Algorithmica (2011) 59: 215–239
DOI 10.1007/s00453-009-9304-5

Linear-Time Recognition of Helly Circular-Arc Models and Graphs

Benson L. Joeris · Min Chih Lin · Ross M. McConnell · Jeremy P. Spinrad · Jayme L. Szwarcfiter

Received: 26 February 2008 / Accepted: 24 March 2009 / Published online: 8 April 2009

Abstract A circular-arc model $\mathcal{M}$ is a circle C together with a collection $\mathcal{A}$ of arcs of C. If $\mathcal{A}$ satisfies the Helly Property then $\mathcal{M}$ 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 algorithms 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 arti-

M.C. Lin was partially supported by UBACyT Grants X456 and X143 and by PICT ANPCyT Grant 1562.

J.L. Szwarcfiter 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, Brasil.

B.L. Joeris · R.M. McConnell
Computer Science Department, Colorado State University, Fort Collins, CO 80523-1873, USA

B.L. Joeris
e-mail: bjoeris@cs.colostate.edu

R.M. McConnell
e-mail: rmm@cs.colostate.edu

M.C. Lin (✉)
Facultad de Ciencias Exactas y Naturales, Departamento de Computación, Universidad de Buenos Aires, Buenos Aires, Argentina
e-mail: oscarlin@dc.uba.ar

J.P. Spinrad
Computer Science Department, Vanderbilt University, Nashville, TN 37235, USA
e-mail: spin@vuse.vanderbilt.edu

J.L. Szwarcfiter
Instituto de Matemática, NCE and COPPE, Universidade Federal do Rio de Janeiro, Caixa Postal 2324, 20001-970 Rio de Janeiro, RJ, Brazil
e-mail: jayme@nce.ufrj.br

cle, 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

An **interval graph** is the intersection graph of a set of intervals of a line. That is, given a set of intervals of a line, one may construct the corresponding interval graph by making a vertex of each of the intervals, and an edge between each pair of intervals that intersects. Interval graphs arise in scheduling problems, where the intervals represent time intervals occupied by tasks and the edges represent scheduling conflicts. Natural optimization problems correspond to finding a maximum independent set or a minimum coloring of the interval graph.

Before the structure of DNA was well-understood, the problem of recognizing whether a given graph is an interval graph played a role in establishing its linear topology. Seymour Benzer [1] developed a means of damaging connected regions in copies of viral DNA using X-ray photons. He created a graph where the vertices were a few scores of the damaged regions and the edges were damaged regions that contained a common gene, indicating an intersection of the damaged regions in the genome. The vast majority of graphs with this many vertices are not interval graphs, so by showing that the procedure gave rise to an interval graph, he provided compelling evidence that the fragments were segments of a substrate that has a linear topology.

This prompted interest in algorithms for recognizing whether a given graph G is an interval graph, and for characterizing properties that distinguish interval graphs from other graphs. A characterization of interval graphs as those that do not contain one of two forbidden types of subgraphs was given by Lekkerker and Boland in 1962 [12]. The first linear-time recognition algorithm was given in 1974 by Booth and Lueker [2].

Let a **clique** of a graph denote a *maximal* set of pairwise adjacent vertices. Booth and Lueker's algorithm is based on the characterization of interval graphs as exactly those graphs whose cliques have the **consecutive-ones property**, that is, that there exists a way to linearly order the cliques so that, for each vertex, the cliques that contain the vertex are consecutive in the ordering.

There are two natural generalizations of interval graphs to the circle. The first is to generalize the characterization of interval graphs as the intersection graph of intervals on a line. This gives rise to the **circular-arc (CA) graphs**, which are the intersection graphs of arcs on a circle. A **circular-arc (CA) model** of a circular-arc graph G is a set of arcs whose intersection graph is G. They have attracted much interest since their first characterization by Tucker, almost forty years ago [20]. The interest in circular-arc graphs has continued through the present. For instance, recent books such as those by Kleinberg and Tardos [11] and Spinrad [19] dedicate a fair number of pages to this class. Some of the motivations for studying circular-arc graphs are their rich structure, in addition to their applications in cyclic scheduling problems, such as those that arise in traffic light scheduling, in assignment of variables to registers in loops, and in other areas. See [6, 18].

Unfortunately, circular-arc graphs lack many of the convenient combinatorial properties of interval graphs. For instance, a circular-arc graph can have an exponential number of cliques, while the consecutive-ones characterization of interval graphs constrains them to have at most n cliques, where n is the number of vertices. When Booth and Lueker formulated their linear-time algorithm for recognizing interval graphs, Booth conjectured that recognition of circular-arc graphs would turn out to be NP-complete. This was proved false by Tucker [21], who gave an $O(n^3)$ algorithm. However, despite a great deal of work on recognition algorithms over the years, they resisted linear-time recognition until quite recently (McConnell [16, 17], Kaplan and Nussbaum [9]), partly because of failure to possess many of the combinatorial properties available to algorithms on interval graphs.

The second natural generalization of interval graphs has the advantage of capturing more of these structural properties of interval graphs, while retaining the relevance to many cyclic scheduling problems. This generalization is based on the second characterization of interval graphs, as the graphs whose cliques have the consecutive-ones property. The cliques of a graph have the **circular-ones property** if there is a way to assign them a cyclic order such that, for every vertex, the cliques that contain the vertex are consecutive in the cyclic order. A graph G is a **Helly circular-arc (HCA) graph** if it has this property.

The Helly circular-arc graphs are a special case of circular-arc graphs. Two vertices are adjacent if and only if they are contained in a common clique. Treating the consecutive block of cliques containing a vertex v as v's "arc" on the circle, one obtains a set of circular arcs whose intersection graph is exactly G. Such a model is called a **Helly circular-arc (HCA) model**. Analogously to the consecutive-ones property that characterizes interval graphs, the circular-ones property constrains them to have at most n cliques, and it forces the arcs to observe the **Helly property**, which is that any set of pairwise intersecting arcs has a nonempty intersection. Illustrations are given below in the figures. It is not hard to see that, conversely, the Helly property forces the cliques of the represented graph to have the circular-ones property; it can be obtained from a model that has the Helly property by finding the common intersection of the arcs in each clique, and ordering the cliques in the order in which these intersection points appear around the circle.

Helly circular-arc graphs were introduced in the 1970's by Gavril [5], who described a recognition algorithm that requires $O(n^3)$ time, and that is based on the circular-ones property. (See also Golumbic [6], Spinrad [19]).

