Certifying Algorithms for Recognizing Proper Circular-Arc Graphs and Unit Circular-Arc Graphs

Haim Kaplan and Yahav Nussbaum

School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel.
{haimk,nuss}@post.tau.ac.il

Abstract. We give two new algorithms for recognizing proper circular-arc graphs and unit circular-arc graphs. The algorithms either provide a model for the input graph, or a certificate that proves that such a model does not exist and can be authenticated in $O(n)$ time.

1 Introduction

A *certifying algorithm* for a decision problem is an algorithm that provides a *certificate* together with its answer. A certificate is an evidence that can be used to authenticate the correctness of the answer (cf. [7, 11]). An *authentication algorithm* is an algorithm that validates the certificate. Certifying algorithms reduce the risk of erroneous answer, caused by bugs in the implementation.

For example, a recognition algorithm of bipartite graphs can provide as a certificates a 2-coloring when the graph is bipartite and an odd cycle when the graph is not bipartite. Graph classes that have certifying recognition algorithm include chordal graphs [15], planar graphs [11], interval graphs and permutation graphs [7], proper interval graphs [4, 12], proper interval bigraphs [4], and more.

A *circular-arc graph* is an intersection graph of arcs on the circle. Every vertex is represented by an arc, such that two vertices are adjacent if and only if the corresponding arcs intersect. The arcs constitute a *circular-arc model* of the graph. Circular-arc graphs can be recognized in linear time [9, 6].

A circular-arc model in which no arc contains another is a *proper circular-arc (PCA) model*. A circular-arc graph that admits a PCA model is a *proper circular-arc (PCA) graph*. Tucker gave characterizations of PCA graphs, in terms of the adjacency matrix [16] and forbidden subgraphs [17]. Skrien [13] and Deng, Hell and Huang [1] gave characterizations that use orientation of the edges. The characterization of [1] leads to a linear-time recognition algorithm for PCA graphs. Spinrad [14] showed that the characterization of [16] also leads to a linear-time recognition algorithm for PCA graphs. Both algorithms construct a PCA model if the graph is a circular-arc graph, but fail to provide a certificate otherwise.

A circular-arc model in which all arcs are closed (or all arcs are open) and of the same length is a *unit circular-arc (UCA) model*. A circular-arc graph that admits a UCA model is a *unit circular-arc (UCA) graph*. Every UCA graph is a PCA graph. Tucker [17] gave a characterization of PCA graphs which are not UCA graphs. Recently, Durán, Gravano, McConnell, Spinrad, and Tucker

[2] presented a quadratic recognition algorithm for UCA graphs, based on this characterization. This algorithm does not provide a certificate for its answer. Even more recently, Lin and Szwarcfiter [8] gave new characterization for UCA graphs based on the length of the arcs in a PCA model. This gave a linear-time algorithm that constructs a UCA model if the input is a UCA graph, but fails to provide a certificate otherwise.

Circular-arc graphs generalize *interval graphs* which are the intersection graphs of intervals on the line. Kratsch, McConnell, Mehlhorn, and Spinrad [7] gave a linear-time certifying algorithm for interval graphs.

Another related graph class is *interval bigraphs*. A bipartite graph with the bipartition (X, Y) is an interval bigraph, if it can be represented by intervals on the line, such that the interval of $x \in X$ intersects the interval of $y \in Y$ if and only if x and y are adjacent in the graph. Two intervals corresponding to two vertices in X or to two vertices in Y, may or may not intersect. An interval bigraph that have a model in which no arc contains another, is a *proper interval bigraph*. Hell and Huang [5] showed that the class of proper interval bigraphs, is exactly the class of the complements of co-bipartite PCA graphs. These graph classes are known to be equivalent to many other well known graph classes including bipartite permutation graphs, bipartite AT-free graphs and bipartite trapezoid graphs. Hell and Huang [4] also gave a simple linear-time certifying algorithm for recognizing proper interval bigraphs, which we use in our algorithm.

In this paper, we present characterizations for PCA graphs and UCA graphs which are based on [16, 17]. Those characterization leads to linear-time certifying algorithms for recognizing these classes of graphs. If the input graph is a member of the graph class, the algorithm provide the appropriate model for it. Otherwise, if the input graph is not a member of the graph class, we provide a certificate for this answer that can be authenticated in $O(n)$ time, where n is the number of vertices in the graph. This time bound is better than the time bound of the recognition algorithm, so the certificate is a *strong certificate* [7].

2 Preliminaries

Let $G(V, E)$ be a finite simple graph, and let $n = |V(G)|$ and $m = |E(G)|$. The *(closed) neighborhood* of v is $N[v] = \{v\} \cup \{u \mid vu \in E\}$. For $u, v \in V(G)$, if $uv \notin E(G)$ then we say that uv is a *non-edge*.

