Grafos Arco-Circulares

Problemas de Grafos y Tratabilidad Computacional

Definición

Un grafo $G = (V, E)$ es arco-circular si existe una familia de arcos $\mathcal{A} = \{A_1, \cdots, A_n\}$ sobre un círculo C tal que G es el grafo intersección de $\mathcal{A}$, es decir que, $V = \{v_1, \cdots, v_n\}$ donde cada v_i corresponde al arco A_i y la arista $(v_i, v_j) \in E \Leftrightarrow i \neq j \wedge$ $A_i \cap A_j \neq \emptyset$. En tal caso, $(\mathcal{A}, C)$ es un modelo o representación arco-circular de G.

Observación: Todo grafo de intervalos es arco-circular y no es cierta la vuelta, por ejemplo los C_k inducido con $k \geq 4$.

Teorema

Dado $G = (V, E)$ un grafo arco-cicular si y sólo si existe un orden circular $\sigma = [v_1, \cdots, v_n]$ de los vértices de V de manera tal que si $(v_i, v_j) \in E$ entonces $v_{i+1}, \cdots, v_j \in N(v_i) \vee v_{j+1}, \cdots, v_i \in N(v_j)$.

- $\Rightarrow$) Sea $(\mathcal{A}, C)$ un modelo arco-circular de G y $A_1 = (s_1, t_1), \cdots, A_n = (s_n, t_n)$ los arcos de $\mathcal{A}$ ordenados por el extremo de comienzo. Sea $[v_1, \cdots, v_n]$ el orden circular de los vértices donde v_i corresponde a A_i. Si $(v_i, v_j) \in E$ entonces $A_i \cap A_j \neq \emptyset$, lo cual quiere decir que $s_j \in A_i$ o $s_i \in A_j$. En el primer caso, $s_{i+1}, \cdots, s_j \in A_i$ y por lo tanto, $v_{i+1}, \cdots, v_j \in N(v_i)$. En el último caso, $s_{j+1}, \cdots, s_i \in A_j$ y $v_{j+1}, \cdots, v_i \in N(v_j)$.
- $\Leftarrow$) Sea $\sigma = [v_1, \cdots, v_n]$, el orden circular que cumple la hipótesis, un $\epsilon < \frac{\pi}{10000n}$ positivo y $p_i = \frac{2i\pi}{n}$ para $1 \leq i \leq n$, los n puntos de un círculo C. Para cada v_i, buscar en el orden circular a partir de v_{i+1}, el primer vértice v_j no adyacente a v_i (en el peor de los casos $j = i$) y definir $A_i = (p_i + \epsilon, p_j - i\epsilon)$. Si $(v_i, v_j) \notin E$ entonces $s_j = p_j + \epsilon \notin A_i$ y $s_i = p_i + \epsilon \notin A_j$, por lo cual $A_i \cap A_j = \emptyset$. Si $(v_i, v_j) \in E$ entonces $v_{i+1}, \cdots, v_j \in N(v_i) \vee v_{j+1}, \cdots, v_i \in N(v_j)$. En el primer caso, $s_j = p_j + \epsilon \in A_i$. En el último caso, $s_i = p_i + \epsilon \in A_j$. En cualquier caso,$A_i \cap A_j \neq \emptyset$.

Reconocimiento

- La caraterización anterior no conduce a un algoritmo de reconocimiento de tiempo polinomial.
- $O(n^3)$ Alan Tucker. An Efficient Test for Circular-Arc Graphs. SIAM Journal on Computing, 1980.
- $O(mn)$ Wen-Lian Hsu. $O(mn)$ Algorithms for the Recognition and Isomorphism Problems on Circular-Arc Graphs. SIAM Journal on Computing, 1995.
- $O(n^2)$ Elaine M. Eschen, Jeremy Spinrad. An $O(n^2)$ Algorithm for Circular-Arc Graph Recognition. Proc. 4th Annual ACM-SIAM Symposium on Discrete Algorithms, 1993.
- $O(n+m)$ Ross M. McConnell. Linear-Time Recognition of Circular-Arc Graphs. Proc. 42nd Annual Symposium on Foundations of Computer Science, 2001.
- $O(n+m)$ Haim Kaplan, Yahav Nussbaum. A simpler linear-time recognition of circular-arc graphs. Proc. 10th Scandinavian Workshop on Algorithm Theory, 2006.

Subclases (¿son hereditarias?)

Dado un modelo arco-circular $(\mathcal{A}, C)$ y G su grafo de intersección.

- Si no existen 2 arcos $A_i, A_j \in \mathcal{A}$ tal que $A_i \cup A_j = C$ entonces este modelo es normal y G es un grafo arco-circular normal.
- Si no existen 2 arcos $A_i, A_j \in \mathcal{A}$ tal que $A_i \subset A_j$ entonces este modelo es propio y G es un grafo arco-circular propio.
- Si todos los arcos de $\mathcal{A}$ tienen la misma longitud entonces este modelo es unitario y G es un grafo arco-circular unitario.
- Si $\mathcal{A}$ cumple propiedad de Helly entonces este modelo es Helly y G es un grafo arco-circular Helly.

Observaciones

- Todo grafo de intervalos es grafo arco-circular Helly.
- Todo modelo arco-circular unitario es arco-circular propio.
- Todo grafo arco-circular propio es arco-circular normal.
- Un grafo arco-circular Helly $G = (V, E)$ tiene a lo sumo $|V|$ cliques.

Teorema

Sea $G = (V, E)$ un grafo. Las siguientes afirmaciones son equivalentes.

1. G es un grafo arco-circular Helly.
2. La matriz clique M de G cumple la propiedad de 1's circular por columnas que es lo mismo decir que se pueden ordenar circularmente los cliques de G de manera tal que para cada vértice $v \in V$, los cliques que lo contienen están en forma consecutiva en este orden circular.

Observación: ¿Cuántos 1's puede tener la matriz clique de un grafo arco-circular Helly $G = (V, E)$?
Reconocimiento: $O(n^3)$ F. Gavril. Algoritms on Circular-Arc Graphs. Networks, 1974.

Characterizations and Linear Time Recognition of Helly Circular-Arc Graphs

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

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

Abstract. A circular-arc model $(C, \mathcal{A})$ is a circle C together with a collection $\mathcal{A}$ of arcs of C. If $\mathcal{A}$ satisfies the Helly Property then $(C, \mathcal{A})$ is a Helly circular-arc model. A (Helly) circular-arc graph is the intersection graph of a (Helly) 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 algorithm have been described both for the general class and for some of its subclasses. However, for Helly circular-arc graphs, the best recognition algorithm is that by Gavril, whose complexity is $O(n^3)$. In this article, we describe different characterizations for Helly circular-arc graphs, including a characterization by forbidden induced subgraphs for the class. The characterizations lead to a linear time recognition algorithm for recognizing graphs of this class. The algorithm also produces certificates for a negative answer, by exhibiting a forbidden subgraph of it, within this same bound.

Keywords: algorithms, circular-arc graphs, forbidden subgraphs, Helly circular-arc graphs.

1 Introduction

Circular-arc graphs form a class of graphs which has attracted much interest, since its first characterization by Tucker, almost fourty years ago [9]. There is a particular interest in the study of subclasses of it. The most common of these subclasses are the proper circular-arc graphs, unit circular-arc graphs and Helly circular-arc graphs (Golumbic [3]). Linear time recognition and representation algorithms have been already formulated for general circular-arc graphs (McConnell [7], Kaplan and Nussbaum [5]), proper circular-arc graphs (Deng,

⋆ Partially supported by UBACyT Grants X184 and X212, PICT ANPCyT Grant 11-09112, CNPq under PROSUL project Proc. 490333/2004-4.
⋆⋆ 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.

D.Z. Chen and D.T. Lee (Eds.): COCOON 2006, LNCS 4112, pp. 73–82, 2006.

Hell and Huang [1]) and unit circular-arc arc graphs (Lin and Szwarcfiter [6]). For Helly circular-arc graphs, the best recognition algorithm is by Gavril [2], which requires $O(n^3)$ time. Such an algorithm is based on characterizing Helly circular-arc graphs, as being exactly those graphs whose clique matrices admit the circular 1's property on their columns [2]. The book by Spinrad [8] contains an appraisal of circular-arc graph algorithms.

In the present article, we propose new characterizations for Helly circular-arc graphs, including a characterization by forbidden induced subgraphs for the class. The characterizations lead to a linear time algorithm for recognizing graphs of the class and constructing the corresponding Helly circular-arc models. In case a graph does not belong to the class, the method exhibits a certificate, namely a forbbiden induced subgraph of it, also in linear time.

Let G be a graph, V_G, E_G its sets of vertices and edges, respectively, $|V_G| = n$ and $|E_G| = m$. Write $e = v_i v_j$, for an edge $e \in E_G$, incident to $v_i, v_j \in V_G$. A *clique* of G is a maximal subset of pairwise adjacent vertices. Denote $N(v_i) = \{v_j \in V_G | v_i v_j \in E_G\}$, call $v_j \in N(v_i)$ a *neighbour* of v_i and write and $d(v_i) = |N(v_i)|$.

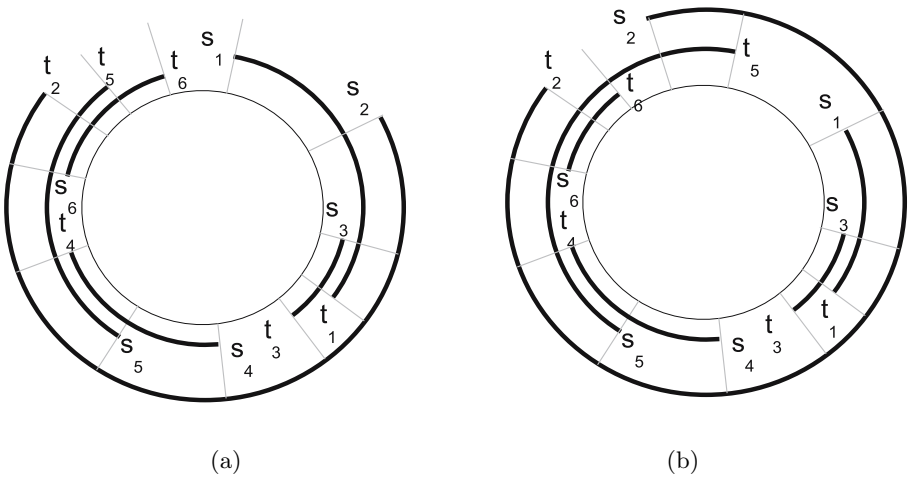
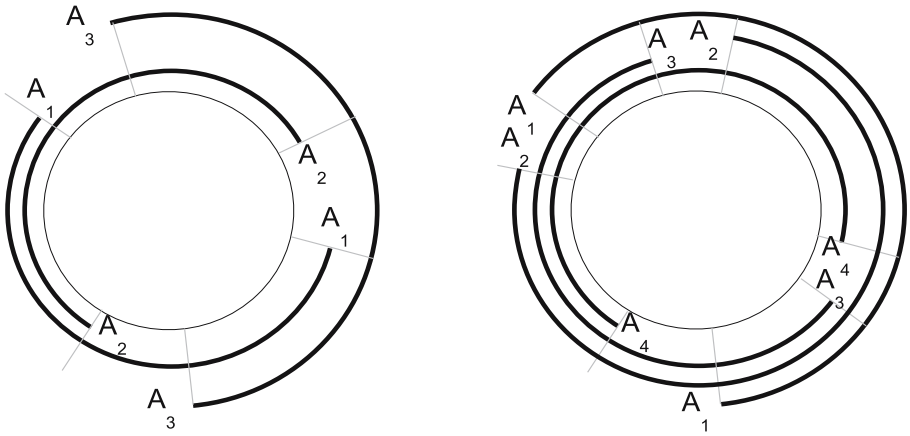


Fig. 1. Two circular-arc models

A *circular-arc (CA) model* $(C, \mathcal{A})$ is a circle C together with a collection $\mathcal{A}$ of arcs of C. Unless otherwise stated, we always traverse C in the clockwise direction. Each arc $A_i \in \mathcal{A}$ is written as $A_i = (s_i, t_i)$, where $s_i, t_i \in C$ are the *extreme points* of A_i, with s_i the *start point* and t_i the *end point* of the arc, respectively, in the clockwise direction. The *extremes* of $\mathcal{A}$ are those of all arcs $A_i \in \mathcal{A}$. As usual, we assume that no single arc of $\mathcal{A}$ covers C, that no two extremes of $\mathcal{A}$ coincide and that all arcs of $\mathcal{A}$ are open. When traversing C, we obtain a circular ordering of the extreme points of $\mathcal{A}$. Furthermore, we also consider a circular ordering $A_1, \ldots, A_n$ of the arcs of $\mathcal{A}$, defined by the

corresponding circular ordering $s_1, \ldots, s_n$ of their respective start points. In general, when dealing with a sequence $x_1, \ldots, x_t$ of t objects circularly ordered, we assume that all the additions and subtractions of the indices i of the objects x_i are modulo t. Figure 1 illustrates two CA models, with the orderings of their arcs.