Let n be the number of vertices and m the number of edges of a graph. In this paper, we propose the following results (parts of the results of the present paper were presented in the extended abstract in [13]):

1. A simple characterization of the ways that the Helly property can be violated in a CA model.

2. Characterizations of those CA graphs that are HCA graphs, including one by forbidden subgraphs.
3. An $O(n)$ recognition algorithm for models that have the Helly property.
4. A recognition algorithm for HCA graphs, with complexity $O(m)$. The time reduces to $O(n)$ if the graph is already given by any of its CA models.
5. Certificates for the recognition of Helly models. That is, if the given model $\mathcal{M}$ is not Helly, then we exhibit a simple minimal non Helly submodel of it, in $O(n)$ time.
6. Certificates for the recognition of HCA graphs. That is, if the given graph is HCA then we exhibit a Helly model of it, and if the graph is a CA graph but not HCA then a forbidden induced subgraph is displayed by the algorithm. Again, the time bound is $O(n)$, if the graph is given by any of its CA models.

In order to achieve the above $O(n)$ time bounds, we employ special functions on arcs of a circle. That is, given an arc A_i of a CA model $\mathcal{M}$, these functions compute the arc of $\mathcal{M}$ with the extremes in a desired position in relation to A_i. We believe that these functions might be useful as a tool for solving other problems involving CA models.

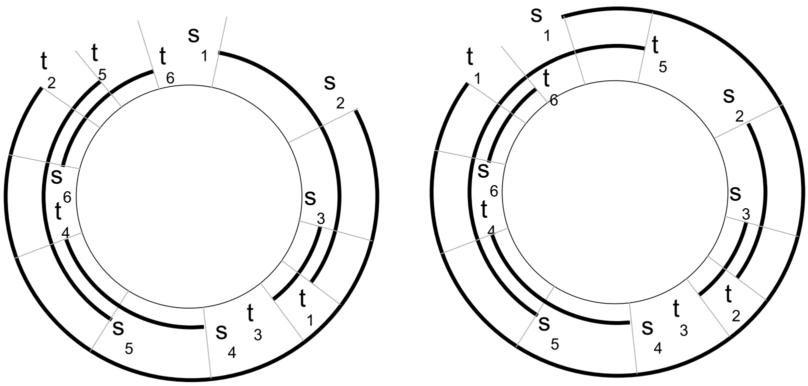
The following is the plan of the paper. In the next section, we define a special family of CA models and a special family of graphs in which the proposed characterizations are based. In Sect. 3, we characterize HCA models, while the characterizations of HCA graphs are in Sect. 4. In Sect. 5, we define the above functions on the arcs of a CA model, together with the algorithms for computing them. Section 6 describes the construction of a special CA model that is employed in the recognition algorithm. Section 7 contains the recognition algorithm for CA models, together with its certificates. Finally, in Sect. 8, we formulate the algorithm for recognizing HCA graphs and exhibiting the corresponding certificates. Some additional remarks form the last section. Other recent related work concerns recognition and characterization of other special cases of circular-arc graphs that are generalizations of special cases of interval graphs. The most common of these subclasses are the *proper circular-arc graphs*, where there exists a circular-arc intersection model where no arc contains another, (Deng, Hell and Huang [3], Kaplan and Nussbaum [10]), and the *unit circular-arc graphs*, where there exists a model where all arcs have the same length, (Lin and Szwarcfiter [14, 15], Kaplan and Nussbaum [10]).

2 Definitions

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 $v_i, v_j \in V_G$. Denote $N(v_i) = \{v_j \in V_G | v_i v_j \in E_G\}$ and $N[v] = N(v) \cup \{v\}$, call $v_j \in N(v_i)$ a ***neighbour*** of v_i and write $d(v_i) = |N(v_i)|$. A vertex $v \in V_G$ satisfying $N[v] = V_G$ is a ***universal*** vertex of G.

A ***circular-arc (CA) model*** $\mathcal{M}$ is a circle C together with a collection $\mathcal{A}$ of arcs of C. Write $\mathcal{M} = (C, \mathcal{A})$, and denote by $|C|$ the length of C. In the special case where there is a point of C that is not in any arc of $\mathcal{A}$ then $\mathcal{M}$ is an interval model, as the circle can be cut at the point and rolled out on the line, together with its arcs, which become intervals. 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 ***left point*** and t_i the ***right 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, without loss of generality, 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. Let us say that an arc of $\mathcal{A}$ is ***universal*** when it intersects all other arcs of $\mathcal{A}$. When traversing C, we obtain a circular ordering of the $2n$ extreme points of $\mathcal{A}$. These points are identified by the integers $1, \dots, 2n$ in the ordering. Furthermore, we also consider a circular ordering $A_1, \dots, A_n$ of the arcs of $\mathcal{A}$, defined by the corresponding circular ordering $s_1, \dots, s_n$ of their respective left points. In general, when dealing with a sequence $x_1, \dots, x_t$ of t objects that are 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 ordering of their arcs.

In a model $(C, \mathcal{A})$, the ***complement*** of an arc $A_i = s_i, t_i$ is the arc $\overline{A_i} = t_i, s_i$. The ***complement*** of $(C, \mathcal{A})$ is the model $(C, \overline{\mathcal{A}})$, where $\overline{\mathcal{A}} = \{\overline{A_i} | A_i \in \mathcal{A}\}$. Let us say that an arc $A_j \in \mathcal{A}$ ***right overlaps*** the arc $A_i \in \mathcal{A}$ when $s_j \in A_i$ and $t_j \in \overline{A_i}$. Similarly, A_j ***left overlaps*** A_i when $t_j \in A_i$ and $s_j \in \overline{A_i}$. See Fig. 2. In general, say that A_j ***overlaps*** A_i if A_j either left overlaps or right overlaps A_i.