The sequence $P = (v_1, v_2, \ldots v_k)$ with $v_i v_{i+1} \in E(G)$ is a *path*. If $v_k v_1 \in E(G)$ then P is also a *cycle*. The sequence $P = (v_1, v_2, \ldots v_k)$ with $v_i v_{i+1} \notin E(G)$ is a *co-path*. If $v_k v_1 \notin E(G)$ then P is also a *co-cycle*. The length of a path, or a co-path, P is denoted by $|P|$. A path, cycle, co-path or co-cycle in which all the vertices are distinct is *simple*.

A graph that can be partitioned into two independent sets is a *bipartite graph*. The complement of a bipartite graph is a *co-bipartite graph*.

To simplify we refer to the clockwise direction of the circle as *right* and to the counterclockwise direction of the circle as *left*, as we view them if we stand at the center of the circle.

Table 1. Intersection types of two arcs in circular-arc model by order of their endpoints.

Endpoints order (left to right)	Intersection type
$[\ell(x), r(y), \ell(y), r(x)]$	$\hat{x}$ and $\hat{y}$ cover the circle
$[\ell(x), \ell(y), r(x), r(y)]$	$\hat{y}$ overlaps the right side of $\hat{x}$
$[\ell(x), r(y), r(x), \ell(y)]$	$\hat{x}$ overlaps the right side of $\hat{y}$
$[\ell(x), r(x), \ell(y), r(y)]$	$\hat{x}$ and $\hat{y}$ are nonadjacent
$[\ell(x), \ell(y), r(y), r(x)]$	$\hat{x}$ contains $\hat{y}$
$[\ell(x), r(x), r(y), \ell(y)]$	$\hat{y}$ contains $\hat{x}$

For a PCA graph G with a PCA model ϱ, every vertex $v \in V(G)$ has an arc in ϱ with two endpoints. We denote the arc of v by $\hat{v}$, the left endpoint by $\ell(v)$ and the right endpoint by $r(v)$.

Every pair of arcs $\hat{x}$ and $\hat{y}$ in ϱ either *cover the circle*, *overlap*, or *nonadjacent*. Containment of arcs in a PCA model is impossible. If $\hat{x}$ overlaps $\hat{y}$ and covers $r(y)$ then $\hat{x}$ *overlaps the right side of* $\hat{y}$.

The *adjacency matrix* of a graph G, denoted by $M(G)$, has 1 in position (i, j) if $v_i v_j \in E$, and 0 otherwise. The *augmented adjacency matrix* of G is the adjacency matrix of G with 1's on the main diagonal, that is $M^*(G) = M(G)+I$, where I is the identity matrix. We refer to the row and column in $M^*(G)$ that correspond to the vertex v as row v and column v.

A $(0, 1)$-matrix has the *consecutive-ones property* if its columns can be ordered so that in every row the 1's are consecutive. McConnell [10] gave a linear-time certifying algorithm for this property. A $(0, 1)$-matrix has the *circular-ones property* if its columns can be ordered so that in every row the 1's are circularly consecutive.

2.1 Representation

The desired graph representation for certifying algorithms on graphs is discussed in [7]. We use, as [7], ordered adjacency list representation of graphs. This representation allow us to get the list of neighbors of a given vertex in constant time, and to certify adjacency of two vertices in constant time. An edge is certified by its location in the ordered adjacency list. A non-edge is certified by the location in the adjacency list of the edge that would be its predecessor, if the non-edge was an edge. To collect the locations of $O(n)$ edges and non-edges, we radix sort them, and scan the sorted list together with the adjacency lists of the graph. The running time for this sort and scan is $O(m + n)$.

We represent a PCA model by a cyclic order of the endpoints. The $2n$ endpoints in the model are indexed according to their ranks in the order, starting at any arbitrary endpoint and going right. Each arc has the indices of its two endpoints. The type of intersection between two arcs can be determined, in constant time, by the order of their endpoints [3] (see Table 1). A unit circular-model obey length constraint, so the exact location of the endpoints on the circle is required.

We represent $(0, 1)$-matrices in a sparse way, similar to the representation

of graphs by ordered adjacency lists. This representation allows algorithms that process matrices to run in time proportional to the number of 1's in the matrix. For $M^*(G)$ the number of 1's is $O(m+n)$.

3 Characterization of Proper Circular-Arc Graphs

We define an *incompatibility graph* for PCA graphs, in a way similar to the definitions of incompatibility graphs for the consecutive-ones property [10] and for permutation graphs [7], as follows.

Let ϱ be a PCA model of G, and v_0 be a vertex in G. We start at $r(v_0)$ and traverse the circle to the right, we get a *traversal ordering* $(v_0, v_1, \ldots, v_{n-1})$ of the vertices according to the order in which we meet their right endpoints. This ordering defines a *traversal order relation* $R = \{(v_i, v_j) \mid i < j\}$.

