curtis, M. C. Lin, R. M. McConnell, Y. Nussbaum, F. J. Soulignac, J. P. Spinrad, and J. L. Szwarcfiter. ‘Isomorphism of graph classes related to the circular-ones property’. In: *Discrete Math. Theor. Comp. Sci.* 15.1 (2013), pp. 157–182.

[Gav74] F. Gavril. ‘Algorithms on circular-arc graphs’. In: *Networks* 4 (1974), pp. 357–369.

[Hsu95] W.-L. Hsu. ‘$O(m \cdot n)$ algorithms for the recognition and isomorphism problems on circular-arc graphs’. In: *SIAM J. Comput.* 24.3 (1995), pp. 411–439.

[JLM+11] B. L. Joeris, M. C. Lin, R. M. McConnell, J. P. Spinrad, and J. L. Szwarcfiter. ‘Linear time recognition of Helly circular-arc models and graphs’. In: *Algorithmica* 59.2 (2011), pp. 215–239.

[KKL+11] J. Köbler, S. Kuhnert, B. Laubner, and O. Verbitsky. ‘Interval graphs: Canonical representations in logspace’. In: *SIAM J. Comput.* 40.5 (2011), pp. 1292–1315.

[KKV12] J. Köbler, S. Kuhnert, and O. Verbitsky. ‘Solving the canonical representation and star system problems for proper circular-arc graphs in logspace’. In: *Proc. 32nd FSTTCS*. Dagstuhl: Leibniz-Zentrum für Informatik, 2012, pp. 387–399.

[KKW12] J. Köbler, S. Kuhnert, and O. Watanabe. ‘Interval graph representation with given interval and intersection lengths’. In: *Proc. 23rd ISAAC*. Berlin: Springer, 2012, pp. 517–526.

[LB79] G. S. Lueker and K. S. Booth. ‘A linear time algorithm for deciding interval graph isomorphism’. In: *J. ACM* 26.2 (1979), pp. 183–195.

[Lin92] S. Lindell. ‘A logspace algorithm for tree canonization. Extended abstract’. In: *Proc. 24th STOC*. 1992, pp. 400–404.

[LS09] M. C. Lin and J. L. Szwarcfiter. ‘Characterizations and recognition of circular-arc graphs and subclasses: A survey’. In: *Discrete Math.* 309.18 (2009), pp. 5618–5635.

[LSS08] M. C. Lin, F. J. Soulignac, and J. L. Szwarcfiter. ‘A simple linear time algorithm for the isomorphism problem on proper circular-arc graphs’. In: *Proc. 11th SWAT*. Berlin: Springer, 2008, pp. 355–366.

[McC03] R. M. McConnell. ‘Linear-Time Recognition of Circular-Arc Graphs’. In: *Algorithmica* 37.2 (2003), pp. 93–147.

[Ueh13] R. Uehara. ‘Tractabilities and intractabilities on geometric intersection graphs’. In: *Algorithms* 6.1 (2013), pp. 60–83.