(a) (b)

Fig. 2. Two minimally non Helly models

In a model $(C, \mathcal{A})$, the *complement* of an arc $A_i = (s_i, t_i)$ is the arc $\overline{A_i} = (t_i, s_i)$. Complements of arcs have been employed before by McConnell [7], under the name *arc flippings*. The *complement* of $(C, \mathcal{A})$ is the model $(C, \overline{\mathcal{A}})$, where $\overline{\mathcal{A}} = \{\overline{A_i} | A_i \in \mathcal{A}\}$.

In the model $(C, \mathcal{A})$, a subfamily of arcs of $\mathcal{A}$ is *intersecting* when they pairwise intersect. Say that $\mathcal{A}$ is *Helly*, when every intersecting subfamily of it contains a common point of C. In this case, $(C, \mathcal{A})$ is a *Helly circular-arc (HCA) model*. When $\mathcal{A}$ is not Helly, it contains a minimal non Helly subfamily $\mathcal{A}'$, that is $\mathcal{A}'$ is not Helly, but $\mathcal{A}' \setminus A_i$ is so, for any $A_i \in \mathcal{A}'$. The model $(C, \mathcal{A}')$ is then *minimally non HCA*. Figure 2 depicts two minimally non Helly models.

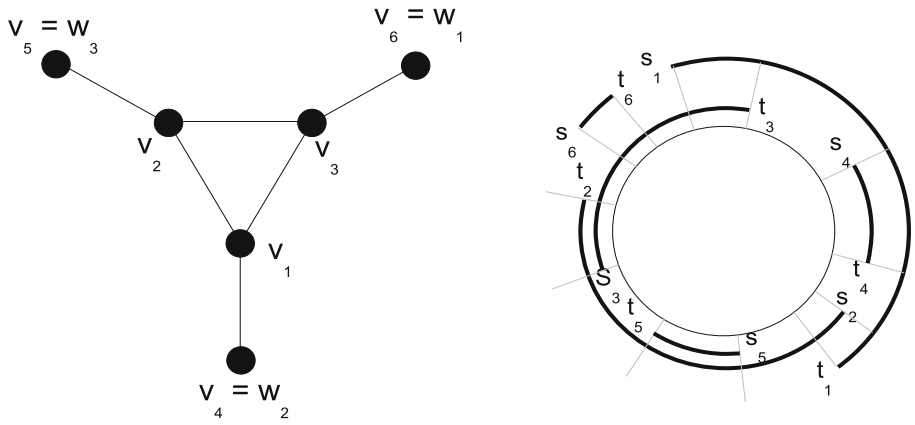
A *circular-arc (CA) graph* G is the intersection graph of some CA model $(C, \mathcal{A})$. Denote by $v_i \in V_G$ the vertex of G corresponding to $A_i \in \mathcal{A}$. Similarly, a *Helly circular-arc (HCA) graph* is the intersection graph of some HCA model. In a HCA graph, each clique $Q \subseteq V_G$ can be represented by a point $q \in C$, which is common to all those arcs of $\mathcal{A}$, which correspond to the vertices of Q. Clearly, two distinct cliques must be represented by distinct points. Finally, two CA models are *equivalent* when they share the same intersection graph.

In the next section, we present the main basic concepts, in which the proposed characterizations are based. In Section 3, we characterize HCA models, while HCA graphs are characterized in Section 4. In Section 5, we describe the

construction of a special CA model, which is employed in the recognition algorithm. Finally, Section 6 describes the recognition algorithm, together with its certificates. Withou loss of generality, we consider all given graphs to be connected.

2 Central Definitions

In this section, we describe usefull concepts for the proposed method. Let G be a graph and $(C, \mathcal{A})$ a CA model of it. First, define special sequences of extremes of the arcs of $\mathcal{A}$.



(a) (b)

Fig. 3. An obstacle and its non Helly stable model

An *s-sequence (t-sequence)* is a maximal sequence of start points (end points) of $\mathcal{A}$, in the circular ordering of C. Write *extreme sequence* to mean an s-sequence or t-sequence. The $2n$ start points and end points are then partitioned into s-sequences and t-sequences, which alternate in C. For an extreme sequence E, denote by $FIRST(E)$ the first element of E, while the notations $NEXT(E)$ and $NEXT^{-1}(E)$ represent the extreme sequences which succeeds and preceeds E in C, respectively. For an extreme point $p \in C$, denote $SEQUENCE(p)$ the extreme sequence which contains p, while $NEXT(p)$ means the sequence $NEXT(SEQUENCE(p))$. Through the paper, we employ operations on the CA models, which possibly modify them, while preserving equivalence. A simple example of such operations is to permute the extremes of the arcs, whitin a same extreme sequence.

Next, we define a special model of interest.

Definition 1. *Let s_i be a start point of $\mathcal{A}$ and $S = SEQUENCE(s_i)$. Say that s_i is stable when $i = j$ or $A_i \cap A_j = \emptyset$, for every $t_j \in NEXT^{-1}(S)$.*

Definition 2. *A model* $(C, \mathcal{A})$ *is* stable *when all its start points are stable.*

As examples, the models of Figures 1(a) and 1(b) are not stable, while that of Figure 3(b) is.

We will employ stable models in the recognition process of HCA graphs.

Finally, define a special family of graphs.

Definition 3. *An* obstacle *is a graph* H *containing a clique* $K_t \subseteq V_H$, $t \geq 3$, *whose vertices admit a circular ordering* $v_1, \ldots, v_t$, *such that each edge* $v_i v_{i+1}$, $i = 1, \ldots, t$, *satisfies:*

(i) $N(w_i) \cap K_t = K_t \setminus \{v_i, v_{i+1}\}$, *for some* $w_i \in V_H \setminus K_t$, *or*
(ii) $N(u_i) \cap K_t = K_t \setminus \{v_i\}$ *and* $N(z_i) \cap K_t = K_t \setminus \{v_{i+1}\}$, *for some adjacent vertices* $u_i, z_i \in V_H \setminus K_t$.

As example, the graph of Figure 3(a) is obstacle.

We will show that the obstacles form a family of forbidden induced subgraphs for a CA graph to be HCA.

3 Characterizing HCA Models

In this section, we describe a characterization and a recognition algorithm for HCA models. The characterization is as follows:

Theorem 1. *A CA model* $(C, \mathcal{A})$ *is HCA if and only if*

(i) if three arcs of $\mathcal{A}$ *cover* C *then two of these three arcs also cover it, and*
(ii) the intersection graph of $(C, \overline{\mathcal{A}})$ *is chordal.*

Proof. By hypothesis, $(C, \mathcal{A})$ is a HCA model. Condition (i) is clear, otherwise $(C, \mathcal{A})$ can not be HCA. Suppose Condition (ii) fails. Then the intersection graph G^c of $(C, \overline{\mathcal{A}})$ contains an induced cycle C^c, with length $k > 3$. Let $\overline{\mathcal{A}'} \subseteq \overline{\mathcal{A}}$ be the set of arcs of $\overline{\mathcal{A}}$, corresponding to the vertices of C^c, and $\mathcal{A}' \subseteq \mathcal{A}$ the sets of the complements of the arcs $\overline{A_i} \in \overline{\mathcal{A}'}$. First, observe that no two arcs of $\overline{\mathcal{A}'}$ cover the circle, otherwise C^c would contain a chord. Consequently, $\overline{\mathcal{A}'}$ consists of k arcs circularly ordered as $\overline{A_1}, \ldots, \overline{A_k}$ and satisfying: $\overline{A_i} \cap \overline{A_j} \neq \emptyset$ if and only if $\overline{A_i}, \overline{A_j}$ are consecutive in the circular ordering. In general, comparing a model $(C, \mathcal{A})$ to its complement model $(C, \overline{\mathcal{A}})$, we conclude that two arcs of $\mathcal{A}$ intersect if and only if their complements in $\overline{\mathcal{A}}$ are either disjoint or intersect without covering the circle. Consequently, $\mathcal{A}'$ must be an intersecting family. On the other hand, the arcs of $\mathcal{A}'$ can not have a common point $p \in C$. Because, otherwise $p \notin \overline{A_i}$, for all $\overline{A_i}$, meaning that the arcs of $\overline{\mathcal{A}'}$ do not cover the circle, contradicting C^c to be an induced cycle. The inexistence of a common point in $\mathcal{A}'$ implies that $\mathcal{A}$ is not a Helly family, a contradiction. Then (ii) holds. The converse is similar.$_\triangle$

The following characterizes minimally non Hely models.

Corollary 1. *A model $(C, \mathcal{A})$ is minimaly non HCA if and only if*

(i) $\mathcal{A}$ is intersecting and covers C, and
(ii) two arcs of $\mathcal{A}$ cover C precisely when they are not consecutive in the circular ordering of $\mathcal{A}$.

Theorem 1 leads directly to a simple algorithm for recognizing Helly models, as follows. Given a model $(C, \mathcal{A})$ of some graph G, verify if $(C, \mathcal{A})$ satisfies Condition (i) and then if it satisfies Condition (ii). Clearly, $(C, \mathcal{A})$ is HCA if and only if both conditions are satisfied. Next, we describe methods for checking them.

For Condition (i), we seek directly for the existence of three arcs $A_i, A_j, A_k \in \mathcal{A}$ that cover C, two of them not covering it. Observe that there exist such arcs if and only if the circular ordering of their extremes is $s_i, t_k, s_j, t_i, s_k, t_j$. For each $A_i \in \mathcal{A}$, we repeat the following procedure, which looks for the other two arcs A_j, A_k whose extreme points satisfy this ordering. Let L_1 be the list of extreme points of the arcs contained in (s_i, t_i), in the ordering of C. First, remove from L_1 all pairs of extremes s_q, t_q of a same arc, which may possibly occur. Let L_2 be the list formed by the other extremes of the arcs represented in L_1. That is, $s_q \in L_1$ if and only if $t_q \in L_2$, and $t_q \in L_1$ if and only if $s_q \in L_2$, for any $A_q \in \mathcal{A}$. Clearly, the extremes points which form L_2 are all contained in (t_i, s_i), and we consider them in the circular ordering of C. Denote by $FIRST(L_1)$ and $LAST(L_2)$ the first and last extreme points of L_1 and L_2, in the considered orderings, respectively. Finally, iteratively perform the steps below, until either $L_1 = \emptyset$, or $FIRST(L_1) = t_k$ and $LAST(L_2) = t_j$, for some j, k.

if $FIRST(L_1)$ is a start point s_q then remove s_q from L_1 and t_q from L_2
if $LAST(L_2)$ is a start point s_q then remove s_q from L_2 and t_q from L_1

If the iterations terminate because $L_1 = \emptyset$ then there are no two arcs which together with A_i satisfy the above requirements, completing the computations relative to A_i. Otherwise, the arcs A_k and A_j, whose end points are $FIRST(L_1)$ and $LAST(L_2)$, form together with A_i a certificate for the failure of Condition (i). Each of the n lists L_2 needs to be sorted. There is no difficulty to sort them all together in time $O(m)$, at the beginning of the process. The computations relative to A_i require $O(d(v_i))$ steps. That is, the overall complexity of checking Condition (i) is $O(m)$.

