Efficient Construction of Unit Circular-Arc Models

Min Chih Lin* Jayme L. Szwarcfiter†

Abstract

In a recent paper, Durán, Gravano, McConnell, Spinrad and Tucker described an algorithm of complexity $O(n^2)$ for recognizing whether a graph G with n vertices 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 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 which leads to linear time algorithms for recognizing UCA graphs and constructing UCA models. Furthermore, we construct models whose extreme of the arcs correspond to integers of size $O(n)$. The proposed algorithms provide positive answers to the three above questions.

1 Introduction.

In a recent paper, Durán, Gravano, McConnell, Spinrad and Tucker [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 [7]. As for problem (i), we mention the characterization of UCA graphs in terms of forbidden subgraphs by Tucker [9]. The proof of this characterization actually contains an algorithm for constructing a UCA model. However, due to the possible manipulation of large integers, the complexity of the algorithm [9] is unknown, and so far the construction of a UCA model in polynomial time remains unsolved [2, 7].

In the present paper, we propose a characterization for UCA graphs, which leads to linear time algorithms for recognizing UCA graphs and constructing UCA models, whose extremes of the arcs correspond to integers of $O(n)$ size. Therefore we give affirmative answers to the above three questions.

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. Write $d_G(v)$ for the ***degree*** of v in G. We may also simple 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. Say that D is ***eulerian*** when $d^-(v) = d^+(v)$, for all vertices $v \in V(D)$. Also, D is ***connected*** when its underlying (undirected) graph is connected. Finally, D is ***strongly connected*** when, D contains paths from u to v and from v to u, for any $u, v \in V(D)$.

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 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

*Universidad de Buenos Aires, Facultad de Ciencias Exactas y Naturales, Departamento de Computación, Buenos Aires, Argentina. E-mail: oscarlin@dc.uba.ar. Partially supported by UBACyT Grants X184 and X212, PICT ANPCyT Grant 11-09112 and PID Conicet Grant 644/98, CNPq under PROSUL project Proc. 490333/2004-4.

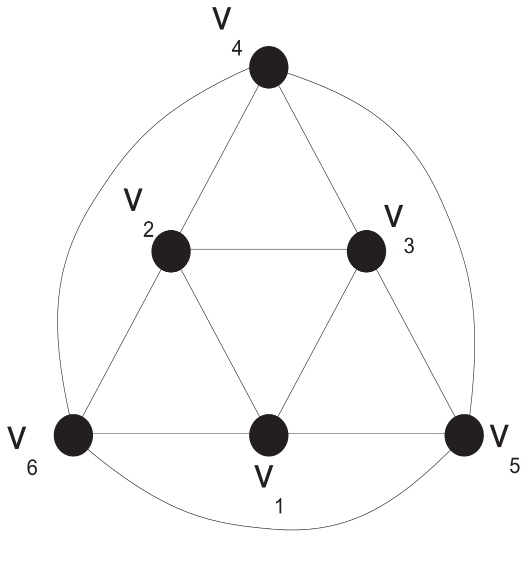
†Universidade Federal do Rio de Janeiro, Instituto de Matemática, NCE and COPPE, Caixa Postal 2324, 20001-970 Rio de Janeiro, RJ, Brasil. E-mail: jayme@nce.ufrj.br. Partially supported by the Conselho Nacional de Desenvolvimento Científico e Tecnológico, CNPq, and Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro, FAPERJ, Brasil.

SODA '06, January 22-26, Miami, FL
©2006 SIAM ISBN 0-89871-605-5/06/01

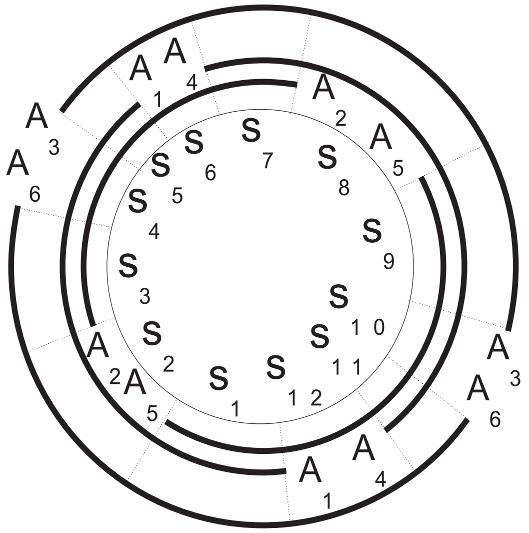
v, and v a ***descendant*** of u. A ***leaf-root ordering*** of T is a sequence $v_1, \ldots, v_n$ of its vertices, such that $i < j$ implies v_i is not an ancestor (descendant) of v_j in T.

