A Simple Linear Time Algorithm for the Isomorphism Problem on Proper Circular-Arc Graphs

Min Chih Lin[⋆1], Francisco J. Soulignac[⋆⋆1] and Jayme L. Szwarcfiter [⋆⋆⋆2]

[1] Universidad de Buenos Aires, Facultad de Ciencias Exactas y Naturales, Departamento de Computación, Buenos Aires, Argentina.
[2] Universidade Federal do Rio de Janeiro, Instituto de Matemática, NCE and COPPE, Caixa Postal 2324, 20001-970 Rio de Janeiro, RJ, Brasil.
{oscarlin, fsoulign}@dc.uba.ar, jayme@nce.ufrj.br

Abstract. A circular-arc model $\mathcal{M} = (C, \mathcal{A})$ is a circle C together with a collection $\mathcal{A}$ of arcs of C. If no arc is contained in any other then $\mathcal{M}$ is a proper circular-arc model, and if some point of C is not covered by any arc then $\mathcal{M}$ is an interval model. A (proper) (interval) circular-arc graph is the intersection graph of a (proper) (interval) circular-arc model. Circular-arc graphs and their subclasses have been the object of a great deal of attention in the literature. Linear time recognition algorithms have been described both for the general class and for some of its subclasses. For the isomorphism problem, there exists a polynomial time algorithm for the general case, and a linear time algorithm for interval graphs. In this work we develop a linear time algorithm for the isomorphism problem in proper circular-arc graphs, based on uniquely encoding a proper circular-arc model. Our method relies on results about uniqueness of certain PCA models, developed by Deng, Hell and Huang in [6]. The algorithm is easy to code and uses only basic tools available in almost every programming language.

Keywords: isomorphism problems, proper circular-arc graphs, proper circular-arc canonization.

1 Introduction

Interval graphs, circular-arc graphs and its subclasses are interesting families of graphs that have been receiving much attention recently. Proper interval

⋆ Partially supported by UBACyT Grants X184 and X212 and by PICT ANPCyT Grant 1562

⋆⋆ Partially supported by UBACyT Grant X184, PICT ANPCyT Grant 1562 and by a grant of the YPF Foundation.

⋆⋆⋆ Partially supported by the Conselho Nacional de Desenvolvimento Científico e Tecnológico, CNPq, and Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro, FAPERJ, Brasil.

and proper circular-arc graphs are two of the most studied subclasses of interval and circular-arc graphs [4, 8, 21]. Booth and Lueker found the first linear time algorithm for recognizing interval graphs using a data structure called PQ-trees [3]. Since then a lot of effort has been put into simplifying this algorithm and avoiding the use of PQ-trees [5, 9, 10, 14]. For circular-arc graphs there is also a great amount of work focused into finding a simple linear time recognition algorithm [12]. The first linear time algorithm is due to McConnell [18] and is not simple to implement.

Algorithms for proper circular-arc recognition in linear time are also known, and they were always much easier to implement than those for the general case [6, 13]. Deng, Hell and Huang exploit their proper interval graph recognition algorithm to develop a linear time algorithm for proper circular-arc graphs, based on local tournaments [6]. They also described results about the uniqueness of connected proper circular-arc graphs and proper circular-arc models, that we use to build our isomorphism testing algorithm.

The isomorphism problem is a hard to solve NP problem, although it is not known whether it is NP-hard. Nevertheless, this problem is known to be polynomial or even linear for several classes of graphs [7]. For interval graphs, labeled PQ-trees can be used to test for isomorphism in linear time [17], while for circular-arc graph the best known algorithm runs in $O(mn)$ time [11].

In this work we present a simple algorithm for the isomorphism problem restricted to proper circular-arc graphs. This algorithm runs in $O(n)$ time, when a proper circular-arc model is given. The objective is to uniquely encode a "canonical" proper circular-arc model of the graph. This canonical model is obtained by rotating, reflecting and sorting each (co-)component of the input model.

Let $G = (V(G), E(G))$ be a graph, $n = |V(G)|$ and $m = |E(G)|$. Denote by $\overline{G}$ the *complement* of G. Graph G is *co-connected* when $\overline{G}$ is connected. A *(co-)component* is a maximal (co-)connected subgraph of G. For $v \in V(G)$, denote by $N(v)$ the set of vertices adjacent to v, and write $N[v] = N(v) \cup \{v\}$ and $\overline{N}(v) = V(G) \setminus N[v]$. A vertex v of G is *universal* if $N[v] = V(G)$.