For Condition (ii), the direct approach would be to construct the model $(C, \overline{\mathcal{A}})$, its intersection graph G^c and apply a chordal graph recognition algorithm to decide if G^c is chordal. However, the number of edges of G^c could be $O(n^2)$, breaking the linearity of the proposed method. Alternatively, we check whether the complement $\overline{G^c}$ of G^c is co-chordal. Observe that two vertices of $\overline{G^c}$ are adjacent if and only if their corresponding arcs in $\mathcal{A}$ cover the circle. Consequently, the number of edges of $\overline{G^c}$ is at most that of G, i.e. $\leq m$. Since co-chordal graphs can be recognized in linear time (Habib, McConnell, Paul and Viennot [4]), the complexity of the method for verifying Condition (ii) is $O(m)$.

Consequently, HCA models can be recognized in linear time.

4 Characterizing HCA Graphs

In this section, we describe the proposed characterizations for HCA graphs.

Theorem 2. *The following affirmative are equivalent for a CA graph G.*

(a) G is HCA.
(b) G does not contain obstacles, as induced subgraphs.
(c) All stable models of G are HCA.
(d) One stable model of G is HCA.

Proof. (a) $\Rightarrow$ (b): By hypothesis, G is HCA. Since HCA graphs are hereditary, it is sufficient to prove that no obstacle H is a HCA graph. By contrary, suppose H admits a HCA model $(C, \mathcal{A})$. Let K_t be the core of H. By Definition 3, there is a circular ordering $v_1, \ldots, v_t$ of the vertices of K_t which satisfies Conditions (i) or (ii) of it. Denote by $\mathcal{A}' = \{A_1, \ldots, A_t\} \subseteq \mathcal{A}$ the family of arcs corresponding to K_t. Define a clique C_i of H, for each $i = 1, \ldots, t$, as follows. If (i) of Definition 3 is satisfied then $C_i \supseteq \{w_i\} \cup K_t \setminus \{v_i, v_{i+1}\}$, otherwise (ii) is satisfied and $C_i \supseteq \{u_i, z_i\} \cup K_t \setminus \{v_i, v_{i+1}\}$. Clearly, all cliques $C_1, \ldots, C_t$ are distinct, because any two of them contain distinct subsets of K_t. Since H is HCA, there are distinct points $p_1, \ldots, p_t \in C$, representing $C_1, \ldots, C_t$, respectively. We know that $v_i \in C_j$ if and only if $i \neq j-1, j$. Consequently, $p_j \in A_i$ if and only if $i \neq j-1, j$. The latter implies that $p_1, \ldots, p_t$ are also in the circular ordering of C. On the other hand, because K_t is a clique distinct from any C_i, there is also a point $p \in C$ representing K_t. Try to locate p in C. Clearly, p lies between two consecutive points p_{i-1}, p_i. Examine the vertex $v_i \in K_t$ and its corresponding arc $A_i \in \mathcal{A}'$. We already know that $p \in A_i$, while $p_{i-1}, p_i \notin A_i$. Furthermore, because $t \geq 3$, there is $j \neq i-1, i$ such that $p_j \in A_i$. Such situation can not be realized by arc A_i. Then $(C, \mathcal{A})$ is not HCA, a contradiction.

(b) $\Rightarrow$ (c): By hypothesis, G does not contain obstacles. By contrary, suppose that there exists a stable model $(C, \mathcal{A})$ of G, which is not HCA. Let $\mathcal{A}' \subseteq \mathcal{A}$ be a minimally non Helly subfamily of $\mathcal{A}$. Denote by $A_1, \ldots, A_t$ the arcs of $\mathcal{A}'$ in the circular ordering. Their corresponding vertices in G are $v_1, \ldots, v_t$, forming a clique $K_t \subseteq V_G$. Let A_i, A_{i+1} be two consecutive arcs of $\mathcal{A}'$, in the circular ordering. By Corollary 1, A_i, A_{i+1} do not cover C. Denote $T = SEQUENCE(t_{i+1})$ and $S = SEQUENCE(s_i)$. Because $(C, \mathcal{A})$ is stable, $S \neq NEXT(T)$. Let $S' = NEXT(T)$ and $T' = NEXT^{-1}(S)$. Choose $s_z \in S$ and $t_u \in T'$. We know that A_z does not intersect A_{i+1}, nor does A_u intersect A_i, again because the model is stable. Since $s_z, t_u \in (t_{i+1}, s_i)$, Corollary 1 implies that $s_z, t_u \in A_j$, for any $A_j \in \mathcal{A}'$, $A_j \neq A_i, A_{i+1}$. Denote by z_i and u_i the vertices of G corresponding to A_z and A_u, respectively. Examine the following alternatives.

If z_i and v_i are not adjacent, rename z_i as w_i. Similarly, if u_i and v_{+1} are not adjacent, let w_i be the vertex u_i. In any of these two alternatives, it follows that $N(w_i) \cap K_t = K_t \setminus \{v_i, v_{i+1}\}$. The latter means that Condition (i) of Definition 3 holds. When none of the above alternatives occurs, the arcs A_z and A_u intersect, because s_z preceeds t_u in (t_{i+1}, s_i). That is, z_i and w_i are adjacent vertices satisfying $N(z_i) \cap K_t = K_t \setminus \{v_{i+1}\}$ and $N(u_i) \cap K_t = K_t \setminus \{v_i\}$. This corresponds

to Condition (ii) of Definition 3. Consequently, for any pair of vertices $v_i, v_{i+1} \in K_t$ it is always possible to select a vertex $w_i \notin K_t$, or a pair of vertices $z_i, u_i \notin K_t$, so that Definition 3 is satisfied. That is, G contains an obstacle as an induced subgraph. This contradiction means all stable models of G are HCA.

The implications (c) $\Rightarrow$ (d) and (d) $\Rightarrow$ (*a*) are trivial, meaning that the proof is complete. $\triangle$

5 Constructing Stable Models

Motivated by the characterizations of HCA graphs in terms of stable models, described in the previous section, we present below an algorithm for constructing a stable model of a CA graph. Let $(C, \mathcal{A})$ be a CA model of some graph G, and $A_1, \ldots, A_n$ the circular ordering of the arcs of $\mathcal{A}$. Define the following expansion operations on the end points t_j and start points s_i of $\mathcal{A}$.

expansion(t_j):
Examine the extremes points of $\mathcal{A}$, starting from t_j, in the clockwise direction, and choosing the closest start point s_i satisfying $i = j$ or $A_i \cap A_j = \emptyset$. Then move t_j so as to become the extreme point preceeding s_i in the model.

expansion(s_i):
First, examine the extreme points of $\mathcal{A}$, starting from s_i, in the counterclockwise direction, and choosing the closest end point t_j satisfying $i = j$ or $A_i \cap A_j = \emptyset$. Let $T = SEQUENCE(t_j)$. Then move s_i counterclocwise towards T, transforming T into the sequences $T' s_i T''$, where $T' = \{t_j \in T | i = j$ or $A_i \cap A_j = \emptyset\}$ and $T'' = T \setminus T'$.

The following lemma asserts that the intersections of the arcs are preserved by these operations.

Lemma 1. *The operations expansion*(t_j) *or expansion*(s_i) *applied to a model* $(C, \mathcal{A})$ *construct models equivalent to* $(C, \mathcal{A})$.

We describe the following algorithm for finding a stable model of a given CA model, with end points t_j and start points s_i:

1. Perform *expansion*(t_j), for $j = 1, \ldots, n$.
2. Perform *expansion*(s_i), for $i = 1, \ldots, n$.

The correctness of this algorithm then follows from Lemma 1 and from the following theorem.

Theorem 3. *The model constructed by the above algorithm is stable.*

Proof. Let $(C, \mathcal{A})$ be a given CA model, input to the algorithm. We show that all its start points are stable, at the end of the process. After the completion of Step 1, we know that $s_i = FIRST(S)$ is already stable, for any s-sequence S. Otherwise, there would exist some end point $t_j \in NEXT^{-1}(S)$ satisfying

$i \neq j$ and $A_i \cap A_j \neq \emptyset$, meaning that t_j would have been moved after s_i in the clockwise direction, by $expansion(t_j)$.

Next, examine Step 2. Choose a start point s_i and follow the operation $expansion(s_i)$. If s_i is already stable, the algorithm does nothing. Suppose s_i is not stable. Let $S^* = SEQUENCE(s_i)$ and S the s-sequence closest to S^* in the counterclockwise direction, where $T = NEXT^{-1}(S)$ contains some t_j satisfying $i = j$ or $A_i \cap A_j = \emptyset$. Then $expansion(s_i)$ transforms T into the sequences $T's_iT''$, where $T' = \{t_j \in T | i = j \text{ or } A_i \cap A_j = \emptyset\}$ and $T'' = T \setminus T'$. Analyze the new sequences that have been formed. Clearly, $T' \neq \emptyset$, otherwise s_i would have been moved further from S^*. On the other hand, T'' could possibly be empty. However, the latter would only imply that T remains unchanged and that s_i has been incorporated to S. In any case, T' is the t-sequence which preceeds s_i. By the construction of T', it follows that s_i is now stable. In addition, previously stable start points of S remain so, because $T'' \subset T$. Furthermore, observe that $s_i \neq FIRST(S^*)$, because $FIRST(S^*)$ was before stable, whereas s_i was not. Consequently, S^* does not become empty by moving s_i out of it, implying that no parts of distinct t-sequences can be merged during the process. The latter assertion preserves the stability of the stable vertices belonging to the s-sequence which follows S^* in C. The remaining start points are not affected by $expansion(s_i)$. Consequently, s_i becomes now stable and all previousloy stable start points remain so. The algorithm is correct. △

Corollary 2. *Every CA model admits an equivalent stable model.*

Next, we discuss the complexity of the algorithm. The number of extreme points examined during the operation $expansion(t_j)$ is at most $d(v_j) + 1$, since the operation stops at the first extreme t_i, such that either $i = j$ or $A_i \cap A_j = \emptyset$. Consequently, Step 1 requires $O(m)$ time. As for the operation $expansion(s_i)$, we divide it into two parts. First, for finding the required s-sequence S, the above argument applies, that is, $O(m)$ time suffices for all s_i. As for the determination of the sequences T' and T'', a straightforward implementation of it would consist of examining the entire t-sequence $T = T' \cup T''$, for each corresponding s_i, meaning $O(n^2)$ time, overall. However, a more elaborate implementation is possible, as follows.

To start, after the completion of Step 1, order the end points of each t-sequence T, in reverse ordering of their corresponding start points. That is, if $t_j, t_k \in T$ then in the clockwise direction, the extreme points of A_j and A_k appear as $\ldots t_j \ldots t_k \ldots s_k \ldots s_j \ldots$. Such an ordering can be obtained in overall $O(n)$ time. With the end points so ordered, when traversing $T = NEXT^{-1}(S)$, for completing the operation $expansion(s_i)$, we can stop at the first $t_j \in T$ satisfying $i = j$ or $A_i \cap A_j = \emptyset$. In case the condition $i = j$ holds, we exchange in T, the positions of t_j and $FIRST(T)$. Afterwards, in any of the two alternatives, move s_i to the position just before t_j in the counterclockwise direction. We would need no more than additional $d(v_i) + 1$ steps for it, in the worst case. Consequently, $expansion(s_i)$ can be completed in $O(m)$ time, for all start points. Therefore the complexity of the algorithms is $O(m)$.