A ***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 intersect 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 ***proper circular-arc*** (PCA) model, while if all arcs of $\mathcal{A}$ have the same length then $(C, \mathcal{A})$ is a ***unit circular-arc*** (UCA) model. A PCA (UCA) graph is one admitting a PCA (UCA) model. A ***normal*** model is a PCA model where no two 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, 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. 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 whose extremes are two extremes of $\mathcal{A}$, which are consecutive in the circular ordering 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 12 segments of the corresponding CA model. If p is an extreme of $\mathcal{A}$, denote by $PRED(p)$ and $SUC(p)$, the extreme of $\mathcal{A}$ which immediately precedes and succeeds p, in the circular ordering, respectively.

The first characterization of CA graphs leading to a polynomial time recognition algorithm is due to Tucker [8]. Subsequently, faster algorithms have been described by Hsu [5] and by Eschen and Spinrad [3]. Recently, a linear time recognition algorithm for CA graphs has been formulated by McConnell [6]. 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. The first polynomial time algorithm for recognizing UCA graphs is that by Durán, Gravano, McConnell,



(a)



(b)

Figure 1: A PCA graph and a normal model of it.

Spinrad and Tucker [2].

Given a graph G, the algorithm [2] initially employs [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 UCA graph, employing the normal model constructed in the first phase. The complexity of each of the algorithms corresponding to these two phases is $O(n^2)$. 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 a generalization of eulerian digraphs. Section 3 presents the algorithm for constructing normal models. Section 4 contains the actual algorithm for recognizing UCA graphs and constructing UCA models.

2 A characterization for UCA graphs.

In this section, we define weakly eulerian digraphs and characterize UCA graphs in terms of this class. The following two theorems are basic to our purposes.

Theorem 2.1. (Golumbic[4], Tucker[9]) *: Let G be a PCA graph. Then G admits a normal model.*

Theorem 2.2. (Tucker[9]) *: 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 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 $(C, \mathcal{A})$:

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

$$l_j > 0,\ j = 1, \ldots, 2n$$

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 2.1. *: 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.*

Next, we describe a graph theoretical model for R. 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 additional, 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.

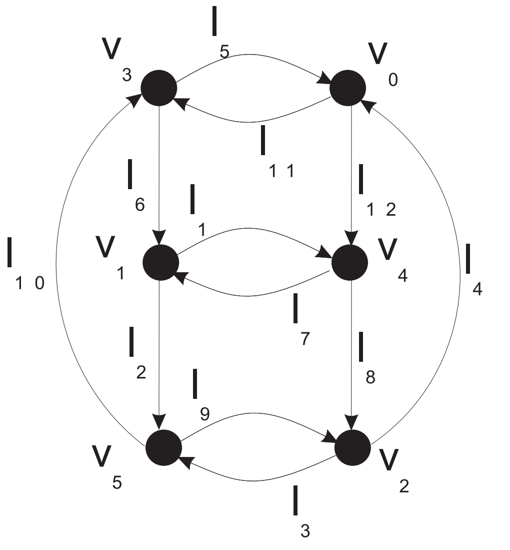
The description of D is complete. Clearly, D has n vertices and $2n$ edges and no loops, except possibly at v_0.

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).

Consider the following generalization of eulerian digraphs. Let D be a digraph and W a weighting of its edges, where each edge $e_j \in E(D)$ is assigned a positive real weight w_j. For each $v_i \in V(D)$, denote $w^-(v_i) = \sum_{e_j \in E^-(v_i)} w_j$ and $w^+(v_i) = \sum_{e_j \in E^+(v_i)} w_j$. Say that W is a ***weakly eulerian weighting*** if $w^-(v_i) = w^+(v_i)$, for all $v_i \in V(D)$. Finally, a ***weakly eulerian digraph*** is one admitting a weakly eulerian weighting. Clearly, eulerian digraphs are weakly eulerian.

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

(a)



(b)

Figure 2: A reduced system and its segment digraph.

The following theorem characterizes UCA graphs in terms of weakly eulerian digraphs.