A *circular-arc* (CA) model $\mathcal{M}$ is a pair $(C, \mathcal{A})$, where C is a circle and $\mathcal{A}$ is a collection of arcs of C. When traversing the circle C, we will always choose the clockwise direction, unless explicitly stated. If s, t are points of C, write (s, t) to mean the arc of C defined by traversing the circle from s to t. Call s, t the *extremes* of (s, t), while s is the *start point* and t the *end point* of the arc. The *extremes* of $\mathcal{A}$ are those of all arcs $A \in \mathcal{A}$. The *reverse model* of $\mathcal{M}$ is denoted by $\mathcal{M}^{-1}$, i.e. $\mathcal{M}^{-1}$ is the reflection of $\mathcal{M}$ with respect to some chord of the circle. Unless otherwise stated, we always assume that $\mathcal{A} = \{A_1, \ldots, A_n\}$ where $A_i = (s_i, t_i)$. Moreover, in a traversal of C the order in which the start points appear is $s_1, \ldots, s_n$. The set $s_i, \ldots, s_j$ $(t_i, \ldots, t_j)$, $1 \leq i < j \leq n$, is called an *s-range (t-range)* with $\emptyset$ being the empty s-range (t-range). Similarly, a set of contiguous extremes is an *st-range*, and $A_i, \ldots, A_j$ is a *range*. Without loss of generality, all arcs of C are considered as open arcs, no two extremes of distinct arcs of $\mathcal{A}$ coincide and no single arc entirely covers C.

When no arc of $\mathcal{A}$ contains any other, $(C, \mathcal{A})$ is a *proper* circular-arc (PCA) model. A (proper) interval model is a (proper) CA model where $\bigcup_{A \in \mathcal{A}} A \neq C$. A CA (PCA) graph is the intersection graph of a CA (PCA) model. A (proper) interval graph is the intersection graph of a (proper) interval model. We may use the same terminology used for vertices when talking about arcs (intervals). For example, we say an arc in a CA model is *universal* when its corresponding vertex in the intersection graph is universal. Similarly, a *connected* model is one whose intersection graph is connected.

Let Σ be an alphabet. A *string S (over Σ)* is a sequence $S(1), \ldots, S(|S|)$ where $|S|$ is the *length* of S. The set $\{1, \ldots, |S|\}$ is the set of *positions* of S. For positions $i < j$ we denote by $S[i; j]$ the substring $S(i), \ldots, S(j)$. If $<$ is a total order over Σ, then $<_{lex}$ denotes the *lexicographical order* of strings, i.e. $S <_{lex} T$ if and only if there exists $k < |T|$ such that $S(i) = T(i)$ for all $1 \leq i \leq k$ and either $k = |S|$ or $S(k+1) < T(k+1)$. The *rotation* $S \ll i$ is the string $S[i; |S|]$ followed by string $S[1; i-1]$. Position i is *canonical* if $S \ll i \leq_{lex} S \ll j$ for every $1 \leq j \leq |S|$. Observe that since $<$ is a total order, then $S \ll i = S \ll j$ for every pair i, j of canonical positions.

2 PCA representations

Let $\mathcal{M} = (C, \mathcal{A})$ be a PCA model of a graph G and fix some arc $A_i = (s_i, t_i) \in \mathcal{A}$. The *arc representation* $R^i(\mathcal{M})$ of $\mathcal{M}$ is a string obtained by transversing C from s_i and writing the character '$\mathrm{a_{j+1}}$' ('$\mathrm{b_{j+1}}$') when s_{i+j} (t_{i+j}) is reached. Thus, the j-th start (end) point that appears after s_i (t_i) is designated with the character '$\mathrm{a_{j+1}}$' ('$\mathrm{b_{j+1}}$'). Observe that we consider the order $s_1, \ldots, s_n$ to be fixed for $\mathcal{M}$, but in a computer program we do not have access to this order. What we have is an arc representation that allows us to gain access to that order.

It is clear that there are at most n different arc representations of $\mathcal{M}$, one for each arc A_i. Algorithms on (proper) CA graphs usually perform a linear time preprocessing on its input, given as an arc representation. For instance, by simply transversing the representation it is possible to build a data structure where t_i can be found in $O(1)$ time, given s_i.

Arc representations have a lot of redundant information for encoding PCA models. Fix some arc $A_i \in \mathcal{M}$. The *extreme sequence (of $\mathcal{M}$) from s_i* is the string $E^i(\mathcal{M})$ that is obtained by replacing '$\mathrm{a_j}$' ('$\mathrm{b_j}$') with 'a' ('b') in $R^i(\mathcal{M})$ for every $1 \leq j \leq n$. In other words, $E^i(\mathcal{M})$ is the string obtained by transversing C from s_i and writing the character 'a' ('b') when a start point (an end-point) is reached. The *mark point (of $\mathcal{M}$) from s_i* is the position $t^i(\mathcal{M})$ where '$\mathrm{b_1}$' appears in $R^i(\mathcal{M})$. Define the function r, such that $r(E^i(\mathcal{M}), t^i(\mathcal{M}))$ is the string obtained from $E^i(\mathcal{M})$ by replacing the j-th character 'a' with '$\mathrm{a_j}$' and the j-th character 'b' that appears from position $t^i(\mathcal{M})$ with character '$\mathrm{b_j}$'.

Remark 1. Function r is a bijection between $r(E^i(\mathcal{M}), t^i(\mathcal{M}))$ and $R^i(\mathcal{M})$. Moreover, r and r^{-1} can both be computed in $O(n)$ time.