6 Recognition Algorithm for HCA Graphs

We are now ready to formulate the algorithm for recognizing HCA graphs. Let G be a graph.

1. Apply the algorithm [7] to recognize whether G is a CA graph. In the affirmative case, let $(C, \mathcal{A})$ be the model constructed by [7]. Otherwise terminate the algorithm (G is not HCA).
2. Transform $(C, \mathcal{A})$ into a stable model, applying the algorithm of Section 5.
3. Verify if $(C, \mathcal{A})$ is a HCA model, applying the algorithm of Section 3. Then terminate the algorithm (G is HCA if $(C, \mathcal{A})$ is HCA, and otherwise G is not HCA).

The correctness of the algorithm follows directly from Theorems 1, 2 and 3.

Each of the above steps can be implemented in $O(m)$ time. The complexity of the algorithm is $O(m)$.

The algorithm constructs a HCA model of the input graph G, in case G is HCA. If G is CA but not HCA, we can exhibit a certificate of this fact, by showing a forbidden subgraph of G, that is, an obstacle. In order to construct the obstacle, we may need certificates of non co-chordality. There is no difficulty to modify the algorithm [4] so as to produce such certificates. The entire process can also be implemented in linear time.

References

1. X. Deng and P. Hell and J. Huang, Linear time representation algorithms for proper circular-arc graphs and proper interval graphs, *SIAM J. Computing*, **25** (1996), pp. 390-403.
2. F. Gavril, Algorithms on circular-arc graphs, *Networks* **4** (1974), pp. 357-369.
3. M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, 1980, 2nd ed. 2004.
4. M. Habib, R. M. McConnell, C. Paul, and L. Viennot, Lex-bfs and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing, *Theoretical Computer Science*, **234** (2000), pp. 59-84.
5. H. Kaplan and Y. Nussbaum, A Simpler Linear-Time Recognition of Circular-Arc Graphs, accepted for publication in *10th Scandinavian Workshop on Algorithm Theory* (2006).
6. M. C. Lin and J. L. Szwarcfiter, Efficient Construction of Unit Circular-Arc Models, *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms* (2006), pp. 309-315.
7. R. M. McConnell, Linear-time recognition of circular-arc graphs, *Algorithmica* **37 (2)** (2003), pp. 93-147.
8. J. Spinrad, Efficient Graph Representations, *American Mathematical Society* (2003).
9. A. Tucker, Characterizing circular-arc graphs, *Bull. American Mathematical Society* **76** (1970), pp. 1257-1260.

SIAM J. DISCRETE MATH.
Vol. 22, No. 1, pp. 409–423

UNIT CIRCULAR-ARC GRAPH REPRESENTATIONS AND FEASIBLE CIRCULATIONS*

MIN CHIH LIN† AND JAYME L. SZWARCFITER‡

Abstract. In a recent paper, Durán et al. [*J. Algorithms*, 58 (2006), pp. 67–78] described an algorithm of complexity $O(n^2)$ for recognizing whether a graph G with n vertices and m edges is a unit circular-arc (UCA) graph. Furthermore, the following open questions were posed in the above paper: (i) Is it possible to construct a UCA model for G in polynomial time? (ii) Is it possible to construct a UCA model, whose extremes of the arcs correspond to integers of polynomial size? (iii) If (ii) is true, could such a model be constructed in polynomial time? In the present paper, we describe a characterization of UCA graphs, based on network circulations. The characterization leads to a different recognition algorithm and to answering these questions in the affirmative. We construct a UCA model whose extremes of the arcs correspond to integers of size $O(n)$. The proposed algorithms, for recognizing UCA graphs and constructing UCA models, have complexities $O(n+m)$. Furthermore, the complexities reduce to $O(n)$, if a proper circular-arc (PCA) model of G is already given as the input, provided the extremes of the arcs are ordered. We remark that a PCA model of G can be constructed in $O(n+m)$ time, using the algorithm by Deng, Hell, and Huang [*SIAM J. Comput.*, 25 (1996), pp. 390–403]. Finally, we also describe a linear time algorithm for finding feasible circulations in networks with nonnegative lower capacities and unbounded upper capacities. Such an algorithm is employed in the model construction for UCA graphs.

Key words. algorithm, circular-arc graph, circular-arc model, circulations, networks, proper circular-arc graph, unit circular-arc graph

AMS subject classifications. 05C62, 05C85, 05C05, 68R10, 90B10

DOI. 10.1137/060650805

1. Introduction.

In a recent paper, Durán et al. [2] described an algorithm of complexity $O(n^2)$ for recognizing whether a graph G, with n vertices and m edges, is a unit circular-arc (UCA) graph. Furthermore, the following open questions were posed in [2]:

(i) Is it possible to construct a UCA model for G in polynomial time?
(ii) Is it possible to construct a UCA model, whose extreme points of the arcs correspond to integers of polynomial size?
(iii) If (ii) is true, could such a model be constructed in polynomial time?

Problems (ii) and (iii) were also proposed in the book by Spinrad [10]. As for problem (i), the proof of the characterization of UCA graphs in terms of forbidden subgraphs by Tucker [13], actually contains an algorithm for constructing a UCA model. However, due to the possible manipulation of large integers, the complexity of this algorithm is unknown, and so far the construction of a UCA model in polynomial time remains unsolved [2, 10].

*Received by the editors January 24, 2006; accepted for publication (in revised form) November 12, 2007; published electronically March 7, 2008.
http://www.siam.org/journals/sidma/22-1/65080.html

†Departamento de Computación, Universidad de Buenos Aires, Facultad de Ciencias Exactas y Naturales, Buenos Aires 1428, Argentina (oscarlin@dc.uba.ar). This author's research was partially supported by UBACyT grants X184 and X212, PICT ANPCyT grant 11-09112, and CNPq under PROSUL project Proc. 490333/2004-4.

‡Universidade Federal do Rio de Janeiro, Instituto de Matemática, NCE and COPPE, Caixa Postal 2324, 20001-970 Rio de Janeiro, Brazil (jayme@nce.ufrj.br). This author's research was 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, Brazil.

In the present paper, we propose a characterization for UCA graphs, which leads to a different recognition algorithm and to answering in affirmative the previous three questions. An extended abstract of the present paper appeared in [7]. The proposed algorithm recognizes UCA graphs and constructs UCA models whose extremes of the arcs correspond to integers of $O(n)$ size, in overall time $O(n+m)$. Furthermore, if the graph G is already given by a proper circular-arc (PCA) model of it, then the complexity of the proposed algorithm reduces to $O(n)$, provided the extremes of the arcs are ordered. Observe that a PCA model for G, when given by its vertices and edges, can be constructed in $O(n+m)$ time, employing the algorithm by Deng, Hell, and Huang [1]. Recall that UCA graphs are PCA graphs.

Our method employs network circulations. We formulate an algorithm finding a feasible circulation in a network with nonnegative lower capacities and unbounded upper capacities. The algorithm has linear time complexity in the size of the network.

Denote by G an undirected graph, with vertex set $V(G)$ and edge set $E(G)$. For an edge $e \in E(G)$, write $e = uv$, where u and v are the *extremes* of e. Denote by $d_G(v)$ for the *degree* of v in G. We may also simply write $d(v)$, instead. Use a similar notation for a digraph D. For a directed edge $e = uv \in E(D)$, say that u is the *start* and v the *end* of e. Write $d_G^-(v)$ and $d_G^+(v)$ for the *indegree* and *outdegree* of vertex v, respectively. Again, also write $d^-(v)$ and $d^+(v)$, simply. Moreover, $E^-(v)$, $E^+(v) \subseteq E(D)$, represent the set of edges of D entering and leaving v, respectively. Also, D is *connected* when its underlying (undirected) graph is connected. Say that D is *strongly connected* when D contains paths from u to v and from v to u, for any $u, v \in V(D)$. Finally, the *bridge* of D is an edge contained in no (directed) cycle of D.

An *out-arborescence* (*in-arborescence*) T of D is a spanning connected subdigraph of D such that there is a distinguished vertex $v^* \in V(T)$, called the *root* of T, satisfying $d_T^-(v^*) = 0$ ($d_T^+(v^*) = 0$), while $d_T^-(v) = 1$ ($d_T^+(v) = 1$) for all of the remaining vertices $v \neq v^*$ of T. Clearly, a strongly connected digraph admits both an out-arborescence and in-arborescence, with root at any arbitrary vertex. When $d_T^+(v) = 0$ ($d_T^-(v) = 0$), call v a *leaf* of T. For $u, v \in V(T)$, if T contains a path from u to v, then u is an *ancestor* of v, and v a *descendant* of u. A *leaf-root ordering* of T is a sequence $v_1, \ldots, v_n$ of its vertices, such that $i < j$ implies v_i is not an ancestor (descendant) of v_j in T.

A *network* is a digraph D, having real values b_j, c_j associated to each edge $e_j \in E(D)$. Call b_j the *lower capacity* of e_j, while c_j is the *upper capacity* of e_j. A *circulation* of D is a function W assigning a real number w_j to each edge e_j of D, called the *flow of* e_j. Denote $w^-(v_i) = \sum_{w_j \in E^-(v_i)} w_j$ and $w^+ = \sum_{w_j \in E^+(v_i)} w_j$, for each $v_i \in V(D)$. A circulation is *feasible* when it satisfies

$$w^-(v_i) = w^+(v_i), \text{ for each } v_i \in V(D), \text{ and } b_j \leq w_j \leq c_j, \text{ for each } e_j \in E(D).$$

For our purposes, we only consider networks with finite nonnegative lower capacities and flow values, and unbounded the upper capacities. In this case, the condition $w_j \leq c_j$ is always satisfied.

A *circular-arc* (CA) model for a graph G is a pair $(C, \mathcal{A})$, where C is a circle and $\mathcal{A}$ is a collection of arcs of C, such that each arc $A_i \in \mathcal{A}$ corresponds to a vertex $v_i \in V(G)$, and any A_i, A_j intersects precisely when v_i, v_j are adjacent in G, $1 \leq i, j \leq n$, and $i \neq j$. A CA graph is one admitting a CA model. When no arc of $\mathcal{A}$ properly contains another arc of $\mathcal{A}$ then $(C, \mathcal{A})$ is a PCA model, while if all arcs of $\mathcal{A}$ have the same length, then $(C, \mathcal{A})$ is a UCA model. A PCA (UCA) graph is one admitting a PCA (UCA) model. A *normal* model is a PCA model where no two