(a) (b)

Fig. 1 Two circular-arc models

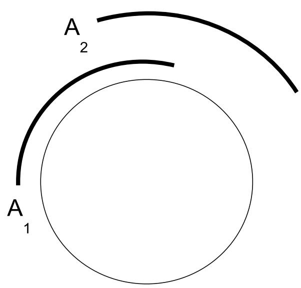


Fig. 2 A_2 right overlaps A_1

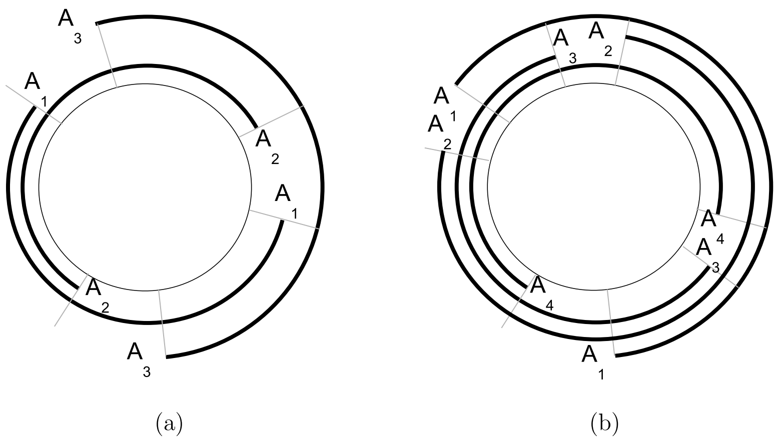


Fig. 3 Two minimally non Helly models

In the model $(C, \mathcal{A})$, a subfamily of arcs of $\mathcal{A}$ is ***intersecting*** when they pairwise intersect. Note 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***. Not all sets of arcs are Helly: three arcs can collectively cover the circle without all three meeting at a common point. 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 3 depicts two minimally non Helly models.

A Helly model associates each clique of the corresponding circular-arc graph with a region of locally maximal coverage by arcs of $\mathcal{A}$, and this gives a circular-ones ordering of the cliques, hence its intersection graph is a Helly circular-arc graph. Conversely, a circular-ones ordering of the cliques of a graph defines an HCA model: each clique is assigned a point p on the circle, and each vertex v is represented by an arc A that contains p if and only if v is a member of the clique corresponding to p. This arrangement precludes a non-Helly subset of arcs, since they would imply a clique that does not occupy a place in the circular-ones ordering.

A graph is a Helly circular-arc (HCA) graph iff there exists a Helly circular-arc model for it. Note that this does not imply that all circular-arc models of an HCA graph are Helly. As a simple example, the complete graph K_3 is an HCA graph, since it can be represented by three arcs that cover a common point, but it also has a CA model consisting of three arcs that cover the circle without intersecting at a common point.

Given a circular-arc model and a numbering $\{A_1, A_2, \ldots, A_n\}$ of its arcs, 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 $q \in C$ that is common to all those arcs of $\mathcal{A}$ that 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.

Let $\mathcal{M} = (C, \mathcal{A})$ be a CA model. We examine some subsequences of the extreme points of $\mathcal{M}$. An ***s-sequence*** (***t-sequence***) of $\mathcal{M}$ is a maximal sequence of consecutive left points (right points) of $\mathcal{A}$ in the circular ordering of C. Let an ***extreme sequence*** mean an s-sequence or t-sequence. The $2n$ extreme points are then partitioned into s-sequences and t-sequences, which alternate in C. For an extreme sequence E, the notations $NEXT(E)$ and $NEXT^{-1}(E)$ represent the extreme sequences that succeed and precede E in C, respectively. For an extreme point $p \in \mathcal{A}$, denote by $SEQUENCE(p)$ the extreme sequence that contains p, while $NEXT(p)$ means the sequence $NEXT(SEQUENCE(p))$.

Throughout the paper, we employ operations on the CA models that modify the arcs, while preserving equivalence. A simple example of such operations is to permute the extremes of the arcs within a same extreme sequence.

Next, we define a special model of interest.

Let s_i be a left point of $\mathcal{A}$ and $S = SEQUENCE(s_i)$. Let us say that s_i is ***stable*** when $i = j$ or $A_i \cap A_j = \emptyset$, for every $t_j \in NEXT^{-1}(S)$. A model $(C, \mathcal{A})$ is ***stable*** when all of its left points are stable. Let t_j be a right point of $\mathcal{A}$ and $T = SEQUENCE(t_i)$. Let us say that t_j is ***stable*** when $i = j$ or $A_i \cap A_j = \emptyset$, for every $s_i \in NEXT(T)$.

Lemma 2.1 *A model is stable precisely when all of its right points are stable.*

As examples, the models of Figs. 1(a) and 1(b) are not stable, while those of Figs. 4(b) and 5(b) are.

If $\mathcal{M} = (C, \mathcal{A})$ is a stable model and G the intersection graph of $\mathcal{A}$, then let us say that $\mathcal{M}$ is a ***stable model of*** G. We will employ stable models in the recognition process of HCA graphs.

Next, define a special family of graphs. 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:

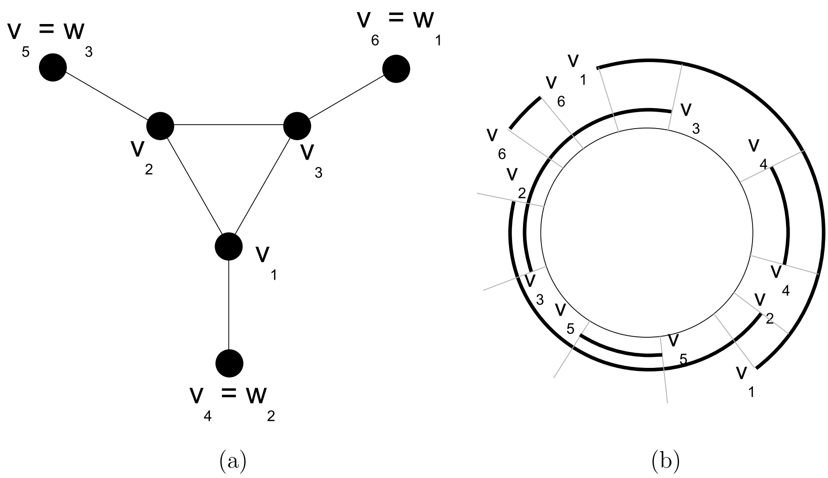


Fig. 4 An obstacle and its non Helly stable model

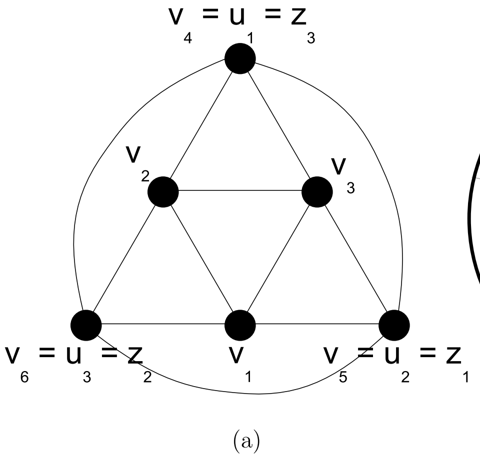


Fig. 5 Another obstacle and its non Helly stable model

(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 examples, the graphs of Figs. 4(a) and 5(a) are obstacles.