Remark 2. For $1 \leq i, j \leq n$, $R^i(\mathcal{M}) = R^j(\mathcal{M})$ if and only if $E^i(\mathcal{M}) = E^j(\mathcal{M})$ and $t^i(\mathcal{M}) = t^j(\mathcal{M})$.

From now on we may not write the superscripts if we want to refer to any representation of $\mathcal{M}$. Also, when $\mathcal{M}$ is understood, we do not write it explicitly as a parameter. Let $\mathcal{M}$ be a PCA model. *Extreme representation* (E, t) uses only $O(n)$ bits while R uses $O(n \log n)$ bits. However, some operations, as taking the end point of some arc when the start point is given, are not (a-priori) fast enough when using (E, t). This is why in this work and others (see e.g. [1, 16]) arc representations are taken as the input of the algorithms. When we say that these algorithms run in $O(n)$ time when an arc representation is given what we mean is that they run in $O(n)$ time where n is the length of codification R. But if we instead use (E, t) as input, we have to build R first, so the algorithms take $O(n \log n)$ time where n is the length of (E, t). With this in mind, when we say that $\mathcal{M}$ is given as input, we mean that an arc representation (or some linear-time preprocessing of it) is given as input.

Let $\mathcal{M}$,$\mathcal{N}$ be two PCA models with n arcs. We write $\mathcal{M} =_M \mathcal{N}$ ($\mathcal{M}$ is *equal* to $\mathcal{N}$) if $R^i(\mathcal{M}) = R^j(\mathcal{N})$ for some $1 \leq i, j \leq n$. This is what one would intuitively assume as equality of models, i.e., do they have the extremes in the same order? Clearly, if two PCA models are equal then their intersection graphs are isomorphic, but the converse is not always true. Testing if two PCA models are equal can be trivially done in $O(n^2)$ time by fixing some $1 \leq i \leq n$ and then testing whether $R^i(\mathcal{M}) = R^j(\mathcal{N})$ for every $1 \leq j \leq n$. However this can also be verified in $O(n)$ time.

3 Basic algorithms

In this section we describe linear-time algorithms that we use several times in the paper. Below is the list of problems we need to solve throughout the paper:

- Is a PCA graph co-bipartite? If so, give a unique co-bipartition of its co-components.
- Compute the components of a PCA graph.
- Is the intersection graph of a PCA model an interval graph? If so, output a proper interval model [15].
- Find all canonical positions of a circular string [2, 20].

In the next subsections, we show how to solve each of the above problems and discuss the complexities of the corresponding algorithms. The proposed solutions are easy to implement. Furthermore, we believe they are of interest on their own.

3.1 Co-bipartitions of the co-components of a PCA graph

The first problem we need to solve is to determine whether a PCA graph is co-bipartite and, if so, output the co-bipartitions of its co-components. The following algorithm determines the co-component of a graph G, containing a given vertex $v \in V(G)$.

Algorithm 1 *Co-component containing v in a graph G*

```
1. Unmark all vertices and define $V_1^0 := \{v\}$, $V_2^0 := \emptyset$ and $k = 0$.
2. While there exists an unmarked vertex $w \in V_1^k \cup V_2^k$, perform the following
   operation. Let $i, j \in \{1, 2\}$, such that $v \in V_i^k$ and $j \neq i$. Mark $v$ and compute
   $V_i^{k+1} := V_i^k$, $V_j^{k+1} := V_j^k \cup \overline{N}(w)$ and $k := k + 1$.
3. Output $V_1 := V_1^k, V_2 := V_2^k$
```

Lemma 1. *$V_1 \cup V_2$ is the co-component containing v in G. Moreover, V_1, V_2 is a co-bipartition if and only if $V_1 \cap V_2 = \emptyset$.*

For the general case this algorithm can be implemented in $O(n^2)$ time. Next, we consider that G is a PCA graph given by a PCA model $\mathcal{M}$. Define a subset of $V(G)$ to be a *range* whenever their corresponding arcs in $\mathcal{M}$ form a range. The following lemma is relevant to our purposes.

Lemma 2. *At every step k, V_1^k and V_2^k are ranges.*

Proof. Clearly V_1^0 and V_2^0 are ranges. Now consider the k-th iteration and let i, j and $w \in V_i^k$ be as in Step 2. Since G is a PCA graph, $\overline{N}(w)$ is a range. If $\overline{N}(w) \cap V_j^k = \emptyset$, then $w = v$ and $k = 0$, thus $V_j^1 = \overline{N}(w)$ corresponds to a range. If $\overline{N}(w) \cap V_j^k \neq \emptyset$ then it follows that $k > 0$ and V_j^k is a range by the inductive hypothesis. Hence $\overline{N}(w) \cup V_j^k$ is also a range, because the union of two intersecting ranges is a range. □

Now we consider the complexity of the algorithm when a PCA ordering of the vertices is given. Let i, j and k be as in Step 2, $V_i = V_i^k$ and $V_j = V_j^k$. By Lemma 2, V_i is a range. The invariant we use is that V_i is partitioned into three ranges L_i, C_i and R_i, where L_i, C_i, and R_i appear in this order. The set V_j has an analogous partition L_j, C_j and R_j. The set of marked vertices of V_i is C_i, while $L_i \cup R_i$ is the set of unmarked vertices. To maintain the invariant, vertex w can be selected from V_i if and only if $C_i \cup \{w\}$ is also a range, thus there are at most four unmarked vertices that could be selected at each step. Suppose w is chosen from L_i. For the next iteration we have to modify each of the ranges to reflect the inclusion of $\overline{N}(w)$.