The following holds for any PCA model of G. For every $x, y \in V(G)$, (x, y) and (y, x) cannot both appear in the same traversal order relation. We say that (x, y) is *incompatible* with (y, x). For every $w \in V(G)$, the right endpoints of all the vertices in $N[w]$ must be consecutive around the circle. Assume that $v_0 \notin N[w]$. Then, in a traversal ordering that starts with v_0 the vertices of $N[w]$ must be consecutive. Therefore, if $x, z \in N[w]$ and $y \notin N[w]$, the vertex y cannot be between x and z. So (x, y) and (y, z) are incompatible, with w as a *witness*. Assume that $v_0 \in N[w]$, now the vertices of $N[w]$ are not necessarily consecutive in a traversal ordering that starts with v_0, because it might be that v_{n-1} is also in $N[w]$. But $V(G) - N[w]$ must be consecutive in this ordering, so if $x, z \notin N[w]$ and $y \in N[w]$ then (x, y) and (y, z) are incompatible, with w as a witness.

The incompatible pairs define the *incompatibility graph* $IC(G; v_0)$ of G with starting vertex v_0. The vertex set of $IC(G; v_0)$ is $\{(x, y) \mid x, y \in V(G), x \neq y\}$. The edge set of $IC(G; v_0)$ are edges of the forms $(x, y)(y, x)$, $(x, y)(y, z)$ such that $x, z \in N[w]$, $y \notin N[w]$ for some $w \notin N[v_0]$ and $(x, y)(y, z)$ such that $y \in N[w]$, $x, z \notin N[w]$ for some $w \in N[v_0]$.

The definition of $IC(G; v_0)$ is analogous to the definition of the incompatibility graph for the consecutive-ones property $IC(M)$, presented by McConnell [10]. Since a consecutive-ones ordering is linear, we do not need a starting point to define $IC(M)$. The edge set of $IC(M)$ are $(x, y)(y, x)$, for every pair of columns x and y, and $(x, y)(y, z)$ such that there is a row w, with ones in the column of x, z but not in the column of y.

Theorem 1. *Let G be a PCA graph. For any $v_0 \in V(G)$, the incompatibility graph $IC(G; v_0)$ is bipartite.*

Proof. Let ϱ be a PCA model of G. Let $v_0 \in V(G)$ and let R be the traversal order relation defined by the traversal ordering of ϱ that starts with v_0. The relation R is made of vertices of $IC(G; v_0)$. The relation R cannot have any incompatible pairs, so the vertices of R are an independent set in $IC(G; v_0)$. Let ϱ' be a PCA model of G that is obtained by replacing the right and left directions of ϱ. Let R' be the traversal order relation defined by the traversal ordering of ϱ' starting with v_0. Any vertex in $IC(G; v_0)$ that is not in R is in R'. The vertices of R' are also an independent set, and therefore $IC(G; v_0)$ is bipartite. □

To certify that G is not a PCA graph we can provide an odd cycle in one of its incompatibility graphs. We do so without explicitly constructing the entire incompatibility graph, since the size of this graph might be as large as $\Theta(n^4)$. Note that $IC(G; v_0)$ might be bipartite even when G is not a PCA graph.

4 Certifying Algorithm for Proper Circular-Arc Graphs

Our certifying algorithm for PCA graphs consists of two cases, depending on whether G is co-bipartite or not. We begin the algorithm by deciding whether G is co-bipartite. If G is co-bipartite, then it is covered by two cliques. At least one of those cliques should cover half of $V(G)$, so $m \geq \frac{n}{2}(\frac{n}{2} - 1)$. If this inequality does not hold then G is not co-bipartite. Otherwise, $m = \Theta(n^2)$, and we check if $\overline{G}$ is bipartite in $O(n^2) = O(m)$ time.

4.1 The Complement of G is Not Bipartite

In the case where G is not co-bipartite, Tucker [16] showed that G is a PCA graph if and only if $M^*(G)$ has the circular-ones property.

To check if $M^*(G)$ has the circular-ones property, we use the following reduction of [16] from testing this property to testing the consecutive-ones property. Let M_1 be a $(0,1)$-matrix. Fix a column j. Form the matrix M_2 by complementing those rows with 1 in column j of M_1. Then M_1 has the circular-ones property if and only if M_2 has the consecutive-ones property.

Let v_0 be a vertex of minimum degree in G. To perform the reduction stated above in linear time, we complement the rows of $M^*(G)$ which have one in the column of v_0. Since the degree of v_0 is $O(m/n)$, we complement $O(m/n)$ rows. It takes $O(n)$ time to complement a row so we perform the entire reduction in $O(m)$ time. We denote by M the new matrix that we obtain.

After the reduction we run the certifying algorithm of McConnell [10] to test if M has the consecutive-ones property. If M has a consecutive-ones ordering, we order the columns of $M^*(G)$ in the same way, to get a circular-ones ordering for $M^*(G)$. Tucker [17] showed an algorithm to produce a PCA model of G from a this ordering that can be implemented in $O(n + m)$ time.

If M does not have the consecutive-ones property, then the algorithm of [10] produces a certificate for this fact. This certificate is an odd cycle C of length at most $n+2$ in the incompatibility graph $IC(M)$. Next, we show that all edges of C exist in $IC(G; v_0)$ so C is also an odd cycle in $IC(G; v_0)$.