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 3.1 *A CA model $\mathcal{M} = (C, \mathcal{A})$ is HCA if and only if the following two conditions are met*:

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

Proof By hypothesis, $\mathcal{M}$ is a HCA model. Condition (i) is clear, otherwise $\mathcal{M}$ 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 $\overline{A_1}, \dots, \overline{A_k}$ 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 $p \in C$. 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 absence 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. □

The following characterizes minimally non Helly models.

Corollary 3.1 *A model $(C, \mathcal{A})$ is minimally 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 3.1 leads directly to a simple algorithm for recognizing Helly models, as follows. Given a CA model $\mathcal{M} = (C, \mathcal{A})$, verify if $\mathcal{M}$ satisfies Condition (i) and then if it satisfies Condition (ii). Clearly, $\mathcal{M}$ is HCA if and only if both conditions are satisfied. Next, we describe methods for checking them. Let G be the intersection graph of $\mathcal{A}$.

For Condition (i), we search 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, $i < j < k$. 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 common arc. 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 extreme points that form L_2 are all contained in t_i, s_i, and we consider them in the circular ordering of C. Denote by $\mathit{FIRST}(L_1)$ and $\mathit{LAST}(L_2)$ 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 $\mathit{FIRST}(L_1) = t_k$ and $\mathit{LAST}(L_2) = t_j$, for some j, k.

if $\mathit{FIRST}(L_1)$ is a left point s_q then remove s_q from L_1 and t_q from L_2
if $\mathit{LAST}(L_2)$ is a left 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 that together with A_i satisfy the above requirements, completing the computations relative to A_i. Otherwise, the arcs A_k and A_j whose right points are $\mathit{FIRST}(L_1)$ and $\mathit{LAST}(L_2)$, 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 using a radix sort. The computations relative to A_i require $O(d(v_i))$ steps where $d(v_i)$ is the degree of v_i in G. 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 [7]), the complexity of the method for verifying Condition (ii) is $O(n+m)$.

Consequently, HCA models can be recognized in $O(n+m)$ time. In Sect. 7, we describe a more efficient algorithm that recognizes HCA models in $O(n)$ time.

4 Characterizing HCA Graphs

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

Theorem 4.1 *The following conditions 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 of an obstacle, there is a circular ordering $v_1, \ldots, v_t$ of the vertices of K_t that 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 Condition (i) is satisfied then $C_i \supseteq \{w_i\} \cup K_t \setminus \{v_i, v_{i+1}\}$, otherwise Condition (ii) is satisfied and $C_i \supseteq \{u_i, z_i\} \cup K_t\{v_i, v_{i+1}\}$. 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. Suppose to the contrary that there exists a stable model $(C, \mathcal{A})$ of G that 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 3.1, A_i, A_{i+1} do not cover C. Denote $T = \mathit{SEQUENCE}(t_{i+1})$ and $S = \mathit{SEQUENCE}(s_i)$. Because $(C, \mathcal{A})$ is stable, $S \neq \mathit{NEXT}(T)$. Let $S' = \mathit{NEXT}(T)$ and $T' = \mathit{NEXT}^{-1}(S)$. 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 and t_u belong to the arc t_{i+1}, s_i,

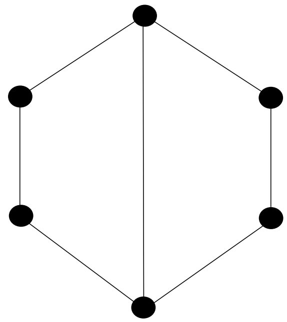


Fig. 6 A graph that is not CA and has no obstacles

Corollary 3.1 implies that s_z and t_u are 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_{i+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 the definition of obstacles holds. When none of the above alternatives occurs, the arcs A_z and A_u intersect, because s_z precedes t_u in (t_{i+1}, s_i). That is, z_i and u_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 the definition of an obstacle. 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 the requirements are 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. □

We remark that the family of obstacles does not contain all the forbidden subgraphs for a HCA graph, but restricted to the class of CA graphs. Figure 6 shows a graph that is not CA (and consequently not HCA), but does not contain obstacles.

5 Functions on Arcs

In this section, we describe some functions on graphs that will be employed in the algorithms for constructing stable models and recognizing Helly models. First, we define these functions and then describe algorithms for computing them. We consider models $(C, \mathcal{A} \cup \mathcal{B})$, where there are two families of arcs, $\mathcal{A}$ and $\mathcal{B}$ not necessarily distinct. Define the following functions from $\mathcal{B}$ into $\mathcal{A}$, such that, for $x_i, y_i = B_i \in \mathcal{B}$

* $CR_{\mathcal{A}}(B_i)$ is the arc $A_j \in \mathcal{A}$ properly contained in B_i, whose right point is closest to y_i.
* $DR_{\mathcal{A}}(B_i)$ is the arc of $\mathcal{A}$ disjoint of B_i, whose left point is closest to y_i.
* $OR_{\mathcal{A}}(B_i)$ is the arc of $\mathcal{A}$ that right overlaps B_i, whose right point is farthest from y_i.
* $OR'_{\mathcal{A}}(B_i)$ is the arc of $\mathcal{A}$ that right overlaps B_i, whose right point is closest to y_i.

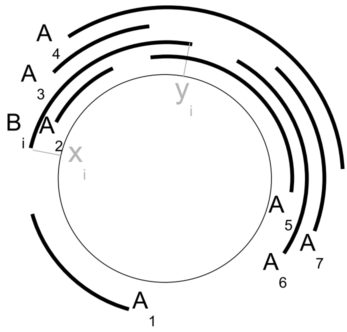


Fig. 7 Functions on arcs

In addition, define the functions CL, DL, OL and OL' in a similar way as CR, DR, OR and OR' by exchanging the roles of left and right, as well as x_i and y_i. For example, $CL_{\mathcal{A}}(B_i)$ is the arc of $\mathcal{A}$ contained in B_i, whose left point is closest to x_i. When the codomain $\mathcal{A}$ is clear from the context, we may drop it from the notation, for instance writing $CR(B_i)$, instead of $CR_{\mathcal{A}}(B_i)$.

For any of the above functions and for a given $B_i \in \mathcal{B}$, if no arc of $\mathcal{A}$ exists that satisfies it, then its value equals $\emptyset$.

On the example of Fig. 7, $CR(B_i) = A_3$, $DR(B_i) = A_6$, $OR(B_i) = A_5$, $OR'(B_i) = A_4$, and $OL(B_i) = \emptyset$.