For the implementation, each range in the algorithm can be represented by a pair of integers, the *low index* and the *high index*, corresponding to the first and the last vertices of the range. Before applying the algorithm, range $\overline{N}(v)$ can be found for every vertex v in $O(n)$ time. Hence, $V_j \cup \overline{N}(w)$ in Step 2 can be computed in constant time. For this, the low index of L_j is updated to the smallest of the low indices of L_j and $\overline{N}(w)$. Similarly, for the high index of R_j we would choose the greatest of the two high indices. Finally, the mark of w is done by decreasing by one the high index of L_i and increasing by one the low index of C_i when $w \in L_i$. The case when $w \in R_i$ is analogous. With such an implementation, each iteration of Step 2 takes $O(1)$ time, thus each component C can be found in $O(|C|)$ time. That is in overall $O(n)$ time the algorithm finds the co-bipartitions of all co-components.

3.2 Components of a PCA graph

The second problem we solve is how to find the components of a PCA graph, when the input is a PCA model $\mathcal{M}$. A *leftmost* arc is an arc whose start point is not contained in any arc. When the graph is not connected, every model has at least two leftmost arcs. It is easy to find every leftmost arc in $O(n)$ time by traversing twice the circle. In the first traversal mark A_i when s_i is crossed, and then unmark A_i when t_i is crossed. In the second traversal, the start points having no marks when crossed correspond to leftmost arcs. Conversely, the start points that are crossed when there are marks are not from leftmost arcs. Now, if A_i and A_j are leftmost arcs with no leftmost arc between them then $A_i, \ldots, A_{j-1}$ is the range corresponding to the component of A_i. To sum up, the components of a PCA graph can be found in $O(n)$ time.

3.3 PCA representation of interval graphs [15]

The third problem is to determine whether the intersection graph G of a PCA model $\mathcal{M} = (C, \mathcal{A})$ is an interval graph. If affirmative, then we need to construct a proper interval model. We can check if $\mathcal{M}$ is an interval model in $O(n)$ time by checking if it contains any leftmost arc. In this case the output is $\mathcal{M}$. But if G is an interval graph and $\mathcal{M}$ is not an interval model, we can transform it into an interval model in $O(n)$ time, employing the algorithm described in [15]. There, it is shown that $\mathcal{M}$ must have three arcs covering the circle. Moreover, one of these arcs, say (s, t), must be universal. Then $\mathcal{M}' = (C, (\mathcal{A} \setminus \{(s,t)\}) \cup \{(t,s)\})$ is a proper interval model of G.

3.4 Minimum circular string [2, 20]

Finally we need to find the minimum of a circular string. The minimum circular string problem is to find every canonical position of S. For this it is enough to find one canonical position i and a period w such that $i + kw$ is canonical for every $k \geq 0$. This problem can be solved in $O(n)$ comparisons over the alphabet Σ [2, 20].

4 Canonical representation of PCA models

In this section we describe how to canonize a representation of a PCA model, so that equality of models can be tested by equality of representations. What we want is a function C from models to arc representations, so that $C(\mathcal{M}) = C(\mathcal{N})$ if and only if $\mathcal{M} =_M \mathcal{N}$. The idea is to take the "minimum" arc representation as the canonical representation. Fix a PCA model $\mathcal{M}$ and let 'a' $<$ 'b'. Define $\prec$ as the order over the arcs, where $A_i \prec A_j$ if and only if $E^i <_{lex} E^j$. Define also, $<_R$ as the total order over arc representations where $R^i(\mathcal{M}) <_R R^j(\mathcal{N})$ if and only if either $E^i(\mathcal{M}) <_{lex} E^j(\mathcal{N})$ or $E^i(\mathcal{M}) = E^j(\mathcal{N})$ and $t^i(\mathcal{M}) < t^j(\mathcal{N})$. Arc A_i is *canonical* if A_i is minimum with respect to $\prec$ and R^i is a *canonical representation* when R^i is minimum with respect to $<_R$. Since R^i can be uniquely determined from (E^i, t^i) then $<_R$ is a total order and therefore the canonical representation of $\mathcal{M}$ is unique. That is, if R^i and R^j are canonical representations then $R^i = R^j$.

Proposition 1. *Let $\mathcal{M}$ be a PCA model and $1 \leq i, i+j \leq n$. Then $E^{i+j} = E^i \ll j$ and $t^j = b_j - a_j + 1$, where a_j (b_j) is the position of the j-th 'a' ('b') in R^i.*

Theorem 1. *Let $\mathcal{M}$ be a PCA model. Then A_i is a canonical arc if and only if R^i is a canonical representation of $\mathcal{M}$.*

Proof. If A_i is not a canonical arc, then there exists $A_j \prec A_i$. Consequently $E^j <_{lex} E^i$ which implies that $R^j <_R R^i$, so R^i is not a canonical representation.