Edges of C in $IC(M)$ have one of two forms. Edges of the form $(x, y)(y, x)$ always exist in $IC(G; v_0)$. Consider an edge $(x, y)(y, z)$ with a witness w, where w is a row in M such that the columns of x and z have 1 in this row, but the column of y has 0 in it. If $w \notin N[v_0]$ then the row of w in M is the same as in $M^*(G)$. So, $x, z \in N[w]$ while $y \notin N[w]$ and therefore $(x, y)(y, z)$ is an edge of $IC(G; v_0)$, with vertex $w \notin N[v_0]$ as a witness. Otherwise, if $w \in N[v_0]$, then the row of w in M is the complement of the row of w in $M^*(G)$. So, $y \in N[w]$ while $x, z \notin N[w]$ and therefore $(x, y)(y, z)$ is an edge of $IC(G; v_0)$, with vertex $w \in N[v_0]$ as a witness.

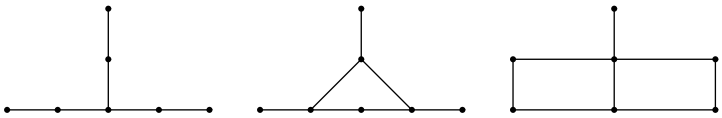

Fig. 1. Forbidden subgraphs.

We provide the odd cycle C in $IC(G; v_0)$, together with v_0 as a certificate. To complete the certificate, we need to add a certificate for all edges and non-edges of G that are involved in it. For an edge $(x, y)(y, z)$ with a witness w in $IC(G; v_0)$, we need to provide a certificate for the edges or non-edges xw, yw, zw and wv_0 in G. The length of the cycle in $IC(G; v_0)$ is $O(n)$, and thus there are $O(n)$ edges or non-edges to certify.

4.2 The Complement of G is Bipartite

In this case, when G is co-bipartite, we use the following forbidden subgraphs characterization of Tucker [17].

Theorem 2. *[17] Let G be a graph. If $\overline{G}$ contains an induced even cycle of length ≥ 6 or one of the graphs in Fig. 1, then G is not a PCA graph.*

A co-bipartite graph G is a PCA graph if and only if $\overline{G}$ is a proper interval bigraph [5]. So, we use the certifying algorithm for recognizing proper interval bigraph of Hell and Huang [4] on $\overline{G}$. Note that the graphs in Theorem 2 are exactly the graphs that the certifying algorithm for recognizing proper interval bigraphs [4] uses as certificates.

The graph G is covered by two cliques, one of those two cliques must cover at least $n/2$ of the vertices of G, therefore $m = \Theta(n^2)$. So we can produce $\overline{G}$ from G in $O(n^2) = O(m)$ time.

If $\overline{G}$ is an interval bigraph, we get a model for it, and we use an algorithm of Hell and Huang [5] to construct a PCA model of G, from this model. Otherwise, we have one of graphs in Theorem 2 as a certificate. For a graph of Fig. 1, we use its complement to certify that G is not a PCA graph.

If we have an induced even cycle of length ≥ 6 as a certificate that $\overline{G}$ is not a proper interval bigraph, we transform it into an odd cycle in an incompatibility graph of G. We do so for two reasons. First, a straightforward authentication of an even length cycle takes $O(m + n)$ time, while authentication of an odd cycle in an incompatibility graph takes $O(n)$ time. Second, we reduce the number of cases that the authentication algorithm has to deal with, since it has to verify an odd cycle in an incompatibility graph in the case where G is not co-bipartite.

Let $C = (x_0, x_1, \ldots, x_{2r-1})$ be an even induced cycle in $\overline{G}$. For every $i = 0, \ldots, 2r - 1$, and for every $j \neq i \pm 1$, we have that $(x_i, x_j) \in E(G)$, where the subscripts of the vertices are modulo $2r$. We find an odd cycle in the incompatibility graph $IC(G; x_1)$.

If r is even, we start the cycle in $IC(G; x_1)$ with (x_0, x_r). From the vertex (x_i, x_j) in the cycle, we continue to (x_j, x_{i+2}). We can use x_{i+1} as a witness for the edge $(x_i, x_j)(x_j, x_{i+2})$, since if we start with (x_0, x_r), we always have $x_{i+1} \in N[x_1]$ and $x_i, x_{i+2} \notin N[x_{i+1}]$ while $x_j \in N[x_{i+1}]$. After r edges we get to (x_r, x_0), we add an edge $(x_r, x_0)(x_0, x_r)$ to complete an odd cycle of length $r+1$.

If r is odd, we start the cycle in the incompatibility graph in (x_0, x_{r+1}) and continue in the same way. After r edges we get back to (x_0, x_{r+1}), and we have odd cycle of length r.

Building the cycle in the incompatibility graph $IC(G; x_1)$, together with certificate for all the edges in it takes $O(n^2) = O(m)$ time.

4.3 The Certificate and the Authentication Algorithm

The certificate of the recognition algorithm is either a PCA model of G, an odd cycle in an incompatibility graph $IC(G; v_0)$, or one of the graphs of Fig. 1 as an induced subgraph of $\overline{G}$. If we got a PCA model for G, then G is a PCA graph. For the other certificates, G is not a PCA graph by Theorem 1 or Theorem 2.