We describe a method for computing the function $CR_{\mathcal{A}}(B_i)$, for all $B_i \in \mathcal{B}$. Let $\mathcal{A} = A_1, \ldots, A_n$ and $\mathcal{B} = B_1, \ldots, B_k$, in circular ordering, where $A_j = s_j, t_j$ and $B_i = x_i, y_i$. Denote by T and Y the set of all right points $t_j \in A_j$ and $y_i \in B_i$, respectively.

Our proposed method is divided into two parts. First, we compute the CR function for interval models. Then, we show how to transform a given CA model into an interval model, such that the CR functions of the CA model can be deduced from that of the interval model.

Next, we consider the first of the above parts. By hypothesis $\mathcal{M} = (C, \mathcal{A} \cup \mathcal{B})$ is an interval model and the aim is to compute $CR_{\mathcal{A}}(B_i)$, for each $B_i \in \mathcal{B}$. The method employs a list L, whose entries $l \in L$ are the right points $t_i \in T$ and $y_i \in Y$. Initially, L consists of all the right points of $T \cup Y$, in the same linear order as they appear in $\mathcal{M}$. The list is maintained in the decreasing order, that is, from right to left in the interval model. For $l \in L$, $LEFT(l)$ denotes the node of L that follows l, in the decreasing order. Write $LEFT(l) = \emptyset$, if l is the last node. The values $CR(B_i)$ are computed while traversing L, as follows.

The arcs $B_i \in \mathcal{B}$ are considered in increasing order of x_i. For each i, we traverse a portion of L, in decreasing order, starting from y_i, aiming to compute $CR(B_i)$. During the traversal, we ignore any right points $y_k \in Y$ that we come across. Suppose the node $l \in L$ is being visited. If l is a right point $y_p \in Y$, do nothing and proceed to $LEFT(l)$. Otherwise, l is a right point $t_j \in T$ and discuss the alternatives.

Case 1: x_i is at the left of s_j.

Then $A_j \subseteq B_i$, and furthermore t_j is closest possible to y_i, since L is being traversed in decreasing order. Consequently, $CR(B_i) = A_j$, terminating the computation relative to B_i.

Case 2: s_j is at the left of x_i.

Then $A_j \not\subseteq B_i$. Furthermore, $A_j \not\subseteq B_p$, for any $p > i$, since we are considering the arcs of $\mathcal{B}$ in increasing order of x_i. Consequently, we can remove t_j from L, without affecting any further computations.

Finally, if $l = \emptyset$, then no arc A_j of $\mathcal{A}$ is contained in B_i, implying that $CR(B_i) = \emptyset$.

The above discussion leads to an algorithm for computing $CR(B_i)$, for all $B_i \in \mathcal{B}$, as follows. Let $\mathcal{M} = (C, \mathcal{A} \cup \mathcal{B})$ be the given model. Initially, construct a list L formed by all right points of $\mathcal{M}$, in the order they appear in $\mathcal{M}$, from right to left. Then for $i = 1, \dots, k$, compute procedure $VISIT(y_i)$, below.

proc $VISIT(l)$
    if $l = \emptyset$ ***then*** $CR(B_i) := \emptyset$
    else if $l \in Y$ ***then*** $VISIT(LEFT(l))$
        else let $l = t_j \in T$
            if x_i is at the left of s_j ***then*** $CR(B_i) := A_j$
            else remove t_j from L
                $VISIT(LEFT(l))$

Following the previous discussion, we conclude that the above procedure correctly computes $CR(B_i)$, for each $B_i \in \mathcal{B}$, of an interval model $(C, \mathcal{A} \cup \mathcal{B})$. For evaluating the complexity, first note that each call of $VISIT(l)$ requires no more than constant time. So, the complexity of the algorithm corresponds to the number of calls of the procedure. When $l \in T$, each call $VISIT(l)$ terminates either with an assignment for $CR(B_i)$ or by removing some right point $t_j \in T$ from L. Consequently, the right points $l \in T$ contribute with $O(n + k)$ time to the complexity. However, the contribution of the right points $l \in Y$ may reach $O(k^2)$, since the algorithm may re-visit long sequences of right points $y_p \in Y$.

We can compute the CR values in $O(n + k)$ time, by employing a variation of the above algorithm. Basically, we use the same strategy, as for the previous algorithm. However, we transform L into a list of right points $t_j \in T$, together with subsets $Y_p \subseteq Y$ of right points of Y, instead of simply right points $t_j \in T$ and together with single right points $y_i \in Y$. Each subset Y_p is dynamically formed in L, by joining in a single subset, all the right points belonging to two subsets $Y_a, Y_b \subseteq Y$ that are consecutive members of L. The construction of each Y_p can be done by a disjoint *UNION* operation $Y_a \cup Y_b$. However, we would need to locate which subset Y_p of L contains a given right point $y_i \in Y$. This can be achieved by a $FIND(y_i)$ operation.

Next, we incorporate the above changes in procedure *VISIT*.

The initial content of list L is the same as for the previous algorithm except each y_i is replaced by $\{y_i\}$ and the external calls become $VISIT(FIND(y_i))$, for $i = 1, \dots, k$.

The procedure itself has to be modified only to handle this situation where $l \subseteq Y$. The alterations are just to replace the statement (third line)

    else if $l \in Y$ ***then*** $VISIT(LEFT(l))$

by

    else if $l \subseteq Y$ ***then*** $UNION(Y_p, l)$
                        $VISIT(LEFT(l))$

where Y_p is the subset containing y_i, and which has been determined by $FIND(y_i)$.

Observe that all *UNION* operations are with two consecutive subsets of L. Consequently, we can employ the UNION-FIND method of Gabow and Tarjan [4], as described by Itai [8]. There are $O(k)$ UNION's and FIND's. Consequently, the overall complexity of the algorithm for computing the *CR* function for an interval model is $O(n+k)$.

We remark that when $\mathcal{A}=\mathcal{B}$, we can compute the values $CR_{\mathcal{A}}(A_i)$, for all $A_i \in \mathcal{A}$, in $O(n)$ time, by a much simpler algorithm that employs no UNION-FIND structure.

Next, we consider the second part of our proposed method for computing the *CR* function for general CA models, namely to derive a convenient interval model from a general CA model, such that the *CR* function for this interval model would lead to the *CR* function for the general one.