Now suppose that A_i is a canonical arc, and let R^{i+j} be a canonical representation of $\mathcal{M}$. Then both E^i and E^{i+j} are minimum sequences with respect to $<_{lex}$, so $E^i = E^{i+j}$. By Remark 2, it is enough to see that $t^i = t^{i+j}$. Let a_k be the position of '$\mathrm{a_k}$' in R^i and b_k be the position of '$\mathrm{b_k}$' in R^i for every $1 \leq k \leq n$. By Proposition 1, $E^{i+j} = E^i \ll a_j$ and $t^{i+j} = b^j - a_j + 1$.

Since $E^i = E^{i+j} = E^i \ll a_j$ and there is the same quantity of symbols 'a' and 'b' in E^i, then in $E^i[1+k; a_j + k - 1]$ there is also the same quantity of 'a' and 'b' for every $1 \leq k \leq 2n - a_j + 1$. Moreover, this quantity must be $j-1$ because in $E^i[1; a_j - 1]$ there are $j-1$ characters 'a'. Then, in $E[b_1; a_j + b_1 - 2]$ there are $j-1$ characters 'b' and $E^i(a_j + b_1 - 1)$ is also a 'b'. Consequently $b_j = a_j + b_1 - 1$, because b_j is the position of the j-th character 'b' after b_1. Hence $t^{i+j} = b^j - a_j + 1 = b_1 = t^i$ as required. $\square$

From now on, we denote by $C(\mathcal{M})$ the unique canonical representation of $\mathcal{M}$. The algorithm we present below finds $C(\mathcal{M})$ for a PCA model $\mathcal{M}$ using some arc representation R^i as input.

Algorithm 2 *Canonical representation of model $\mathcal{M}$*

```
1. Compute $E^i$ and $t^i$.
2. Find some canonical position $a_c$ of $E^i$ that corresponds to some character
   '$\mathrm{a_c}$' in $R^i$, and let $b_c$ be the position of character '$\mathrm{b_c}$' in $R^i$.
3. Output $r(E^i \ll a_c, a_c - b_c + 1)$.
```

The algorithm finds $C(\mathcal{M})$ by Remark 1, Proposition 1 and Theorem 1. With respect to its time complexity, Step 1 can be done in $O(n)$ time by Remark 1, Step 2 takes $O(n)$ time as shown in Subsection 3.4, and Step 3 takes $O(n)$ time by Remark 1. Thus the time complexity is $O(n)$.

In the next section we show how to find a unique canonical model $\mathcal{M}(G)$ of a PCA graph G, so that $C(\mathcal{M}(G))$ is the unique canonical representation of a PCA graph.

5 Canonical models of PCA graphs

We divide the canonization of PCA graphs in three non-disjoint cases. These are the connected PCA graphs which are co-connected or non co-bipartite, the proper interval graphs, and the co-bipartite PCA graphs.

In this section we need to sort models according to their canonical representations. Define $<_M$ as the total order between models where $\mathcal{M}_1 <_M \mathcal{M}_2$ if and only if $C(\mathcal{M}_1) <_R C(\mathcal{M}_2)$. Note that $\mathcal{M}_1 =_M \mathcal{M}_2$ if and only if $\mathcal{M}_1 \not<_M \mathcal{M}_2$ and $\mathcal{M}_2 \not<_M \mathcal{M}_1$ because $C(\mathcal{M}_1)$ is unique. Although $<_M$ corresponds to a natural way to compare models, it does not behave so well for the sorting. Nevertheless, all the information in $C(\mathcal{M})$ can be encoded nicely into a somehow compressed string by combining $E(\mathcal{M})$ and $t(\mathcal{M})$. Let $(E(\mathcal{M}), t(\mathcal{M}))$ be the extreme representation $r^{-1}(C(\mathcal{M}))$ for a model $\mathcal{M}$. Define $S(\mathcal{M})$ as the string that is obtained from $E(\mathcal{M})$ by replacing the character 'b' at position $t(\mathcal{M})$ with a character 'm'. Extend $<$ so that 'a' $<$ 'm' $<$ 'b'.

Proposition 2. *$\mathcal{M}_1 <_M \mathcal{M}_2$ if and only if $S(\mathcal{M}_1) <_{lex} S(\mathcal{M}_2)$.*

Now, if $\{\mathcal{M}_1, \ldots, \mathcal{M}_k\}$ is a multiset of models, we can lexicographically sort it in $O(\sum_{i=1}^{k} |\mathcal{M}_i|)$ time using the well known most significant digit (MSD) radix sort algorithm.

5.1 Connected PCA graphs which are co-connected or non co-bipartite

We start first with the connected PCA graphs which are co-connected or non co-bipartite. The motivation for considering this case is the following theorem.