To authenticate a PCA model we first authenticate that the model is a circular-arc model of G, this step is described by McConnell [9]. The model is a PCA model only if no arc is contained in another, we check this for every pair of adjacent vertices by checking the order of their endpoints (see Table 1). The size of this certificate is $O(n)$ and the time to authenticate it is $O(n + m)$.

To authenticate an odd cycle in an incompatibility graph $IC(G; v_0)$, we first verify that it has an odd length not larger then $n + 2$. Then, we verify that the certificate is indeed a cycle. We also verify that each edge of the cycle belongs to $IC(G; v_0)$, by checking that every edge is either of the form $(x, y)(y, x)$ or has a valid witness. The size of the cycle is $O(n)$ and validating it takes $O(n)$ time.

If the certificate is one of the graphs of Fig. 1, we verify that every edge exists in the certificate if and only if it exists in the graph. The size of each of these graphs is $O(1)$, and hence the authentication time is also $O(1)$.

When the algorithm found that G is not a PCA graph, both possible certificates can be authenticated in $O(n)$ time and therefore are strong certificates.

5 Characterization of Unit Circular-Arc Graphs

In this section we present the structure theorem of Tucker [17] for UCA graphs together with some relaxations of it that we use. Let G be a PCA graph with a PCA model ϱ. Every PCA graph has a PCA model in which no pair of arcs covers the circle [17, 3]. In fact, the algorithm of Sect. 4 never constructs a model with a pair of arcs that covers the circle. Thus, in this section we assume that ϱ does not contain a pair of arcs that covers the circle.

Let $C = (x_0, \ldots, x_{p-1})$ be a simple cycle in G, such that for $i = 1, \ldots, p-1$, the arc $\hat{x}_i$ overlaps the right side of $\hat{x}_{i-1}$, and the arc $\hat{x}_0$ overlaps the right side of $\hat{x}_{p-1}$, in ϱ. We call such a cycle C, a *bounding cycle*. Assume that we traverse the cyclic list of endpoints of ϱ, starting immediately after $\ell(x_0)$, going right to $r(x_0)$

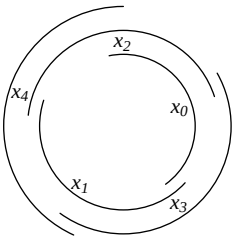

Fig. 2. Bounding cycle with ratio 5/2

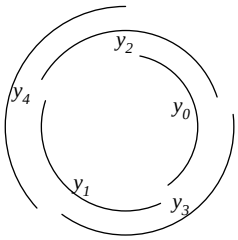

Fig. 3. Bounding co-cycle with ratio 5/2

and continuing from $r(x_i)$ to $r(x_{i+1})$. We call the list of endpoints obtained this way the *walk* of C. The number of times that C goes around the circle is the number of times that the walk of C hits $\ell(x_0)$, we denote this number by $t(C)$. The *ratio* of C, denoted by $p(C)$, is $|C|/t(C)$. See Fig. 2. We call C a *minimum bounding cycle* if there is no other bounding cycle C' with $p(C') < p(C)$. We denote a minimum bounding cycle by C^m. If the union of the arcs in ϱ does not cover the circle, then there are no bounding cycles. In this case, we let $p(C^m) = \infty$.

Let $I = (y_0, \ldots, y_{p-1})$ be a simple co-cycle in G. We call I a *bounding co-cycle*. We define the *walk* of I as we define it for bounding cycles. To compute $t(I)$, the number of times that I goes around the circle, we add 1 to the number of times that the walk of I hit $\ell(y_0)$, to count also the last partial turn. The *ratio* of I, denoted by $p(I)$, is $|I|/t(I)$. See Fig. 3. We call I a *maximum bounding co-cycle* if there is no other bounding co-cycle I' with $p(I') > p(I)$. We denote a maximum bounding co-cycle by I^M.

The circumference of a UCA model of closed arcs with a bounding cycle C can be at most $p(C)$. On the other hand, the circumference of a UCA model with a bounding co-cycle I must be strictly greater than $p(I)$. So for any UCA model $p(I^M) < p(C^m)$. The following theorem shows that this condition is also sufficient. Furthermore, the bounds do not depend on the specific model.

Theorem 3. *[17] A PCA graph G is also a UCA graph if and only if for any PCA model of G with no pair of arcs that cover the circle, $p(I^M) < p(C^m)$.*

Given a PCA model ϱ, the algorithm of Durán et al. [2] finds a minimum bounding cycle and a maximum bounding co-cycle that start from a certain arc in the model. To do so in linear time they use complicated data structures.

We relax the definitions of a bounding cycle and a bounding co-cycle, in two ways, in order to get a simple implementation. First, we allow repetitions of vertices. Second, we use paths instead of cycles.