Let $\mathcal{M}=(C,\mathcal{A})$ be a CA model and $A_1,\dots,A_n$ the arcs of $\mathcal{A}$, in the circular ordering. First, we bipartition the arcs of $\mathcal{A}$ into two kinds, relative to the left point s_1 of A_1. For $A_i \in \mathcal{A}$, say that A_i is a ***forward arc*** when $s_1 \notin A_i$, otherwise $s_1 \in A_i$ and A_i is a ***back arc***. Note that A_1 is a forward arc. Clearly, if $\mathcal{A}$ contains no back arcs then $\mathcal{M}$ is already an interval model. Otherwise, construct a model $\mathcal{M}'=(C',\mathcal{A}')$, as follows. Define $|C'|=2|C|$, while the arcs of A' are defined for each $A_i \in \mathcal{A}$, as

- if A_i is a forward arc, then A_i corresponds to two arcs $A_i', A_i'' \in \mathcal{A}'$, such that,

$$A_i' = s_i, t_i$$
$$A_i'' = s_i + |C|, t_i + |C|$$

- if A_i is a back arc, then A_i corresponds to a single arc $A_i' \in \mathcal{A}'$, such that,

$$A_i' = s_i, t_i + |C|$$

  and write $A_i'^{-1} = A_i$.

See Fig. 8.

Observe that $\mathcal{A}'$ has no back arcs, meaning that $\mathcal{M}'$ is an interval model. Call $\mathcal{M}'$ the ***associated (interval) model*** of $\mathcal{M}$. There is no difficulty to construct $\mathcal{M}'$ in $O(n)$ time, given $\mathcal{M}$.

Next, let $\mathcal{M}=(C,\mathcal{A}\cup\mathcal{B})$ and its associated model $\mathcal{M}'=(C',\mathcal{A}'\cup\mathcal{B}')$. As usual, denote $B_i = x_i, y_i$, for $B_i \in \mathcal{B}$. Suppose the CR function has been computed for $\mathcal{M}'$. That is, $CR_{\mathcal{A}'}(B_i')$ and $CR_{\mathcal{A}'}(B_i'')$ are known, for all $B_i', B_i' \in \mathcal{B}'$. Denote by $CR_{\mathcal{A}'}(B_i')^{-1} \in \mathcal{A}$, the arc of $\mathcal{M}$ corresponding to the arc $CR_{\mathcal{A}'}(B_i') \in \mathcal{A}'$ of $\mathcal{M}'$. The following theorem describes how these arcs are related.

Theorem 5.1 *Let $\mathcal{M}=(C,\mathcal{A}\cup\mathcal{B})$ be a CA model, $\mathcal{M}'=(C',\mathcal{A}'\cup\mathcal{B}')$ its associated model and $B_i \in \mathcal{B}$. Then $CR_{\mathcal{A}}(B_i) = CR_{\mathcal{A}'}(B_i')^{-1}$.*

Proof Denote $A_j = CR_{\mathcal{A}}(B_i)$. We compute A_j. Consider the alternatives.

Case 1: B_i is a forward arc.

Then A_j is also a forward arc, since $A_j \subseteq B_i$. Then $\mathcal{M}'$ contains the arcs A_j', $A_j'' \in \mathcal{A}'$ and $B_i', B_i'' \in \mathcal{B}'$. Furthermore, $A_j' \subseteq B_i'$ if only if $A_j'' \subseteq B_i''$. Consequently, there is a one-to-one correspondence between the arcs $A_p \in \mathcal{A}$ contained

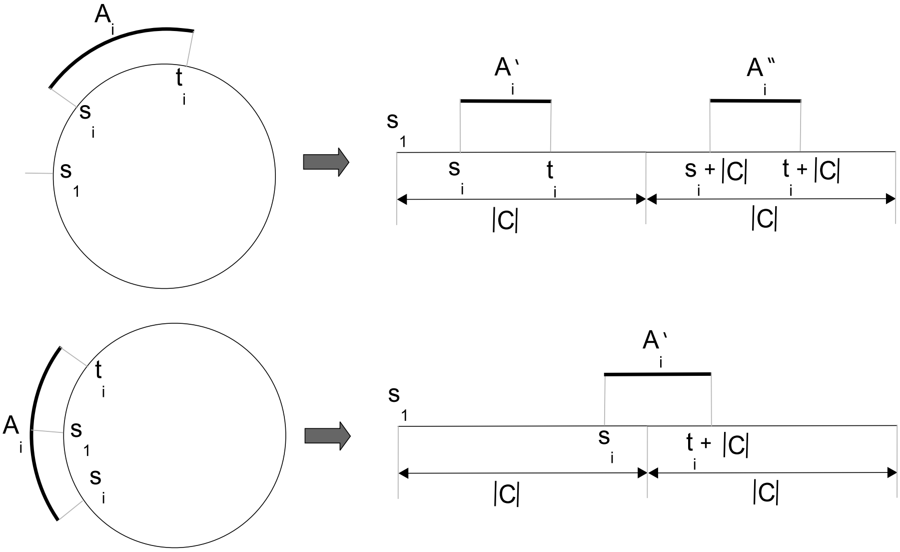


Fig. 8 The interval model $\mathcal{M}' = (C', \mathcal{A}')$

in B_i and the arcs $A'_p \in \mathcal{A}'$ contained in B'_i, preserving the circular ordering. That is, $A'_j = CR_{\mathcal{A}'}(B'_i)$.

Case 2: B_i is a back arc.

In this situation, we can partition the family of arcs $A_p \in \mathcal{A}$ contained in $B_i = x_i, y_i$ into three types, $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$, as follows. The arcs of $\mathcal{A}_1$ are those contained in the arc x_i, s_1, while $\mathcal{A}_2$ contains the arcs A_p, where $s_p \in x_i, s_1$ and $t_p \in s_1, y_i$. Finally, $\mathcal{A}_3$ is formed by the arcs of $\mathcal{A}$ contained in s_1, y_i. Observe that the arcs of $\mathcal{A}_1$ and $\mathcal{A}_3$ are all forward arcs, while those of $\mathcal{A}_2$ are back arcs. The following properties can be shown to be true, for $A_j \in \mathcal{A}$.

$$
\begin{array}{lcll}
A_j \in \mathcal{A}_1 & \Leftrightarrow & A'_j \subseteq B'_i & \text{and} \quad A''_j \cap B'_i = \emptyset \\
A_j \in \mathcal{A}_2 & \Leftrightarrow & A'_j \subseteq B'_i & \text{and} \quad B''_i = \emptyset \\
A_j \in \mathcal{A}_3 & \Leftrightarrow & A''_j \subseteq B'_i & \text{and} \quad A'_j \cap B'_i = \emptyset
\end{array}
$$