Theorem 2 ([6]). *Connected PCA graphs which are co-connected or non co-bipartite have at most two non-equal models, one being the reverse of the other.*

Let $\mathcal{M}$ be a PCA model of a connected PCA graph which is co-connected or non co-bipartite G. Define $\mathcal{M}(G)$ as the minimum of $\mathcal{M}$ and $\mathcal{M}^{-1}$ with respect to $<_M$; $\mathcal{M}(G)$ can be computed in $O(n)$ time.

Corollary 1. *Let G_1, G_2 be two connected PCA graphs which are co-connected or non co-bipartite. Then G_1 is isomorphic to G_2 if and only if $\mathcal{M}(G_1) =_M \mathcal{M}(G_2)$*

5.2 Proper interval graphs

The second class is that of proper interval graphs. As for PCA graphs, we employ a basic theorem.

Theorem 3 ([6, 19]). *Connected proper interval graphs have at most two non-equal proper interval models, one being the reverse of the other.*

Extend $\mathcal{M}(G)$ to connected proper interval graphs, i.e. if $\mathcal{M}$ is a proper interval model of a connected graph G, define $\mathcal{M}(G)$ as the minimum between $\mathcal{M}$ and $\mathcal{M}^{-1}$.

Corollary 2. *Let G_1, G_2 be two connected proper interval graphs. Then G_1 is isomorphic to G_2 if and only if $\mathcal{M}(G_1) =_M \mathcal{M}(G_2)$.*

Now, let G be a proper interval graph and $G_1, \ldots, G_k$ be its components, where $\mathcal{M}(G_i) \leq_M \mathcal{M}(G_{i+1})$. Extend $\mathcal{M}(G)$ to the model where the circle is partitioned into k consecutive segments $S_1, \ldots, S_k$ and $\mathcal{M}(G_i)$ is contained in the i-th segment that appears in a traversal of the circle (see Figure 5.2). Clearly, $\mathcal{M}(G)$ is a proper interval model of G and is uniquely defined.

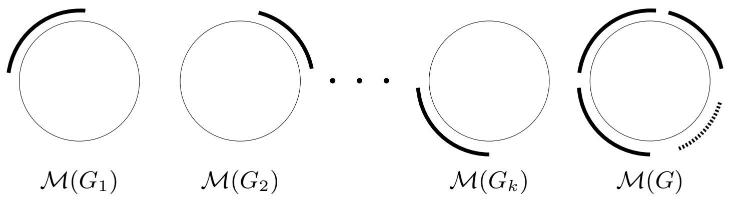


Fig. 1. The figures, except the last one, show the k segments whose corresponding models lie in $\mathcal{M}(G)$, whereas the last figure depicts $\mathcal{M}(G)$ itself.

Theorem 4. *Let G_1, G_2 be two proper interval graphs. Then G_1 is isomorphic to G_2 if and only if $\mathcal{M}(G_1) =_M \mathcal{M}(G_2)$.*

We describe below the algorithm to find $\mathcal{M}(G)$ for proper interval graphs when the input is (any arc representation of) $\mathcal{M}$.

Algorithm 3 *Canonical model of a proper interval graph G.*

```
1. Let $\mathcal{M}$ be some proper interval model of $G$
2. Find the components $\mathcal{M}_1, \ldots, \mathcal{M}_k$ of $\mathcal{M}$.
3. Define $\mathcal{M}(i)$ as the minimum of $\mathcal{M}_i$ and $\mathcal{M}_i^{-1}$ for $1 \leq i \leq k$.
4. Sort the multiset $\{\mathcal{M}(1), \ldots, \mathcal{M}(k)\}$ so that $\mathcal{M}(i) \leq_M \mathcal{M}(i+1)$ for every
   $1 \leq i < k$.
5. Output the model with $k$ segments where $\mathcal{M}(i)$ is contained in the $i$-th seg-
   ment.
```

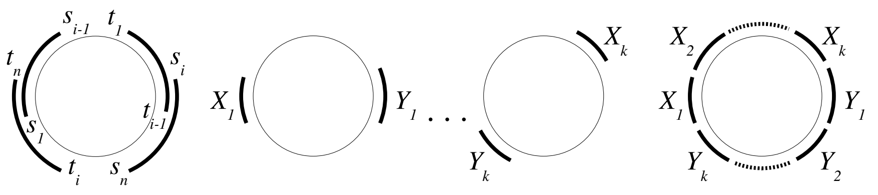
We now consider the complexity of the algorithm when $\mathcal{M}$ is given as in Step 1. Step 2 can be solved in $O(n)$ time as in Subsection 3.2, where we obtain a range representing each component. Step 3 is done by reversing $\mathcal{M}_i$ and then comparing $\mathcal{M}_i$ and $\mathcal{M}_i^{-1}$. Both the reversal and the comparison can be computed in $O(|\mathcal{M}_i|)$ for $1 \leq i \leq k$. Step 4 can done as explained at the beginning of this section in $O(n)$ time. For Step 5 traverse the range corresponding to $\mathcal{M}_i$ that was obtained in Step 1 and insert it into the new circle. The algorithm then runs in $O(n)$ time.