Let $P = (x_0, \ldots, x_{p-1})$ be a path of vertices in G, not necessarily simple, such that for $i = 1, \ldots, p-1$, the arc $\hat{x}_i$ overlaps the right side of $\hat{x}_{i-1}$, in ϱ. The path P is a *bounding path*. We define the *walk* of P and count the number of times that P goes around the circle, denoted by $t(P)$, in the same way as we do it for bounding cycles. The *ratio* of P is $p(P) = |P|/t(P)$. Note that since a bounding path is not necessarily a cycle, it might be that $t(P) = 0$, in this case we assume that $p(P) = \infty$.

Let $Q = (y_0, \ldots, y_{p-1})$ be a (not necessarily simple) co-path of vertices in G. We call Q a *bounding co-path*. We define the *walk* of Q and count the number of times that Q goes around the circle, denoted by $t(Q)$, in the same way as we do it for bounding co-cycles. The *ratio* of Q is $p(Q) = |Q|/t(Q)$.

6 Certifying Algorithm for Unit Circular-Arc Graphs

Every UCA graph is a PCA graph, so we start by testing whether G is a PCA graph using the algorithm of Sect. 4. If G is not a PCA graph then it is also not a UCA graph, and the algorithm of Sect. 4 certifies that. Otherwise, if G is a PCA graph then we have a PCA model of it which we denote by ϱ. As in Sect. 5, we assume that there is no pair of arcs in ϱ that covers the circle.

We generate bounding paths and bounding co-paths in ϱ, which are simple to find. Then, we show that we can find from this set of bounding paths and co-paths a minimum bounding cycle and a maximum bounding co-cycle. We compare the ratios of the minimum bounding cycle and maximum bounding co-cycle, and use Theorem 3 to decide whether G is a UCA graph.

It can be shown that a PCA graph with a dominating vertex or a PCA graph that has a PCA model in which the union of the arcs does not cover the circle, is a UCA graph. Therefore, we assume that G does not contain any dominating vertex and that the union of arcs in ϱ covers the circle.

To obtain a certificate when G is a UCA graph, we use the algorithm of [8] to find a UCA model for G. Note, that the first two steps of [8] are to find a PCA model of G and to eliminate pairs of arcs that cover the circle. Thus, the implementation of [8] can be simplified by using the algorithm in Sect. 4.

The algorithm of [2] iteratively looks for the minimal bounding cycle and maximal bounding co-cycle that starts with every vertex in the graph. This is not necessary since the bounding cycles and bounding co-cycles are cyclic, so we only need to start from one of their vertices, not from all of them. Furthermore, we show that for every vertex $v \in V(G)$, we can start only from vertices in $N[v]$.

Let $C^m = (x_0, \ldots, x_{p-1})$ be a minimum bounding cycle in a PCA model, and let $I^M = (y_0, \ldots, y_{q-1})$ be a maximum bounding co-cycle. Let v be any fixed vertex in $V(G)$. The arcs of C^m covers the circle at least once, so there must be an arc $\hat{x}_i$ that overlaps $\hat{v}$. Because C^m is a cycle, we may assume that $\hat{x}_i$ is $\hat{x}_0$, hence, $x_0 \in N[v]$. Assume that v is not adjacent to any vertex of I^M, we can add v to I^M and get a bounding co-cycle I, with $t(I) = t(I^M)$ and $|I| = |I^M|+1$, and therefore $p(I) > p(I^M)$, contradicting the fact that I^M is maximum. Because I^M is a co-cycle, we may assume that $y_0 \in N[v]$.

For a particular vertex u, we find n bounding paths, of lengths 1 to n, each starting with u, using the following greedy algorithm. Let $u_0 = u$. We start with $P_1 = (u_0)$ as a bounding path with $t(P_1) = 0$. For $i = 1, \ldots, n-1$, we generate $P_{i+1} = (u_0, \ldots, u_i)$ by adding to P_i the vertex u_i, where $\ell(u_i)$ is the rightmost left endpoint covered by the arc $\hat{u}_{i-1}$. Such a vertex u_i exists since the circle is covered by the union of all arcs. We let $t(C_i) = t(C_{i-1}) + 1$, if the arc $\hat{u}_i$ covers $\ell(u_0)$, and otherwise $t(C_i) = t(C_{i-1})$. We stop after generating P_n. We represent

the n bounding paths by the list of vertices in P_n and the list of the values $t(P_i)$ for $i = 1, \ldots, n$.

Similarly, we find n bounding co-paths, of lengths 1 to n, each starting with u, using the following greedy algorithm. Let $u_0 = u$. We start with $Q_1 = (u_0)$ as a bounding co-path with $t(Q_1) = 1$. For $i = 1, \ldots, n-1$, we generate $Q_{i+1} = (u_0, \ldots, u_i)$ by adding to Q_i the vertex u_i, where $\ell(u_i)$ is the leftmost left endpoint not covered by the arc $\hat{u}_{i-1}$. Since u_{i-1} is not a dominating vertex, $u_{i-1}u_i \notin E(G)$. We let $t(Q_i) = t(Q_{i-1}) + 1$, if the arc $\hat{u}_i$ covers $\ell(u_0)$, and otherwise $t(Q_i) = t(Q_{i-1})$. We stop after generating Q_n. We represent the n bounding co-paths by the list of vertices in Q_n and the list of the values $t(Q_i)$ for $i = 1, \ldots, n$.