It follows that $CR_{\mathcal{A}'}(B'_i)$ must be the arc of $\mathcal{A}'$ whose image in $\mathcal{M}$ is precisely $CR_{\mathcal{A}}(B_i)$, terminating the proof. □

The algorithm for computing the *CR* function for $\mathcal{M} = (C, \mathcal{A} \cup \mathcal{B})$ can now be described. Given $\mathcal{M}$, construct its associated interval model $\mathcal{M}' = (C', \mathcal{A}' \cup \mathcal{B}')$. Then apply the algorithm formulated in this section for computing the *CR* function for $\mathcal{M}'$, obtaining the values $CR_{\mathcal{A}'}(B'_i)$, for each $B'_i \in \mathcal{B}'$. The arcs $B''_i \in \mathcal{B}'$ are disregarded. Then convert each $CR_{\mathcal{A}'}(B'_i)$ into $CR_{\mathcal{A}}(B_i)$, by finding the image of $CR_{\mathcal{A}'}(B'_i)$ in $\mathcal{M}$. The overall time bound is $O(n + k)$.

Similarly, we can compute the *CL* function in $O(n + k)$ time.

Function	Condition
$DR_{\mathcal{A}}(B_i)$	A_j and B_i are disjoint, and y_i, s_j is minimum
$DL_{\mathcal{A}}(B_i)$	A_j and B_i are disjoint, and t_j, x_i is minimum
$OR_{\mathcal{A}}(B_i)$	A_j right overlaps B_i, and y_i, t_j is maximum
$OR'_{\mathcal{A}}(B_i)$	A_j right overlaps B_i, and y_i, t_j is minimum
$OL_{\mathcal{A}}(B_i)$	A_j left overlaps B_i, and s_j, x_i is maximum

Fig. 9 Some functions on arcs

The DR and DL functions can also be obtained in $O(n+k)$ time by computing the CL and CR functions for the complements $\overline{B_i}$ of the arcs $B_i \in \mathcal{B}$, respectively, according to the following lemma, whose proof is straightforward. The following lemma is immediate.

Lemma 5.1 *Let $\mathcal{M} = (C, \mathcal{A} \cup \mathcal{B})$ be a CA model. Then $DR_{\mathcal{A}}(B_i) = CL_{\mathcal{A}}(\overline{B_i})$ and $DL_{\mathcal{A}}(B_i) = CR_{\mathcal{A}}(\overline{B_i})$, for each $B_i \in \mathcal{B}$.*

Using similar methods, we can compute the functions OR, OR', OL, OL' in $O(n+k)$ time.

Finally, we mention that all of the functions described in this section can be computed by direct methods in $O(|E(G)|)$ time, where G is the intersection graph of the given model.

In the next sections, we employ some of the above functions in the algorithms for constructing stable models and recognizing HCA graphs. In particular, we make use of the functions listed in Fig. 9. Let $\mathcal{A}, \mathcal{B}$ be two families of arcs on a circle. Each of the functions of Fig. 9 maps an arc $x_i, y_i = B_i \in \mathcal{B}$ into the arc $s_j, t_j = A_j \in \mathcal{A}$, satisfying the corresponding condition. When no arc of $\mathcal{A}$ exists that can satisfy the required condition, then assign the function is assigned the value $\emptyset$.

6 Constructing Stable Models

In this section, we describe an algorithm for transforming a given model into a stable model, equivalent to it. Such a transformation is required for applying the characterization of HCA models in terms of stable models. Without loss of generality, we assume that the given model has no universal arcs.

Let $\mathcal{M} = (C, \mathcal{A})$ be a CA model and $A_1, \ldots, A_n$ a circular ordering of the arcs of $\mathcal{A}$. The idea is to stretch as far as possible all the extremes of the arcs, while preserving adjacencies. Define the following operations on the right and left points of the arcs. We employ the functions DL and DR, described in Sect. 5.

STRETCH LEFT:
Compute $A_p := DL(A_i)$, for each arc $A_i \in \mathcal{A}$. Then move each s_i to the left, so as to be just after t_p.
STRETCH RIGHT:
Compute $A_p := DR(A_j)$, for each arc $A_j \in \mathcal{A}$. Then move each t_j to the right, so as to be just before s_p.

The following lemmas are clear.

Lemma 6.1 *Let $\mathcal{M}$ be a CA model with no universal arcs, t_j a right point of it and $s_i \in NEXT(t_j)$. Then $i \neq j$.*

Lemma 6.2 *The operations STRETCH LEFT and STRETCH RIGHT preserve the intersections of the arcs.*

We transform a given model into a stable model by repeatedly applying the stretching operations. We show that two applications of the operations, together with a reordering of the left points, are sufficient to leading to a stable model. The reordering is an additional operation that permutes the left points belonging to a same s-sequence, as follows.

> *REORDER*:
> Order the left points of each s-sequence S, so as to satisfy: s_i precedes s_j precisely when t_j precedes t_i, for all $s_i, s_j \in S$.

Observe that after the *REORDER* operation, each set of arcs, having left point in a same s-sequence, becomes linearly ordered by inclusion. The arcs in a same s-sequence appear in decreasing order.

The algorithm for constructing stable models is described next. The input is a CA model $\mathcal{M} = (C, \mathcal{A})$.

Algorithm 6.1 *STABLE MODEL*

1. *STRETCH LEFT*
2. *REORDER*
3. *STRETCH RIGHT*

Theorem 6.1 *The above algorithm transforms $\mathcal{M}$ into an equivalent stable model.*

Proof Let $\mathcal{M}_i$ be the model obtained by the algorithm, at the end of Step i, $i = 1, 2, 3$. By Lemma 6.2, the operations *STRETCH RIGHT* and *STRETCH LEFT* preserve intersections. Clearly, so does *REORDER*. Hence all models $\mathcal{M}_i$ are equivalent to $\mathcal{M}$. We show that $\mathcal{M}_3$ is a stable model.

Since $\mathcal{M}$ has no universal arcs, the *STRETCH LEFT* operation assures that $\mathcal{M}_1$ has the following property (i): for each left point $s_i \in S$, there is some $t_j \in T$, satisfying $A_i \cap A_j = \emptyset$, for any s-sequence S of $\mathcal{M}_1$ and $T = NEXT^{-1}(S)$. Moreover, a stronger fact holds. Let t_p be the right point of T, such that A_p contains the minimum number of sequences. It then follows from property (i) that $A_p \cap A_i = \emptyset$, for all $s_i \in S$. Then $\mathcal{M}_1$ also satisfies the stronger property (ii): each t-sequence T contains a stable right point.