Theorem 2.3. *: Let G be a PCA graph, $(C, \mathcal{A})$ a normal model of it and D its segment digraph. Then G is a UCA graph if and only if D is weakly eulerian.*

Sketch of the proof: Let G be a UCA graph, and $(C, \mathcal{A})$ a normal model of G. By Theorem 2.2, 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 digraph. Assign a weight to each edge e_j of D, equals to the length l'_j of the segment of $(C', \mathcal{A}')$, corresponding to S_j. It can be shown that such an assigment W satisfies the conditions for D to be weakly eulerian.

Conversely, by hypothesis D is weakly eulerian.We prove that G is UCA, by contructing 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 be a weakly eulerian weighting of D, and w_j the weight 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 weight w_j of e_j. It can be shown that the model so constructed is a UCA model.

Finally, when there is a segment S_j contained in all arcs of $\mathcal{A}$ or in none of them, assign any positive length to S_j. $\triangle$

The above theorem implies that the problem of recognizing if a given graph G is UCA is equivalent to deciding if the corresponding segment digraph is weakly eulerian. The following is a characterization of general weakly eulerian digraphs.

Theorem 2.4. *: A digraph D is weakly eulerian if and only if each of its connected components is strongly connected.*

Sketch of the proof: By hypothesis, D is weakly eulerian. Let $w_j > 0$ be the weight of each edge e_j of it, in a weakly eulerian weighting W of D. Assume the contrary, that is, D contains a connected component D' which is not strongly connected. Let B be a special strongly connected component of D', such that no edge starts at B and ends at another strongly connected component of D'. Clearly, B must exist. It can be proved that all edges ending at B and starting at some other component have weight zero, a contradiction. Therefore D' is strongly connected.

Converserly, by hypothesis, each connected component of D is strongly connected. We prove that D is weakly eulerian. Let B be any connected component of G. We assign a weighting W to the edges of B, and show W to be weakly eulerian. Start by assigning a perhaps temporary weight $w_j = 1$ to each edge e_j of B. Then classify the vertices of D into three types. Say that a vertex v is ***in-deficient*** when $w^-(v) < w^+(v)$, ***out-deficient*** when $w^+(v) < w^-(v)$ and ***balanced*** for the situation $w^-(v) = w^+(v)$. The idea is first to increase the weights of the edges ending at the in-deficient vertices, until all of them become balanced. With such an aim, choose an arbitrary vertex v^* of B and a spanning out-arborescence T of B, with root v^*. By traversing T in leaf-root ordering, we transform all in-deficient vertices into balanced ones, except possibly v^*, while maintaning the balanced vertices as such. Afterwards, increase the weights of the out-deficient vertices, similarly by traversing a spanning in-arborescence of B, also with root v^*. Once the two trees have been traversed, all vertices of G are indeed balanced, except possibly v^*. However, because B is a connected component of G, the fact that all $v \neq v^*$ are balanced imples that v^* must be balanced too. Therefore D is weakly eulerian. $\triangle$

COROLLARY 2.1. *: Let D be a weakly eulerian digraph. Then D admits a weakly eulerian weighting in which all weights are positive integers.*

3 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, 9]. 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 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 2.1 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 $s'_i, \ldots, t'_i$. Let p 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 alternative. If p is a start point 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. 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, q are points of C denote by $POINT(p, q)$ an arbitrarily chosen point of the 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 algorithm is correct because we can prove that after completing $NORMAL(i)$, for $i = 1, \ldots, n$, no two arcs A_k and A_j cover C, with $1 \leq k \leq i$. Furthermore, $(C, \mathcal{A})$ is maintained as a PCA model for G, through the process.

THEOREM 3.1. *: The algorithm terminates within $O(n)$ time. Moreover, there are less than $5n$ iterations of the* ***repeat*** *block, during the entire process.*

Sketch of the 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 can prove that the total number of such assignments is less than $5n$. The first value of p is s_1 and the last one is t_n, in the worst case. On the other hand, p never moves counterclockwise in C. Consequently, there are at most $2n$ assigments to p, when moving forwards. On the other hand, p may stay stationary in two consecutive iterations. However, 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 $n - 1$ steps, 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 an additional $k + 1$ 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.