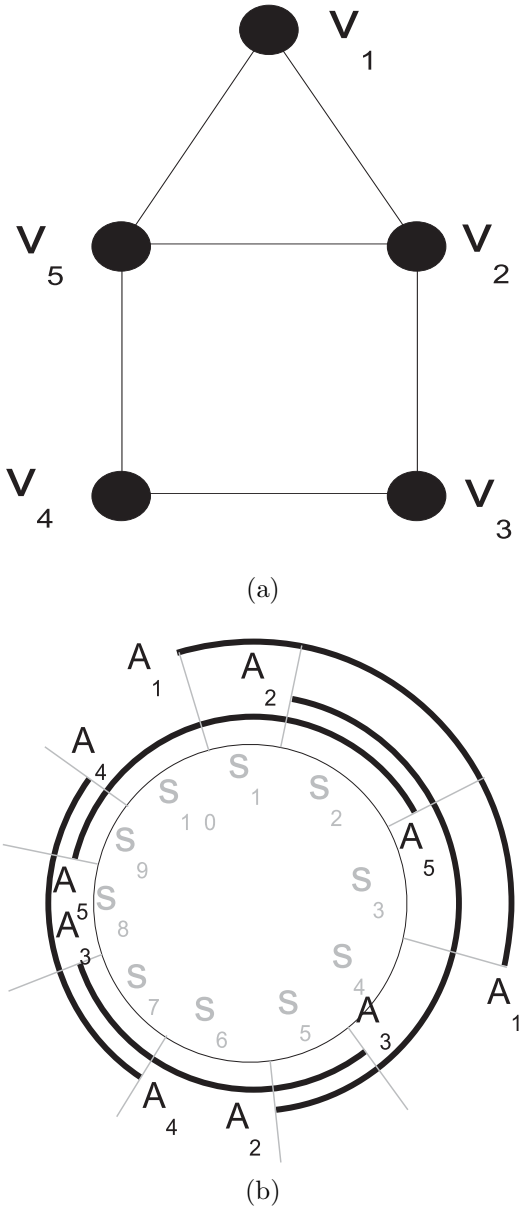


FIG. 1. *A PCA graph and a normal model of it.*

arcs cover the circle. Figure 1(a) depicts a PCA graph and a normal model of it is in Figure 1(b). When traversing the circle C, always choose the clockwise direction. If s, t are points of C, then 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* and t the *end* of the arc. The *extremes* of $\mathcal{A}$ are the extremes of $A_i \in \mathcal{A}$. For $A_i \in \mathcal{A}$, write $A_i = (s_i, t_i)$. We assume the labelling $A_1, \ldots, A_n$ of the arcs is such that the sequence of the start points $s_1, \ldots, s_n$ is in the circular ordering of C. For $p, q \in A_i$, if $(s_i, p) \subseteq (s_i, q)$, then p *precedes* q, and q *succeeds* p in A_i. For $p, q, t \in C$, write $\max\{p, q\}_t$ to represent the

point, p or q, which is farthest from t, in the circular ordering of C. If p is an extreme of $\mathcal{A}$, then denote by $PRED(p)$ and $SUC(p)$, the extreme of $\mathcal{A}$, which immediately precedes and succeeds p, in the circular ordering, respectively. We assume that all arcs of C are open, no two extremes of distinct arcs of $\mathcal{A}$ coincide, and no single arc covers C. An arc of C defined by two consecutive extremes of $\mathcal{A}$ is a *segment* of $(C, \mathcal{A})$. Clearly, C is the union of the $2n$ segments of $(C, \mathcal{A})$ and the extreme points. Figure 1(b) shows the ten segments of the corresponding CA model.

The first characterization of CA graphs leading to a polynomial time recognition algorithm is due to Tucker [12, 14]. Subsequently, faster algorithms have been described by Hsu [6] and by Eschen and Spinrad [3]. More recently, a linear time recognition algorithm for CA graphs has been formulated by McConnell [8]. Deng, Hell, and Huang [1] described an algorithm for recognizing PCA graphs in linear time. This algorithm also constructs a corresponding PCA model for the graph. As for UCA graphs, the first polynomial time recognition algorithm is that by Durán et al. [2].

Given a graph G, the algorithm [2] initially employs algorithm [1] to construct a PCA model $(C, \mathcal{A})$ of G, if any. Clearly, G is not UCA if such a model does not exist. Afterwards, this algorithm proceeds with two main phases: The first of them is to transform $(C, \mathcal{A})$ into a normal model. The last phase is the actual algorithm for deciding if G is a UCA graph, employing the normal model constructed in the first phase. The complexities of the algorithms corresponding to these two phases are both $O(n^2)$. The method proposed in the present paper is also composed by similar phases. However, the complexities of the two corresponding algorithms are $O(n)$.

In section 2, we describe the characterizations in which are based the proposed algorithms. The characterization of UCA graphs relates them to circulations in special networks. Section 3 describes the algorithm for finding feasible circulations in general networks, having nonnegative lower capacities and unbounded upper capacities. Section 4 presents the algorithm for constructing normal models. Section 5 contains the actual algorithm for recognizing UCA graphs and constructing UCA models. Further comments form the last section.

2. A characterization for UCA graphs. In this section, we characterize UCA graphs in terms of circulations of a special network. The following two theorems are basic to our purposes.

THEOREM 1 (see Golumbic [4] and Tucker [13]). *Let G be a PCA graph. Then G admits a normal model.*

THEOREM 2 (see Tucker [13]). *Let G be a UCA graph and $(C, \mathcal{A})$ a normal model of it. Then G admits a UCA model, such that its extremes are in the same circular ordering as those of $(C, \mathcal{A})$.*

Let G be a CA graph and $(C, \mathcal{A})$ a CA model of it. Denote by $S_1, \ldots, S_{2n}$ the segments of $(C, \mathcal{A})$ in circular ordering, where the start of S_1 and that of A_1 coincide. Denote by l_j the length of S_j. Clearly, any arc $A_i \in \mathcal{A}$ may be decomposed into the segments which form it. The length of A_i equals $\sum_{S_j \subseteq A_i} l_j$. The decomposition allows one to represent relations between lengths of arcs by relations between lengths of the corresponding segments. Consequently, the condition of equality between any two arc lengths required by a UCA model can be expressed by a system of $n-1$ linear equations q_i, together with $2n$ inequalities, called the *full system* of $(C, \mathcal{A})$:

$$q_i : \sum_{S_j \subseteq A_i} l_j = \sum_{S_j \subseteq A_{i+1}} l_j,\ i = 1, \ldots, n-1, \tag{2.1}$$

$$l_j > 0,\ j = 1, \ldots, 2n. \tag{2.2}$$

Clearly, equation q_i corresponds to the condition that the length of arc A_i equals that of A_{i+1}. The segment lengths l_j are the variables of such a system of equations. Our aim is to find a solution of the full system, if existing. The values of l_j, obtained from the solution of the system, would define a UCA model for G. In equation q_i, call the term $\sum_{S_j \subseteq A_i} l_j$ the *left side* of q_i, while $\sum_{S_j \subseteq A_{i+1}} l_j$ is its *right side*. Each equation q_i can possibly be simplified if the length l_j of a same segment appears in both the left and right sides of q_i. In this case, subtract l_j from both sides of q_i. The equations so obtained form the *reduced system* of $(C, \mathcal{A})$. The following lemma describes a useful property of these systems.

LEMMA 3. *Let G be a PCA graph, $(C, \mathcal{A})$ a normal model of it, and R the reduced system of $(C, \mathcal{A})$. Then each segment length l_j of $(C, \mathcal{A})$ appears at least once in R, unless S_j is a segment contained in all arcs of $\mathcal{A}$ or in none of them. Furthermore, l_j appears at most twice in R. In the latter situation, the two instances of l_j in R occur in different sides of their equations.*

Proof. Let S_j be any segment of the normal model $(C, \mathcal{A})$. Denote by $\mathcal{A}'$ the set of arcs of $\mathcal{A}$ covering S_j. If $\mathcal{A}' = \emptyset$, then no arc of $\mathcal{A}$ contains S_j, implying that l_j does not appear in R, and the lemma is satisfied. Consider $\mathcal{A}' \neq \emptyset$. We know that $\mathcal{A}'$ does not entirely cover C, otherwise there would be two arcs of $\mathcal{A}$ which would do so, contradicting $(C, \mathcal{A})$ to be normal. Consequently, the circular ordering of the extremes of $(C, \mathcal{A})$, when restricted to the arcs of $\mathcal{A}'$, is a linear ordering. Let A_i and A_k be the first and last arcs of $\mathcal{A}'$, respectively, in the considered linear ordering.

First, let $i \leq k$. Observe that l_j is not on the right side of any of the equations $q_i, \ldots, q_k$, of R. Because if l_j is on the right side of q_t, then $k \geq t \geq i$ in the full system. However, l_j is also on the left side of q_t, hence it gets simplified in R. However, l_j is on the right side of q_{i-1} in R provided $i > 1$. Similarly, l_j is not on the left side of any $q_i, \ldots, q_{k-1}$ but it is at the left of q_k, provided $k < n$. On the other hand, $i = 1$ and $k = n$ imply that S_j is common to all arcs of $\mathcal{A}$. Consequently, each l_j appears at least once in R, unless S_j is contained in all arcs of $\mathcal{A}$, or in none of them.

Consider the alternative $i > k$. If $i - 1 > k$, then l_j appears on the left of q_{i-1} and on the right of q_k. There are no other occurrences of l_j. Finally, when $i - 1 = k$ it follows that S_j is common to all arcs of $\mathcal{A}$. Again, each l_j always appears at least once in R, except when S_j is contained in all arcs of $\mathcal{A}$. Furthermore, it appears at most twice in R. In the latter situation, l_j occurs on different sides of the corresponding equations. □

Aiming to solve R efficiently, we describe a graph theoretical model for it. Define the *segment digraph* D, as follows. There is a vertex $v_i \in V(D)$ for each equation q_i of R, and one edge e_j for each segment S_j of $(C, \mathcal{A})$. In addition, there is a distinguished vertex $v_0 \in V(D)$. Each edge $e_j \in E(D)$ is directed according to:

(i) If l_j appears at the left of some equation q_i of R, then e_j starts at v_i; otherwise it starts at v_0.

(ii) If l_j appears at the right of some equation q_k, then e_j ends at v_k; otherwise it ends at v_0.

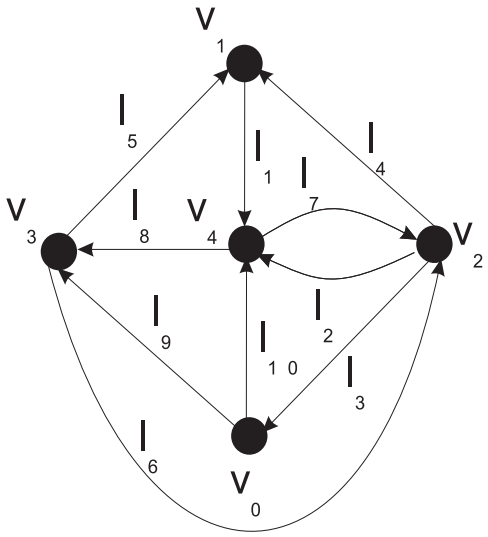
The description of D is complete. Clearly, D has n vertices and $2n$ edges and no loops, except possibly at v_0.

By adding to each edge $e_j \in E(D)$ a lower capacity $b_j = 1$ and an unbounded upper capacity c_j, we obtain the *segment network* of D.

As an example, the reduced system of the PCA model of Figure 1(b) is illustrated in Figure 2(a), while the corresponding segment digraph is in Figure 2(b). Similarly,

$$
R \quad \begin{array}{ll} q_1\colon & l_1 = l_4 + l_5 \\ q_2\colon & l_2 + l_3 + l_4 = l_6 + l_7 \\ q_3\colon & l_5 + l_6 = l_8 + l_9 \\ q_4\colon & l_7 + l_8 = l_{10} + l_1 + l_2 \end{array}
$$

(a)



(b)

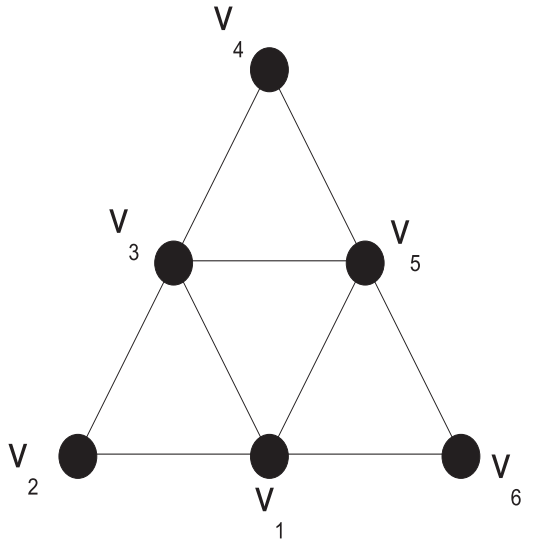
FIG. 2. *The reduced system of Figure* 1(b) *and its segment digraph.*

the reduced system and segment digraph corresponding to the PCA model of Figure 3(b) appear in Figures 4(a) and 4(b), respectively.