To implement these algorithms in $O(n)$ time, we identify in advance, for every arc, the rightmost left endpoint it covers, and the leftmost left endpoint not covered by it. We do so by going around the circle from left to right, starting at some left endpoint, and maintaining $\ell(x)$, the last left endpoint we encountered. When we encounter a right endpoint $r(y)$, the last left endpoint that the arc $\hat{y}$ covers is $\ell(x)$. We can find the leftmost left endpoint following each arc in the same way, by going around the circle from right to left.

Let $v_0 \in V(G)$ be a vertex with a minimum degree, so $|N[v_0]| = O(m/n)$. We find n bounding paths and n bounding co-paths that start with each of the arc of vertices in $N[v_0]$, by the greedy algorithms described above. This takes $O(n|N[v_0]|) = O(n \cdot m/n) = O(m)$ time and $O(m)$ space. We then find among the $O(m)$ bounding paths, the bounding path P^m for which $p(P^m)$ is the smallest and the bounding co-path Q^M for which $p(Q^M)$ is the largest. If there are more than one bounding paths or co-paths with the same ratio we take the shortest.

Lemma 1. *For the bounding path P^m and the bounding co-path Q^M that we have found, we have $p(P^m) \leq p(C^m)$ and $p(I^M) \leq p(Q^M)$.*

Proof. Let C^m be a maximum bounding cycle, starting with $x_0 \in N[v_0]$. Let $k = |C^m|$, since C^m is a simple cycle, $k \leq n$. Let C_i be the prefix of C^m with $|C_i| = i$. The path C_i is a bounding path. Let P_i be the bounding path of length i stating at x_0 that our greedy algorithm has found. We prove by induction on i, that the walk of C_i is a prefix of the walk of P_i. It follows that for every $i = 1, \ldots, k$, we have $t(C_i) \leq t(P_i)$, and in particular $p(C^m) = p(C_k) \geq p(P_k) \geq p(P^m)$, as required.

For $i = 1$, $C_1 = P_1$, and thus the walks of C_1 and P_1 are identical. Assume that the walk of C_i is prefix of the walk of P_i. To get C_{i+1}, we add to C_i a vertex x_i such that the last occurrence of $\ell(x_i)$ in the walk of C_i is not followed by an occurrence of $r(x_i)$. The walk of C_{i+1} starts with the walk of C_i and continues until $r(x_i)$. To get P_{i+1}, we add to P_i the vertex u_i such that $\ell(u_i)$ is the last left endpoint in the walk of P_i. The walk of P_{i+1} starts with the walk of P_i and continues until $r(u_i)$. Since by induction the walk of C_i is a prefix of the walk of P_i, the last occurrence of $\ell(x_i)$ in the walk of C_i corresponds to an occurrence of $\ell(x_i)$ in the walk of P_i preceding or equal to the last occurrence of $\ell(u_i)$ in the walk of P_i. Therefore the last occurrence of $r(x_i)$ in C_{i+1} corresponds to an occurrence of $r(x_i)$ in the walk of P_{i+1} preceding or equal to the last occurrence of $r(u_i)$ in the walk of P_{i+1}. Thus, the walk of C_{i+1} is a prefix of the walk of P_{i+1}.

The claim $p(I^M) \leq p(Q^M)$ is proved similarly. □

Lemma 2. *The bounding path P^m and the bounding co-path Q^M that we have found, are a minimum bounding cycle and a maximum bounding co-cycle.*

Proof. Let $P^m = (x_0, \ldots x_{p-1})$ and let $Q^M = (y_0, \ldots y_{q-1})$. By Lemma 1 it suffices to prove that P^m is a simple cycle and that Q^M is a simple co-cycle.

Assume that P^m is not a simple path. So, there are i, j such that $x_i = x_j$ but $i \neq j$. Let i be the minimal index for which there exists $\ell > i$ such that $x_i = x_\ell$, let j be the minimal possible value of ℓ. The way that our greedy algorithm chooses the successor of each vertex does not depend on its location on the path, so $x_{i+k} = x_{j+k}$ for every $k = 1, \ldots, (p-1) - j$. Let $C = (x_{p-(j-i)}, \ldots x_{p-1})$. The path C is simple by the way we choose x_i and x_j, C is a cycle since $x_{p-1} = x_{p-(j-i)-1}$ and since C is a suffix of P^m, it is a bounding cycle. We have a bounding cycle with a ratio $p(C) \geq p(C^m) \geq p(P^m)$. So we can truncate P^m after $x_{p-(j-i)-1}$ and get a new bounding path P with $p(P) \leq p(P^m)$ which is a prefix of P^m. This contradicts the definition of P^m, to be the shortest path with the maximal ratio that the greedy algorithm found. Similarly, we can show that Q^M is a simple co-path.