Consequently, the shrinking operations contribute at most $2n$ units to our account. Therefore the algorithm perform less than $5n$ iterations of the ***repeat*** block. Each of these iterations requires no more than constant time. That is, the overall complexity is $O(n)$. $\triangle$

4 Recognizing and constructing the models.

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 3.
3. Construct the segment digraph D of $(C', \mathcal{A}')$. Find the connected components of D. Verify if each of the connected components is strongly connected. In the affirmative case G is UCA, otherwise it is not.

The correctness follows from Theorems 2.3 and 2.4. The complexity of Step 1 is $O(n + m)$ [1]. There is no difficulty to represent the full system, reduced system and segment digraph in $O(n)$ space and construct them in $O(n)$ time. The complexities of Steps 2 and 3 are both $O(n)$. Therefore the complexity of the algorithm is $O(n+m)$, if G is given by its vertices and edges. The complexity reduces to $O(n)$, if G is given by a PCA model with ordered extremes of the arcs. It should be noted that Steps 2 and 3 require the extremes to be ordered. However, the algorithm by Deng, Hell and Huang can be implemented so as to produce such an ordered PCA model, in $O(n + m)$ time. Finally, if G is given by a PCA model with unordered extremes then an additional $O(n\,log\,n)$ time for sorting is required.

Next, we consider the construction of UCA models. The method is to find a solution for the reduced system of a normal PCA model, and apply it to modify the segment lengths of the model. The UCA model is therefore obtained from the normal model, by possibly moving the extremes of the arcs, while preserving the circular ordering.

Let G be a UCA graph and $(C, \mathcal{A})$ a normal PCA model of it. The following algorithm will construct a UCA model $(C', \mathcal{A}')$ of G. Let $p_1, \ldots, p_{2n}$ be the extremes of $(C, \mathcal{A})$ in circular ordering, with p_1 corresponding to the start point s_1 of A_1.

1. Construct the segment digraph D of $(C, \mathcal{A})$ and the connected components of D. For each $v \in V(D)$, assign to the variables $w^-(v)$ and $w^+(v)$, initial values equal to the indegree and outdegree of v, respectively. For each edge $e_j \in E(D)$, initialize its weight $w_j := 1$.
2. For each connected component B of D, perform the *OUT PROCEDURE* and afterwards the *IN PROCEDURE*, below. 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-1}, p'_j) has length w_j, $1 \leq j \leq 2n$. Finally, for each

$1 \leq i \leq n$, let $A_i = (p_j, p_k)$ and define the arc $A'_i = (p'_j, p'_k) \in \mathcal{A}'$.

OUT PROCEDURE:

Choose an arbitrary vertex $v^* \in V(B)$ and construct 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.

The correctness of the algorithm follows again directly from Theorems 2.3 and 2.4.

As for the complexity, recall that D has at most $2n$ edges. Therefore, the constuction of D and the remaining operations of Step 1 terminate within $O(n)$ time. The same bound applies to the constructions of an out-arborescence and in-arborescence, as well as for the remaining operations. Consequently the complexity of the algorithm is $O(n)$.

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

THEOREM 4.1. *: Let $(C, \mathcal{A})$ be a UCA model constructed by the algorithm. Then the extremes of $(C, \mathcal{A})$ correspond to integers of size $\leq 4n$.*

Sketch of the proof: It can be shown that each weight w_j assigned by the algorithm has size $\leq 4n$. Let $p_1, \ldots, p_{2n}$ be the extreme points of $(C, \mathcal{A})$. Associate to each p_j the length $l_j = w_j$ of the segment $(p_j, SUC(p_j))$. $\triangle$

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] G. Durán, A. Gravano, R. M. McConnell, J. P. Spinrad and A. Tucker, Polynomial time recognition of unit circular-arc graphs, *J. of Algorithms*, (2004), in press.

[3] E. Eschen and J. Spinrad, An $O(n^2)$ Algorithm for Circular-Arc Graph Recognition, *Proceedings of the 4th Annual ACM-SIAM Symposium on Discrete Algorithms* (1993), pp. 128-137.

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

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

[6] R. M. McConnell, Linear-time recognition of circular-arc graphs, *Algorithmica* **37 (2)** (2003), pp. 93-147.

[7] J. P. Spinrad, *Efficient Graph Representations*, Fields Institute Monographs, American Mathematical Society, Providence, Rhode Island, 2003.

[8] A. Tucker, Characterizing circular-arc graphs, *Bull. American Mathematical Society* **76** (1970), pp. 1257-1260.

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

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