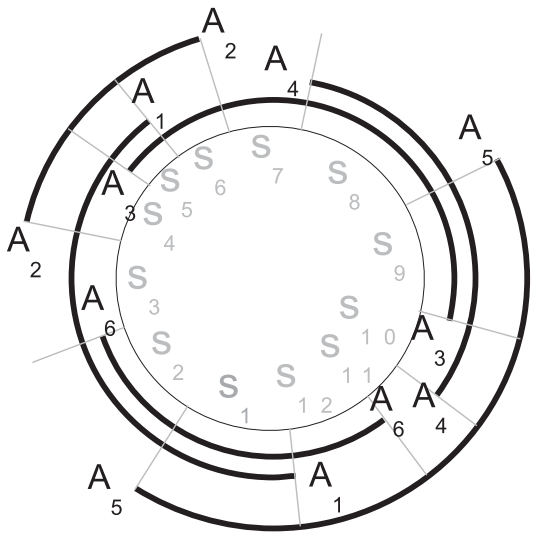
The following theorem characterizes UCA graphs in terms of feasible circulations.

THEOREM 4. *Let G be a PCA graph, $(C, \mathcal{A})$ a normal model of it, and D its segment network. Then G is a UCA graph if and only if D admits a feasible circulation.*

Proof. Let G be a UCA graph, and $(C, \mathcal{A})$ a normal model of G. By Theorem 2, there exists a UCA model $(C', \mathcal{A}')$ of G such that the endpoints of the arcs of $\mathcal{A}$ and $\mathcal{A}'$ are in the same ordering. That is, $(C, \mathcal{A})$ and $(C', \mathcal{A}')$ differ only by the lengths of the corresponding segments. Let R be the reduced system of $(C, \mathcal{A})$ and D be the segment network. Define a circulation W by assigning a flow w_j to each edge e_j of D, equal to the length l'_j of the segment of $(C', \mathcal{A}')$, corresponding to S_j. We show that such an assigment W is a circulation of D. Let $v_i \in V(D)$, $1 \leq i \leq n-1$, and q_i the corresponding equation in R. Then $w^-(v_i) = \sum_{S_j \subseteq A_i \setminus A_{i+1}} l'_j$. Similarly, $w^+(v_i) = \sum_{S_j \subseteq A_{i+1} \setminus A_i} l'_j$. Because $(C', \mathcal{A}')$ is a UCA model, the lengths of arcs A'_i and A'_{i+1} of $\mathcal{A}'$ are the same, implying that $w^-(v_i) = w^+(v_i)$. It remains to show that this equality also holds for v_0. This assertion follows from the fact that $\sum_{0 \leq i \leq n-1} w^-(v_i) = \sum_{0 \leq i \leq n-1} w^+(v_i)$, because every edge e_j counts l'_j units both in $\sum_{0 \leq i \leq n-1} w^-(v_i)$ and $\sum_{0 \leq i \leq n-1} w^+(v_i)$. Since $w^-(v_i) = w^+(v_i)$, for each $1 \leq i \leq n-1$, it follows that $w^-(v_0) = w^+(v_0)$. Consequently, W is indeed a feasible circulation of D.



(a)



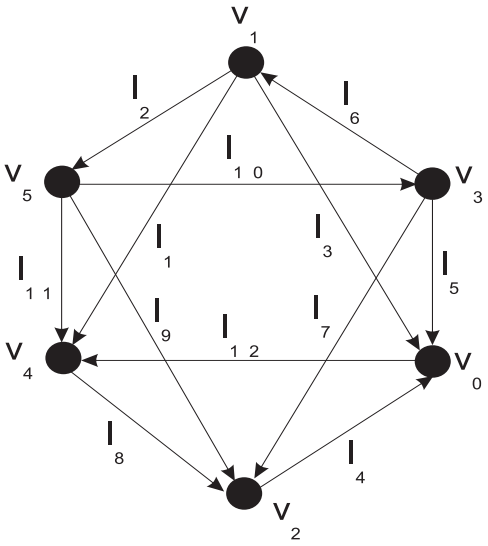
(b)

FIG. 3. *Another PCA graph with its normal model.*

Conversely, by hypothesis, D admits a feasible circulation W. We prove that G is UCA by constructing a UCA model $(C', \mathcal{A}')$ for G. First, consider the situation where no segment S_j of $(C, \mathcal{A})$ is contained in all arcs of $\mathcal{A}$ or in none of them. Let w_j be the flow of edge e_j of D. Construct the model $(C', \mathcal{A}')$ of G, by maintaining the endpoints of the arcs of $\mathcal{A}'$ in the same circular ordering as those in $\mathcal{A}$, while possibly modifying the lengths of the segments. The length of C' is defined as $\sum_{e_j \in E(D)} w_j$, while the length of the segment of $(C', \mathcal{A}')$ corresponding to S_j is set to the flow w_j of e_j. We show that the model so constructed is a UCA model.

$$
R \quad \begin{array}{ll} q_1\colon & l_1 + l_2 + l_3 = l_6 \\ q_2\colon & l_4 = l_7 + l_8 + l_9 \\ q_3\colon & l_5 + l_6 + l_7 = l_{10} \\ q_4\colon & l_8 = l_{11} + l_{12} + l_1 \\ q_5\colon & l_9 + l_{10} + l_{11} = l_2 \end{array}
$$

(a)



(b)

FIG. 4. *The reduced system of Figure* 3(b) *and its segment digraph.*

Let R be the reduced system of $(C, \mathcal{A})$. We show that the assigment $l_j := w_j$ for each segment S_j of $(C, \mathcal{A})$ is a solution to R. Let q_i be an equation of R and v_i the corresponding vertex of D. By the construction of D, the left side of q_i corresponds to the flow values of the edges of D ending at v_i, while the right side of q_i corresponds to the flow values of the edges, starting at v_i. Because D is feasible, $w^-(v_i) = w^+(v_i)$, implying that $\sum_{S_j \subseteq A_i} l_j = \sum_{S_j \subseteq A_{i+1}} l_j$. Moreover, all l_j are positive, because the lower capacities are satisfied. Consequently, W is a solution to R. Clearly, any solution to R is also a solution to the full system. On the other hand, because no segment is contained in all arcs of $\mathcal{A}$ or in none of them, Lemma 3 implies that all segment lengths appearing in the full system also appears in the reduced system, and consequently have been assigned a required value, The latter implies that all arc lengths of $\mathcal{A}'$ are equal, meaning that $(C', \mathcal{A}')$ is a UCA model.

Finally, examine the situation where there is a segment S_j contained in all arcs of $\mathcal{A}$ or in none of them. Construct $(C', \mathcal{A}')$ as above, and assign any positive length to S_j. Clearly, all arc lengths of $\mathcal{A}'$ are equal and $(C', \mathcal{A}')$ is a UCA model. □

The above theorem implies that the problem of recognizing if a given graph G is UCA is equivalent to deciding if the corresponding segment network admits a feasible circulation. We describe a characterization for the existence of circulations in general networks, with arbitrary real nonnegative lower capacities and unbounded upper ca-

pacities. This is a special case of Hoffman's circulation theorem, as below. See [9]. For a subdigraph B of D, write $b^-(B) = \sum_{e_j \in E^-(B)} b_j$ and $c^+(B) = \sum_{e_j \in E^+(B)} c_j$

THEOREM 5 (see Hoffman [5]). *Let D be a network having an arbitrary real lower capacity b_j and upper capacity c_j, for each edge $e_j \in E(D)$. Then D admits a feasible circulation W if and only if $b^-(B) \leq c^+(B)$, for each subdigraph B of D. Moreover, if b and c are integers, then the flow values can be taken as integers.*

COROLLARY 6. *Let D be a network with real nonnegative lower capacities and unbounded upper capacities. Then D admits a feasible circulation if and only if all bridges of D have lower capacity zero. In this case, a circulation with integer flow values always exists.*

Proof. Suppose D admits a feasible circulation W. To the contrary, assume that D has a bridge e_j with positive lower capacity. Then $e_j \in E^+(B), E^-(B')$ for some distinct strongly connected components B, B' of D. In this situation, we know that $w(E^-(B')) > 0$ and $w(E^+(B')) > 0$. The latter implies that there exists some edge e_k leaving B' and enters a strongly connected component $B'' \neq B, B'$, such that $w_k > 0$. Repeating this argument over and over leads to the contradiction that D is not finite. Consequently, no such e_j may exist.

Conversely, by hypothesis all bridges of D have lower capacity equal to zero. Let B be any strongly connected component of D. Then any proper subdigraph B' of B satisfies $E^-(B'), E^+(B') \neq \emptyset$. Since the upper capacities of D are unbounded and the lower capacities are not, it follows that $b^-(B') < c^+(B')$. Using Theorem 5, we conclude that B admits a feasible circulation, then repeat this argument for the remaining strongly connected components of D. Finally, assign the flow value equal to zero to all bridges of D. We have so obtained a feasible circulation. By simply replacing each lower capacity b_j by the value $\lceil b_j \rceil$, we obtain a feasible circulation with integer flow values. □

As examples, observe that the segment digraph of Figure 2(b) is strongly connected, while that of Figure 4(b) is not. Consequently, the graph of Figure 1(a) is a UCA graph, while that of Figure 3(a) is not.

3. Finding feasible circulations. Let D be a network having finite real nonnegative lower capacities and unbounded upper capacities. We describe an algorithm which finds a feasible circulation W of D, or reports that such circulation does not exist. For each edge e_j of D, let b_j be the given nonnegative lower capacity. The algorithm below computes integer flow values w_j of the feasible circulation W of D.

1. Find the strongly connected components and the bridges of D. If any bridge has a positive lower capacity, then report that no feasible circulation exists and stop. Otherwise, set the flow value $w_j := 0$ for each bridge edge e_j of D. Then perform step 2 for each strongly connected component B of D. At termination, stop.
2. Assign the initial flow values $w_j := \lceil b_j \rceil$, for each edge $e_j \in E(B)$. Then compute $w^-(v_i) = \sum_{e_j \in E^-(v_i)} w_j$ and $w^+(v_i) = \sum_{e_j \in E^+(v_i)} w_j$ for each $v_i \in V(B)$. Finally, run *OUT PROCEDURE* and afterwards run *IN PROCEDURE*.

   OUT PROCEDURE. Choose an arbitrary vertex $v^* \in V(B)$ and find an out-arborescence T of B, with root v^*. Repeat the following operations, for each vertex $v \in V(T)$, $v \neq v^*$, in leaf-root ordering: if $w^-(v) < w^+(v)$ then set $w^+(u) := w^+(u) + w^+(v) - w^-(v)$, $w_j := w_j + w^+(v) - w^-(v)$ and $w^-(v) := w^+(v)$, where $e_j = uv$ is the edge of T ending at v.

IN PROCEDURE. Construct an in-arborescence T of B, with the same root v^*, as chosen in the computation of the *OUT PROCEDURE* for B. Repeat the following operations, for each vertex $v \in V(T)$, $v \neq v^*$, in leaf-root ordering: if $w^+(v) < w^-(v)$, then set $w^-(u) := w^+(u) + w^-(v) - w^+(v)$, $w_j := w_j + w^-(v) - w^+(v)$, and $w^+(v) := w^-(v)$, where $e_j = vu$ is the edge of T starting at v.

Next, we assert the correctness of the algorithm.

THEOREM 7. *The above algorithm finds a feasible circulation for D.*

Proof. From Corollary 6, it follows that the algorithm correctly reports the nonexistence of a circulation, whenever there exists a bridge having a positive lower capacity. Otherwise, let B be any strongly connected component of D. We show that the flow values assigned by the algorithm to the edges of B define a feasible circulation for B. Furthermore, since the bridges are assigned a zero flow, the latter implies that the circulation of D so obtained is also feasible.

First, observe that since B is strongly connected it admits both an out-arborescence and an in-arborescence, rooted at the same arbitrary vertex $v^* \in V(B)$. Classify the vertices v_i of D into three types, according to the initial values of $w^-(v_i)$ and $w^+(v_i)$. Say that a vertex v is *in-deficient* when $w^-(v) < w^+(v)$, *out-deficient* when $w^+(v) < w^-(v)$, and *balanced* when $w^-(v) = w^+(v)$.