5.3 Canonization of co-bipartite PCA graphs

Finally we consider co-bipartite PCA graphs. The algorithm for this class of graphs is quite similar in its concept to the one for proper interval graphs. For co-connected PCA graphs $\mathcal{M}(G)$ has already been defined in Subsection 5.1.

Consider a co-connected co-bipartite PCA graph G and denote by $\mathcal{A}_1, \mathcal{A}_2$ the co-bipartition of $\mathcal{M}(G)$. By Lemma 2, both $\mathcal{A}_1$ and $\mathcal{A}_2$ are ranges. Assume w.l.o.g. that A_1 is the first arc of range $\mathcal{A}_1$ and A_i is the first arc of range $\mathcal{A}_2$. Moreover, assume that s_1 is represented by 'a_1' in $C(\mathcal{M}(G))$. In the segment (s_{i-1}, t_1) there is no start point of arcs in $\mathcal{A}_2$, because otherwise every arc of $\mathcal{A}_1$ would contain these start points. Consequently, (s_{i-1}, t_1) is a segment contained by all the arcs of $\mathcal{A}_1$ that is not crossed by any arcs of $\mathcal{A}_2$. Moreover, (s_{i-1}, t_1) is the unique maximal segment in these conditions. The same argument can be applied interchanging $\mathcal{A}_1$ with $\mathcal{A}_2$. But in the case where $\mathcal{A}_2$ is empty and $\mathcal{A}_1$ has only one universal arc A_1, then (t_1, s_1) is the required segment. This means that $X = \{t_i, \ldots, t_n\} \cup \{s_1, \ldots, s_{i-1}\}$ and $Y = \{t_1, \ldots, t_{i-1}\} \cup \{s_i, \ldots, s_n\}$ are two st-ranges that define $\mathcal{M}(G)$ (see Figure 2(a)). We call these two st-ranges as *co-bipartition ranges*, where X is the *low co-bipartition range* and Y is the *high co-bipartition range*. Observe that low and high are well defined for $\mathcal{M}(G)$, because X contains 'a_1' in $C(\mathcal{M}(G))$.

Now we show a unique way to accommodate the co-components when the PCA graph is not co-connected (see also [6]). This is rather similar to what we did in the previous section. Let G be a non interval co-bipartite PCA graph and $G_1, \ldots, G_k$ be its co-components where $\mathcal{M}(G_i) \leq_M \mathcal{M}(G_{i+1})$ for $1 \leq i \leq k$. Let X_i, Y_i be the respective low and high co-bipartitions ranges of $\mathcal{M}(G_i)$ for $1 \leq i < k$. Define $\mathcal{M}(G)$ as the model where the circle is partitioned into $2k$ consecutive segments. The i-th segment in a traversal of the circle contains X_i and the $i+k$ segment contains Y_i for $1 \leq i \leq k$ (see Figure 2). It is not hard to see that $\mathcal{M}(G)$ is a PCA model of G, because the model induced by the arcs in segments i and $i+k$ is precisely $\mathcal{M}(G_i)$ and every arc with one extreme in segment i intersects every arc with one extreme in segment j for $1 \leq i < j \leq k$.



(a) $\mathcal{M}(H)$ (b) $\mathcal{M}(G_1)$ (c) $\mathcal{M}(G_k)$ (d) $\mathcal{M}(G)$

Fig. 2. Figure (a) shows the high co-bipartition range $\{t_i, \ldots, t_n\} \cup \{s_1, \ldots, s_{i-1}\}$ and the low co-bipartition range $\{t_1, \ldots, t_{i-1}\} \cup \{s_i, \ldots, s_n\}$ of a co-connected co-bipartite graph. Figures (b) and (c) show the co-bipartition ranges of the co-components of G in their corresponding segments. In (d) the whole picture of $\mathcal{M}(G)$ is shown.

Theorem 5. *Let G_1, G_2 be two non interval co-bipartite PCA graphs. Then G_1 is isomorphic to G_2 if and only if $\mathcal{M}(G_1) =_M \mathcal{M}(G_2)$.*

The algorithm to find a canonical representation of a co-bipartite PCA graph is very similar to the one for a proper interval graph. The two main changes are that components are replaced by co-components in Steps 2-5, and that the circle is partitioned into $2k$ segments, where segments from 1 to k contain the low co-bipartition ranges and segments from $k+1$ to $2k$ contain the high co-bipartition ranges (Step 6). Since both the co-components and the co-bipartition ranges can be found in $O(n)$ time, as in Section 3.1, the whole algorithm for this case takes $O(n)$ time.

6 Putting it all together

Function $\mathcal{M}$ as defined in the previous section maps every PCA graph to a PCA model. However, it should be mentioned that if G is both proper interval and co-bipartite, then $\mathcal{M}(G)$ is computed as in Subsection 5.2. The complete algorithm is depicted below.