Now, assume that P^m is not a cycle. The last arc $\hat{x}_{p-1}$ does not cover $\ell(x_0)$, so it does not start a new turn around the circle. Let P be the prefix of P^m of length $p-1$. We have $|P| < |P^m|$ but $t(P) = t(P^m)$ therefore $p(P) < p(P^m)$. That is a contradiction to the way we found P^m.

Assume that Q^M is not a co-cycle. It follows that the arc $\hat{y}_{q-1}$ overlaps $\hat{y}_0$. If the arc $\hat{y}_{q-1}$ covers $r(y_0)$ then $\hat{y}_{q-2}$ covers $\ell(y_0)$, because otherwise the greedy algorithm would have chosen y_0 to be y_{q-1}. Since there is no dominating vertex in G, there is a pair of nonadjacent vertices, this pair is a bounding co-cycle with ratio 2, so we have $2 \leq p(Q^M)$. Let Q be the prefix of Q^M of length $q-2$. Since there is one arc less in Q that covers $\ell(y_0)$, we have $t(Q) = t(Q^M) - 1$. And since $p(Q^M) \geq 2$ we have $p(Q^M) \leq p(Q)$, which contradict the way we define Q^M. □

Therefore, by Theorem 3, G is a UCA graph if and only if $p(Q^M) < p(P^m)$. If G is not a UCA graph we use P^m and Q^M as a certificate.

6.1 The Certificate and the Authentication Algorithm

If G is a UCA graph then the certificate is a UCA model. This certificate can be authenticated by authenticating that it is a PCA model as in Sect. 4.3 and comparing the length of all the arcs. This can be done in $O(n+m)$ time.

If G is not a PCA graph, the certificate and its authentication algorithm are as in Sect. 4.3. The size and the authentication time of this certificate are $O(n)$.

If we decided that G is a PCA graph but not a UCA graph then we have a bounding cycle P^m, and a bounding co-cycle Q^M with $p(Q^M) \geq p(P^m)$ in a PCA model ϱ. Authenticating that ϱ is a valid PCA model without pairs of arcs that cover the circle, takes $O(m+n)$ time. We can verify in the same time bound that P^m and Q^M are bounding cycle and bounding co-cycle respectively, and compute $t(P^m)$ and $t(Q^M)$ from the model, by following the walks of P^m and Q^M.

It is also possible to construct a strong certificate that can be authenticated in $O(n)$ time. This certificate proves, using edges and non-edges between vertices of P^m and Q^M, that in any PCA model of G, P^m and Q^M are a bounding cycle and a bounding co-cycle with $p(Q^M) \geq p(P^m)$. Due to space constraint we omit the details of this certificate.

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(2):390–403, 1996.
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(1):67–78, 2006.
3. M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs.* Academic Press, 1980.
4. P. Hell and J. Huang. Certifying LexBFS recognition algorithms for proper interval graphs and proper interval bigraphs. *SIAM J. Discrete Math.*, 18(3):554–570, 2004.
5. P. Hell and J. Huang. Interval bigraphs and circular arc graphs. *J. Graph Theory*, 46(4):313–327, 2004.
6. H. Kaplan and Y. Nussbaum. A simpler linear-time recognition of circular-arc graphs. In *10th Scandinavian Workshop on Algorithm Theory (SWAT)*, Lecture Notes in Computer Science 4059, pages 41–52, 2006.
7. D. Kratsch, R. M. McConnell, K. Mehlhorn, and J. P. Spinrad. Certifying algorithms for recognizing interval graphs and permutation graphs. *SIAM J. Comput.*, 36(2):326–353, 2006.
8. M. C. Lin and J. L. Szwarcfiter. Efficient construction of unit circular-arc models. In *SODA '06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 309–315, 2006.
9. R. M. McConnell. Linear-time recognition of circular-arc graphs. *Algorithmica*, 37(2):93–147, 2003.
10. R. M. McConnell. A certifying algorithm for the consecutive-ones property. In *SODA '04: Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 768–777, 2004.
11. K. Mehlhorn and S. Näeher. *The LEDA Platform for combinatorial and geometric computing.* Cambridge University Press, 1999.
12. D. Meister. Recognition and computation of minimal triangulations for AT-free claw-free and co-comparability graphs. *Discrete Applied Math.*, 146(3):193–218, 2005.
13. D. J. Skrien. A relationship between triangulated graphs, comparability graphs, proper interval graphs, proper circular-arc graphs, and nested interval graphs. *J. Graph Theory*, 6:309–316, 1982.
14. J. P. Spinrad. *Efficient Graph Representations.* Fields Institute Monographs 19. American Mathematical Society, 2003.
15. R. E. Tarjan and M. Yannakakis. Addendum: Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM J. Comput.*, 14(1):254–255, 1985.
16. A. Tucker. Matrix characterizations of circular-arc graphs. *Pacific J. Math.*, 39(2):535–545, 1971.
17. A. Tucker. Structure theorems for some classes of circular-arc graphs. *Discrete Math.*, 7:167–195, 1974.