Consider the computation of *OUT PROCEDURE* and examine the vertices of the chosen out-arborescence T of B, in the ordering chosen by the algorithm. Let $v \neq v^*$ be the vertex next to be visited, in the considered ordering. If v is in-deficient, then the algorithm chooses the edge e_j of T ending at v and modify its weight from w_j to the value $w_j + w^+(v) - w^-(v)$. It is clear that v becomes balanced. Moreover, since T contains exactly one edge e_j entering v, we can assure that v remains balanced until the end of *OUT PROCEDURE*. Consequently, B does not contain in-deficient vertices at the completion of the procedure, except possibly v^*. Similarly, after the completion of *IN PROCEDURE*, B does not contain out-deficient vertices, except possibly v^*. Furthermore, the computation of *IN PROCEDURE* cannot create in-deficient vertices. Consequently, after completing the visits to the second tree, we can assure that all vertices are balanced, except possibly v^*. However, $w_j = 0$, for every bridge e_j of D. Consequently, $\sum_{v \in V(B)} w^-(v) = \sum_{v \in V(B)} w^+(v) = \sum_{e_j \in E(B)} w_j$. That is, $w^-(v^*) = \sum_{v \in V(B)} w^-(v) - \sum_{v \neq v^*} w^-(v) = \sum_{v \in V(B)} w^+(v) - \sum_{v \neq v^*} w^+(v) = w^+(v^*)$, meaning that v^* is automatically balanced. Finally, the vertices of B' outside B are clearly balanced, because all edges entering and leaving them have flow value zero. Therefore W is indeed a feasible circulation. □

It is simple to determine the complexity of the algorithm. Finding strongly connected components can be done in $O(n+m)$ time [11], so as for the initial flow value assignments. The computation of *OUT PROCEDURE* and *IN PROCEDURE* require $O(n)$ time. Consequently, the overall time is $O(n+m)$.

Finally, observe that the flow values of the feasible circulation obtained by the algorithm are all integers of size $w_j < 2 \sum_{e_j \in E(D)} \lceil b_j \rceil$.

4. Constructing normal models. Let G be a PCA graph and $(C, \mathcal{A})$ a PCA model of it. We describe an algorithm for transforming $(C, \mathcal{A})$ into a normal model. The algorithm is based on [2, 4, 13]. For each $A_i \in \mathcal{A}$, the idea is to traverse C, searching for an arc A_j of $\mathcal{A}$ which together with A_i would cover C. If such an arc is found, then the arc A_j is conveniently shrunk with the purpose that A_i, A_j would no longer cover C. Clearly, it has to be assured that the new model so obtained is still a PCA model for G. Recall from Theorem 1 that any PCA graph admits a normal

model.

The proposed method considers the arcs A_i in the circular ordering $A_1, \ldots, A_n$. For each i, only a fraction of the arcs of $\mathcal{A}$ would be examined in general, with the above goal of searching for an arc A_j, which together with A_i, cover C. The algorithm consists of the computation of a procedure $NORMAL(i)$, for $i = 1, \ldots, n$. Each computation $NORMAL(i)$ examines the extreme points $s'_i, \ldots, t'_i \in A_i \cup \{s_i, t_i\}$, consecutive in the circular ordering, where s'_i is the *first point* and t'_i the *last point* of $NORMAL(i)$, respectively. Each iteration inside $NORMAL(i)$ considers one of the extreme points $s'_i, \ldots, t'_i$, in circular ordering. Let p be the current point under examination by $NORMAL(i)$. Clearly, p is either a start point or an end point. The following is the action taken, according to these alternatives. If p is a start point, then nothing is done and the examination of p is terminated. Otherwise, $p = t_j$ for some j, and check whether A_i and A_j cover C. In the affirmative case, shrink A_j in such a way that A_i and A_j would no longer cover C, and terminate the examination of p. The actual shrinking operation consists of moving t_j counterclockwise in C, so that it becomes an interior point of the segment $(PRED(s_i), s_i)$. When A_i and A_j do not cover C, then terminate the computation $NORMAL(i)$, disregarding all possible extreme points of $(C, \mathcal{A})$ contained in A_i which lie after $p = t_j$, when traversing A_i in the circular ordering. Consequently, such an extreme point is the last point t'_i of $NORMAL(i)$. The first point s'_i of $NORMAL(i)$ is determined at the moment $NORMAL(i)$ starts, as follows. When $i = 1$, define $s'_1 = s_1$. For $i > 1$, let $s'_i = \max\{s_i, t'_{i-1}\}_{s_{i-1}}$. Similarly as above, the algorithm disregards all extreme points contained in A_i, which lie before s'_i in A_i.

The procedure is formulated below. If p and q are points of C, then denote by $POINT(p, q)$ an arbitrarily chosen point of the (open) arc (p, q). The external calls are $NORMAL(i)$, for $i = 1, \ldots, n$.

procedure $NORMAL(i)$
  if $i = 1$ ***then*** $p := s_1$ ***else*** $p := \max\{p, s_i\}_{s_{i-1}}$
  repeat
    if p is an end point t_j ***then***
      if $A_i \cup A_j = C$ then $t_j := POINT(PRED(s_i), s_i)$
      else return
    $p := SUC(p)$

The following theorem assures that $NORMAL(i)$ terminates, and $(C, \mathcal{A})$ is a normal model after the computation of $NORMAL(n)$.

THEOREM 8. *The algorithm is correct.*

Proof. The algorithm consists of the computation of $NORMAL(i)$, for $i = 1, \ldots, n$. We prove that after completing $NORMAL(i)$, no two arcs A_k and A_j cover C, with $1 \leq k \leq i$. Furthermore, we ought to prove that the shrinking operations maintain $(C, \mathcal{A})$ as a PCA model for G.

Clearly, if there is an arc A_j such that A_i and A_j cover C, then $s_j, t_j \in A_i$, and t_j precedes s_j in A_i. Suppose the point $p = t_j$ is reached during $NORMAL(i)$. Then $A_i \cup A_j = C$ implies that t_j will be moved to $POINT(PRED(s_i), s_i)$. Let A^*_j be the new arc A_j, after shrinking. We know that A_i, A^*_j no longer cover C. We show that the configuration of $(C, \mathcal{A})$ still reflects the adjacencies of G. Clearly, A_i and A^*_j still intersect, otherwise A_i, A_j would not cover C. We discuss the intersections between A^*_j and any other arc $A_k \neq A_i$. We can restrict to arcs A_k which intersect A_i and have an extreme point, s_k or t_k, inside (s_i, t_j), since the intersections of A^*_j and the remaining arcs are not affected by the shrinking of A_j. The following are the possible

alternatives.

Case 1. $s_k \in (s_i, t_j)$. If $t_k \in (s_i, t_i)$, then t_k precedes s_k in A_i, otherwise $(C, \mathcal{A})$ is not a PCA model. Consequently, A_k and A_j^* also intersect. When $t_k \notin (s_i, t_i)$ it follows that A_j^* and A_k again intersect, because $s_j \in (s_i, t_i)$.

Case 2. $s_k \notin (s_i, t_j)$. By the initial assumption, $t_k \in (s_i, t_j)$. Examine the possibilities for s_k. The situation $s_k \in (s_i, t_j)$ is in Case 1. Let $s_k \notin (s_i, t_j)$. Then $s_k \in (t_j, s_j)$, otherwise A_j contains A_k. Consequently, A_j^* and A_k again intersect.

Consequently, shrinking A_j in the manner $NORMAL(i)$ does actually preserve adjacencies. In what follows, we show that the modified model is still a PCA model. It is sufficient to prove that A_j^* cannot be contained in any other arc of $\mathcal{A}$, except for A_j which has been removed. Suppose the contrary, that is, A_j^* is contained in some arc A_k. In this case, the circular ordering of the considered extreme points must be $t_j^*, s_i, t_k, t_j, s_k, s_j, t_i$; otherwise A_k also contains A_j, or A_k does not contain A_j^*. Consequently, $A_k \cup A_i = C$. Since t_k precedes t_j in A_i, it follows that A_k would have been shrunk before A_j in $NORMAL(i)$. The latter contradicts A_k to contain A_j^*.

It remains to show that any arc A_k, which together with A_i covers C, is detected by the algorithm. Suppose the contrary. That is, $NORMAL(i)$ terminates and does not shrink A_k. Then $t_k \notin (s_i', t_i')$. Because $t_k \in (s_i, t_i)$, there are two possibilities.

Case 1. $t_k \in (s_i, s_i')$. Then $s_i \neq s_i'$, which implies $i > 1$ and $t_{i-1}' \in A_i$. We also know that $s_i' \equiv t_{i-1}'$ is an end point t_j satisfying $A_{i-1} \cup A_j \neq C$. Observe that A_{i-1} and A_j may (or may not) coincide. Because $(C, \mathcal{A})$ is a PCA model, the sequence $s_j, s_{i-1}, s_i, t_k, t_j, t_{i-1}, t_i$ is in the circular ordering. Try to locate s_k. Because $A_i \cup A_k = C$, $s_k \in (t_k, t_i)$. However, $s_k \notin (t_k, t_{i-1})$ because A_k and A_{i-1} would also cover C, implying that $NORMAL(i-1)$ would also have moved t_k from the considered position, by an inductive argument. On the other hand, $s_k \notin (t_{i-1}, t_i)$ because in this case $A_{i-1} \cup A_k \neq C$, implying that t_k would have been reached by $NORMAL(i-1)$ before t_j. In the latter situation, $NORMAL(i-1)$ would have terminated at t_k, contradicting the assumption.

Case 2. $t_k \in (t_i', t_i)$. We know there is an arc A_j, such that $t_j = t_i'$ and $A_i \cup A_j \neq C$. Because $A_i \cup A_k = C$, it follows that $s_k \in (t_k, t_i)$. In this situation, A_k contains A_j, contradicting $(C, \mathcal{A})$ to be a PCA model. Hence, Case 2 also does not occur.

Therefore every arc, which together with A_i covers C, is shrunk by the algorithm, completing the proof of correctness. □

THEOREM 9. *The algorithm terminates within $O(n)$ time. Moreover, there are less than $6n$ iterations of the* repeat *block during the entire process.*

Proof. The number of steps performed by the algorithm corresponds to the number of iterations of the *repeat* block of the described formulation. Such a number is equivalent to the number of assignments of extreme points to p during the entire process. We will prove that the total number of such assignments is less than $6n$. The first value of p is s_1 and the last one, in the worst case, is t_n. We know that p never moves counterclockwise in C. First, assume for a while that p does not stay stationary in two consecutive iterations, and also disregard the shrinking of arcs. Any of the $2n$ extreme points can be assigned to p. If $t_n \notin A_1$, then the assignments would correspond at most to a complete turn around the circle, because the model is PCA. That is, $2n$ assignments. On the other hand, if $t_n \in A_1$, then any extreme lying between s_1 and t_n may be assigned again to p, but at most once more, besides

its assignment in $NORMAL(1)$. This corresponds to an additional n units in our account.

Next, examine the case when p stays stationary in two consecutive iterations. Such a situation may only occur, possibly at the first iteration within $NORMAL(i)$, when $s'_i = t'_{i-1}$, for $i > 1$. Overall, it counts to additional $n-1$ units, at most, in the total number of assignments to p.

Finally, examine the shrinking operations. Each time an arc A_j is shrunk, an extreme point t^*_j is placed in a different position of $(C, \mathcal{A})$, offering the possibility to t^*_j itself to be assigned to p. Hence, the shrinking operations may contribute to our account. In fact, if an arc gets shrunk k times during the algorithm, this may represent additional k assigments to p, in the worst case. Furthermore, there might be $O(n)$ distinct arcs, each of them covering C, together with A_j. However, the next assertion assures that even in this case, the number of times A_j gets shrunk is low.