Algorithm 4 *Canonical representation of a PCA graph G*

```
1. If $G$ is an interval graph, then compute $\mathcal{M}(G)$ as in Subsection 5.2.
2. Else if $G$ is a co-bipartite model then compute $\mathcal{M}(G)$ as in Subsection 5.3.
3. Otherwise, compute $\mathcal{M}(G)$ as in Subsection 5.1.
```

Finally we discuss the complexity of the entire algorithm. The input of the algorithm is a PCA model as in Step 1. This model is encoded as an arc representation, which is obtained as the output of the recognition algorithm for PCA graphs [6]. We can check if G is an interval graph as in Subsection 3.3 in $O(n)$ time. If so, we obtain a proper interval model that we can use in Step 1 to find $\mathcal{M}(G)$ in $O(n)$ time. The rest of the algorithm takes $O(n)$ time as explained in the previous section. When the input is G instead of $\mathcal{M}$, the algorithm takes $O(n+m)$ time by first computing a PCA model [6].

Theorem 6. *Let G and H be two PCA graphs. Then the following are equivalent:*

1. *G and H are isomorphic,*
2. *$\mathcal{M}(G) =_M \mathcal{M}(H)$,*
3. *$C(\mathcal{M}(G)) = C(\mathcal{M}(G))$.*

Proof. It is a direct consequence of Corollary 1 and Theorems 4 and 5, and the fact that $\mathcal{M}$ is a well defined function. □

Corollary 3. *The isomorphism problem for PCA graphs can be solved in $O(n)$ time when a PCA models model is given as input, or in $O(n+m)$ time when the input is a graph given by its sets of vertices and edges.*

References

1. Bhattacharya, B., Hell P., Huang J.: A linear algorithm for maximum weight cliques in proper circular arc graphs. SIAM J. Discrete Math. 9 (2), 274–289 (1996)
2. Booth, K.S.: Lexicographically least circular substrings. Inform. Process. Lett. 10 (4-5), 240–242 (1980)
3. Booth, K.S., Lueker, G.S.: Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. J. Comput. System Sci. 13 (3), 335–379 (1976)
4. Brandstädt, A., Le, V.B., Spinrad, J.P.: Graph classes: a survey. SIAM, Philadelphia (1999)
5. Corneil, D.G., Olariu, S., Stewart, L.: The ultimate interval graph recognition algorithm? (extended abstract). In: 9th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 175–180. ACM, New York (1998)
6. Deng, X., Hell, P., Huang, J.: Linear-time representation algorithms for proper circular-arc graphs and proper interval graphs. SIAM J. Comput. 25 (2), 390–403 (1996)
7. Garey, M.R., Johnson, D.S.: Computers and intractability. W. H. Freeman and Co., San Francisco (1979)
8. Golumbic, M.C.: Algorithmic Graph Theory and Perfect Graphs, second ed. North–Holland Publishing Co., Amsterdam (2004)
9. Habib, M., McConnell, R.M., Paul, C., Viennot, L.: Lex-BFS and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. Theoret. Comput. Sci. 234 (1-2) 59–84 (2000)
10. Hsu, W.: A simple test for interval graphs. In: Mayr, E.W. (eds.) WG 1992. LNCS, vol. 657, pp. 11–16, Springer, Heidelberg (1993).
11. Hsu, W.: $O(m.n)$ algorithms for the recognition and isomorphism problems on circular-arc graphs. SIAM J. Comput. 24 (3), 411–439 (1995)
12. Kaplan, H., Nussbaum, Y.: A simpler linear-time recognition of circular-arc graphs. In: Arge, L., Freivalds, R. (eds.) SWAT 2006. LNCS, vol. 4059, pp. 41–52, Springer, Heidelberg (2006)
13. Kaplan, H., Nussbaum, Y.: Certifying algorithms for recognizing proper circular-arc graphs and unit circular-arc graphs. In: Fomir, F. (eds.) WG 2006. LNCS, vol. 4271, pp. 289–300, Springer, Heidelberg (2006)
14. Korte, N., Möhring, R.H.: An incremental linear-time algorithm for recognizing interval graphs. SIAM J. Comput. 18 (1), 68–81 (1989)
15. Lin, M.C., Soulignac, F.J., Szwarcfiter, J.L.: Proper Helly circular-arc graphs. In: Brandstdt, A., Kratsch, D., Müller, H. (eds) WG 2007, LNCS, vol. 4769, pp. 248–257, Springer, Heidelberg (2007),
16. Lin, M.C., Szwarcfiter, J.L.: Unit Circular-Arc Graph Representations and Feasible Circulations. SIAM J. Discrete Math. 22 (1), 409–423 (2008)
17. Lueker, G.S., Booth, K.S.: A linear time algorithm for deciding interval graph isomorphism. J. Assoc. Comput. Mach. 26 (2), 183–195 (1979)
18. McConnell, R.M.: Linear-time recognition of circular-arc graphs. Algorithmica 37 (2), 93–147 (2003)
19. Roberts, F.S.: Indifference graphs. In: Proof Techniques in Graph Theory (2nd Ann Arbor Graph Theory Conf.), pp. 139–146. Academic Press, New York (1969)
20. Shiloach, Y.: Fast canonization of circular strings. J. Algorithms 2 (2), 107–121 (1981)
21. Spinrad, J.P.: Efficient graph representations. American Mathematical Society, Providence (2003)