Claim. Any arc of $(C, \mathcal{A})$ may be shrunk at most twice during the entire algorithm.

Proof. Let A_j be an arc of $(C, \mathcal{A})$. Clearly, if A_j together with some other arc does not cover C, then A_j is not shrunk at all. Otherwise, consider the model $(C, \mathcal{A})$ at the beginning of the computation $NORMAL(k)$, where A_j gets shrunk for the first time, if any. Denote by $A_{j_1}, \ldots, A_{j_l}$ the arcs of $(C, \mathcal{A})$, each of them covers C, together with A_j. This implies that all extreme points of the arcs $A_{j_1}, \ldots, A_{j_l}$ belong to (s_j, t_j). Furthermore, s_{j_k} precedes t_{j_k} in A_j, $1 \leq k \leq l$. Without loss of generality, the starting points $s_{j_1}, \ldots, s_{j_l}$ are in the circular ordering. Clearly, the arc of least index among $A_j, A_{j_1}, \ldots, A_{j_l}$ corresponds to one of this collection which is first handled by the algorithm. Let $i = \min\{j, j_1, \ldots, j_l\}$. The following are the alternatives for i:

Case 1. $i = j$. We know that $NORMAL(j)$ does not shrink A_j. Furthermore, Theorem 8 implies that $NORMAL(j)$ would shrink all arcs $A_{j_1}, \ldots, A_{j_l}$. Consequently, A_j never gets shrunk.

Case 2. $i = j_1$. Because A_j and A_{j_1} cover C, during $NORMAL(j_1)$ the assignment $p := t_j$ necessarily occurs, since Theorem 8 assures that $NORMAL(j_1)$ shrinks A_j. Then t_j moves to the position t^*_j, becoming the predecessor of s_1. The arc A^*_j so obtained does not intersect any of $A_{j_1}, \ldots, A_{j_l}$, and consequently, it cannot get shrunk. Consequently, A_j is shrunk only once during the process.

Case 3. $i = j_k$, for some $1 < k \leq l$. Then $NORMAL(j_k)$ moves t_j to t^*_j, where t^*_j, this time, is the new predecessor of s_{j_k}. Following the circular ordering of $(C, \mathcal{A})$, the first subsequent computation which can possibly shrink A_j is $NORMAL(j_1)$. By Case 2, the latter would shrink A_j, exactly once more. Consequently, A_j gets shrunk twice.

The conclusion is that any arc may get shrunk at most twice during the process. □

Consequently, the shrinking operations contribute at most $2n$ units to our account. Therefore the algorithm performs less than $6n$ iterations of the *repeat* block. Each of these iterations requires no more than constant time. That is, the overall complexity is $O(n)$. □

5. Recognition and model construction. In this section, we describe the algorithms for recognizing UCA graphs and for constructing UCA models. Also, we determine the size of the obtained model by showing that its extremes correspond to integers of size $O(n)$.

The algorithm is as follows. Let G be the given graph.

1. Verify if G is a PCA graph, using the algorithm [1]. In the affirmative case, let $(C, \mathcal{A})$ be the PCA model so obtained. In the negative case, G is not UCA.
2. Transform $(C, \mathcal{A})$ into a normal model $(C', \mathcal{A}')$, using the algorithm of section 4.
3. Construct the segment digraph D of $(C', \mathcal{A}')$. Find the connected components of D. Verify if all connected components are strongly connected. In the negative case, report that G is not UCA and stop. Otherwise, report that G is UCA and construct a UCA model $(C'', \mathcal{A}'')$ of it by running Step 4.
4. Construct the segment network of $(C', \mathcal{A}')$ by assigning to each edge $e_j \in E(D)$ a unit lower capacity and an unbounded upper capacity. Then find the flow values of a feasible circulation W of D, using the algorithm of section 3. Let $p'_1, \ldots, p'_{2n}$ be the extremes of $(C', \mathcal{A}')$, in circular ordering, with p_1 corresponding to the start point of the first arc of the model. Then construct $(C'', \mathcal{A}'')$ as follows. The length of C'' is $\sum_{e_j \in E(D)} w_j$. The extremes $p''_1, \ldots, p''_{2n}$ of $(C'', \mathcal{A}'')$ are defined by the following conditions: p''_1 is an arbitrary point of C'', and the segment (p''_j, p''_{j+1}) has length w_j, $1 \leq j < 2n$.

The correctness follows basically from Theorems 4, 7, and 8, and Corollary 6. We evaluate the complexity of the recognition algorithm. Step 1 requires $O(n+m)$ time [1]. Step 2 takes $O(n)$ time by Theorem 9. As for Step 3, note that for constructing the segment digraph, we need to construct the full and reduced systems of $(C', \mathcal{A}')$. The first of them is a family of $n-1$ equations, whose variables are segment lengths. In each side of the equations, the segment lengths are consecutive. Consequently, we need $4n-4$ indices to represent these equations, overall. On the other hand, in the reduced system, each segment length appears at most twice. That is, it requires at most $4n$ indices. The segment digraph has n vertices and at most $2n$ edges. There is no difficulty to construct D in $O(n)$ time. The connected and strongly connected components can be constructed in linear time [11]. Step 4 also requires linear time in the size of D, according to section 3. Consequently, the overall complexity of algorithm is $O(n+m)$. Furthermore, Steps 2, 3, and 4 require $O(n)$ time, meaning that the complexity of the algorithm reduces to $O(n)$, if the input is a PCA model of G, with ordered extremes of the arcs.

Finally, we verify the size of the UCA model obtained by the algorithm.

COROLLARY 10. *Let $(C, \mathcal{A})$ be a UCA model constructed by the algorithm. Then the extremes of $(C, \mathcal{A})$ correspond to integers of size $< 4n$.*

Proof. Let $p_1, \ldots, p_{2n}$ be the extremes of $(C, \mathcal{A})$ in circular ordering. We associate the segment length l_j to the extreme p_{j+1}, $1 \leq j < 2n$. From Step 4 of the algorithm, we know that the length l_j of each segment of $(C, \mathcal{A})$ equals the flow value w_j of the corresponding edge e_j of D. On the other hand, we know from section 3 that $w_j < 2m$. Since $m < 2n$, it follows that $l_j < 4n$. □

6. Conclusion. The proposed algorithms for recognizing UCA graphs and constructing UCA models terminate within $O(n+m)$ time. Furthermore, the complexity reduces to $O(n)$ when the input is a PCA model. It should be noted that the algorithms require the extreme points of the given model to be ordered. If the input is a graph given by its vertices and edges, then there is a preprocessing step, which consists of running the algorithm by Deng, Hell, and Huang [1], for constructing a PCA model of the graph. The algorithm [1] can be implemented so as to construct a PCA model with ordered extreme points within $O(n+m)$ time. On the other hand,

if the input is already a PCA model and its extreme points are not ordered, then we need an additional $O(n \log n)$ steps for ordering.

The UCA model constructed by the proposed algorithm is such that its extreme points correspond to integers of size $O(n)$. Regarding this feature, we leave the four related questions below.

Given a UCA graph G, find a UCA model whose extremes of the arcs correspond to integers, satisfying

(i) the maximum segment length is minimized.

(ii) the circle length is minimized.

More generally, for given n, find the least l, such that any UCA graph with n vertices admits a UCA model whose extremes of the arcs correspond to integers, such that

(iii) the maximum segment length is $\leq l$.

(iv) the circle length is $\leq l$.

Corollary 10 implies that $l < 4n$ for Question (iii), and that $l < 4n^2$ for Question (iv).

Acknowledgments. We are grateful to Professor Pavol Hell and an anonymous referee for their most valuable suggestions and comments, which lead to improvements in different parts of this paper.

REFERENCES

[1] X. Deng, P. Hell, and J. Huang, *Linear time representation algorithms for proper circular-arc graphs and proper interval graphs*, SIAM J. Comput., 25 (1996), pp. 390–403.

[2] G. Durán, A. Gravano, R. M. McConnell, J. P. Spinrad, and A. Tucker, *Polynomial time recognition of unit circular-arc graphs*, J. Algorithms, 58 (2006), pp. 67–78.

[3] E. Eschen and J. P. Spinrad, *An $O(n^2)$ algorithm for circular-arc graph recognition*, in Proceedings of the 4th Annual ACM-SIAM Symposium on Discrete Algorithms, Austin, TX, ACM, New York, SIAM, Philadelphia, 1993, pp. 128–137.

[4] M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.

[5] A. J. Hoffman, *Some recent applications of the theory of linear inequalities to extremal combinatorial analysis*, in Proceedings of the Symposia in Applied Mathematics, American Mathematical Society, Providence, RI, 1960, pp. 113–117.

[6] W. L. Hsu, *$O(mn)$ algorithms for the recognition and isomorphism problems on circular-arc graphs*, SIAM J. Comput., 24 (1995), pp. 411–439.

[7] M. C. Lin and J. L. Szwarcfiter, *Efficient construction of unit circular-arc models*, in Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms, Miami, FL, 2006, pp. 309–315.

[8] R. M. McConnell, *Linear-time recognition of circular-arc graphs*, Algorithmica, 37 (2003), pp. 93–147.

[9] A. Schrijver, *Combinatorial Optimization, Vol A: Polyhedra and Efficiency*, Springer-Verlag, Berlin, 2003.

[10] J. P. Spinrad, *Efficient Graph Representations*, Fields Institute Monographs 19, AMS, Providence, RI, 2003.

[11] R. E. Tarjan, *Depth-first search and linear graph algorithms*, SIAM J. Comput., 1 (1972), pp. 146–160.

[12] A. Tucker, *Characterizing circular-arc graphs*, Bull. Amer. Math. Soc., 76 (1970), pp. 1257–1260.

[13] A. Tucker, *Structure theorems for some circular-arc graphs*, Discrete Math., 7 (1974), pp. 167–195.

[14] A. Tucker, *An efficient test for circular-arc graphs*, SIAM J. Comput., 9 (1980), pp. 1–24.

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

Min Chih Lin[1,⋆] and 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.

J. Gudmundsson (Ed.): SWAT 2008, LNCS 5124, pp. 355–366, 2008.

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. □

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.

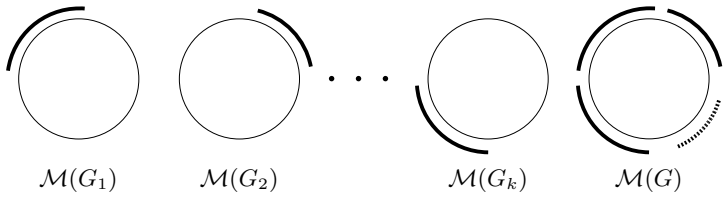


Fig. 1. The figures, except the last one, show the k segments whose corresponding models lie in $\mathcal{M}(G)$, whereas the last figure depicts $\mathcal{M}(G)$ itself

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 segment.*

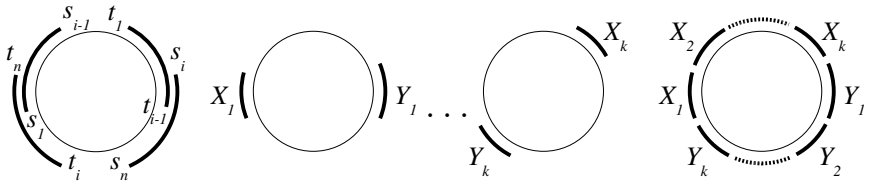
We now consider the complexity of the algorithm when $\mathcal{M}$ is given as in Step 1. Step 2 can be solved in $O(n)$ time as in Subsection 3.2, where we obtain a range representing each component. Step 3 is done by reversing $\mathcal{M}_i$ and then comparing $\mathcal{M}_i$ and $\mathcal{M}_i^{-1}$. 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, 2nd edn. 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. (ed.) 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: Fomin, F.V. (ed.) 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: Brandstädt, A., Kratsch, D., Müller, H. (eds.) WG 2007. LNCS, 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)