In the sequel, the algorithm constructs $\mathcal{M}_2$. As a result, property (i) and (ii) are preserved, since the *REORDER* operation only possibly permutes left points, within a same s-sequence. However, the arcs whose left points belong to a same s-sequence are now linearly ordered by inclusion, in decreasing order.

Finally, the algorithm performs the *STRETCH RIGHT* operation and obtains $\mathcal{M}_3$. Examine the extreme sequences of $\mathcal{M}_3$. Recall that each t-sequence T of $\mathcal{M}_2$ contains a stable right point t_p. Consequently, t_p cannot be moved during the *STRETCH RIGHT* operation, beyond its t-sequence. We can conclude that any s-sequence of $\mathcal{M}_3$ is a subsequence of an s-sequence of $\mathcal{M}_2$. Additionally, t_p is also stable in $\mathcal{M}_3$. Consider any other right point t_j of $\mathcal{M}_2$. Clearly, t_j can have been moved, or not, during the *STRETCH RIGHT* operation. Let s_i be the first left point of $NEXT(t_j)$ in $\mathcal{M}_3$. Then $A_i \cap A_j = \emptyset$. Because of the *REORDER* operation, all the left points s_k which lie after s_i in $NEXT(t_j)$ satisfy $A_k \subset A_i$. Consequently, $A_k \cap A_j = \emptyset$, meaning that t is stable. By Lemma 2.1, $\mathcal{M}_3$ is stable. □

Next, we determine the complexity of the algorithm. The *STRETCH LEFT* and *STRETCH RIGHT* operations first require the computation of the *DL* and *DR* functions. These can be done in $O(n)$ time for all arcs, according to Sect. 5. After the computation of the corresponding function, for all arcs, we know already to which position each extreme point should be placed. Then all the movements can be performed simply by rewriting the model, in the circular ordering, according to the requirements. Consequently, moving all the extreme points also require $O(n)$ time. The *REORDER* operation can be performed in $O(n)$ time. Employing sorting techniques, all the left points can be ordered in $O(n)$ time. Consequently, the overall time bound is $O(n)$.

Finally, we mention that handling universal arcs is simple. If the given model $\mathcal{M} = (C, \mathcal{A})$ contains universal arcs, first we remove them. This can be done in $O(n)$ time, for example, by computing the $DR(A_i)$ values, for each arc $A_i \in \mathcal{A}$. Clearly, A_i is a universal arc precisely when $DR(A_i) = \emptyset$. Let U be the set of universal arcs of $\mathcal{A}$, and $\mathcal{M}'$ the stable model equivalent to $(C, \mathcal{A} \setminus U)$ constructed by the above algorithm. A stable model equivalent to $\mathcal{M}$ can then be obtained as follows. For each universal arc $A_i \in U$, include in $\mathcal{M}'$, a new t-sequence T_i and a new s-sequence S_i, containing solely t_i and s_i, respectively, and satisfying $S = NEXT(T)$.

Corollary 6.1 *Every CA graph admits a stable model.*

7 Recognition of HCA Models

In this section, we describe an algorithm for recognizing HCA models that runs in $O(n)$ time. The algorithm is based on the characterizations of HCA models, given by Theorem 3.1 and Corollary 3.1.

Let $\mathcal{M} = (C, \mathcal{A})$ be a CA model. A ***cover*** of $\mathcal{M}$ is a subset of arcs $\mathcal{C} \subseteq \mathcal{A}$ containing all points of C. Let us say that $\mathcal{C}$ is ***minimal*** when $\mathcal{C} \setminus \{A_i\}$ is not a cover, for any $A_i \in \mathcal{C}$. Following Sect. 3, we know that HCA models are exactly those whose complements do not admit a minimal cover of size ≥ 3. We describe a method for verifying whether a given model admits such a cover. The following definitions are employed.

Let $b \in C$ be a point of C, and $\mathcal{B} \subseteq \mathcal{A}$ the set of arcs of $\mathcal{A}$ containing b. Clearly, $\mathcal{M} - \mathcal{B} = (C, \mathcal{A} \setminus \mathcal{B})$ is an interval model. The family of minimal covers of size ≥ 3

can be partitioned into three types relative to b. A ***type 1 cover*** is a minimal cover of size ≥ 3 having exactly one arc of $\mathcal{B}$. A ***type 2 cover*** has size $= 3$ and two arcs of $\mathcal{B}$, while a ***type 3 cover*** has size > 3 and also two arcs of $\mathcal{B}$.

Lemma 7.1 *Any minimal cover of size* ≥ 3 *is either of type* 1, 2 *or* 3.

Proof Let $\mathcal{C}$ be a minimal cover of size ≥ 3 of the model $\mathcal{M} = (C, \mathcal{A})$. Clearly, $\mathcal{C}$ must contain some arc of $\mathcal{B}$, otherwise it does not cover C. Suppose $\mathcal{C}$ has three arcs B_1, B_2, B_3 containing b. Without loss of generality, let B_1 be the arc among B_1, B_2, B_3 having left point farthest from b, while B_2 has its right point farthest from b. Then B_3 is contained in $B_1 \cup B_2$, contradicting $\mathcal{C}$ to be minimal. $\square$

We handle separately the types of covers and describe methods for recognizing each of them. Some additional notation is needed.

The (***right***) ***overlap digraph*** of a model $\mathcal{M} = (C, \mathcal{A})$ is a digraph having as vertices the arcs $A_i \in \mathcal{A}$, and where there is a directed edge from A_i to A_j precisely when A_j right overlaps A_i. Denote by $\mathcal{F}_{\mathcal{M}}$ the spanning subdigraph of the overlap digraph of $\mathcal{M}$, such that there is an edge from A_i to A_j in $\mathcal{F}_{\mathcal{M}}$ when $A_j = OR_{\mathcal{A}}(A_i)$. In case $OR_{\mathcal{A}}(A_i) = \emptyset$ then A_i, has no outgoing edges. Clearly, if $\mathcal{M}$ is an interval model then $\mathcal{F}_{\mathcal{M}}$ is a directed rooted in-forest, called the ***longest right forest*** of $\mathcal{M}$. In this case, for $A_i \in A$ denote by $ROOT_{\mathcal{M}}(A_i)$ the arc of $\mathcal{A}$ that is the root of the tree in $\mathcal{F}_{\mathcal{M}}$ that contains A_i. Figure 10(a) depicts a model $\mathcal{M} = (C, \mathcal{A})$ and a point $b \in C$. Its overlap digraph is shown in Fig. 10(b), while Fig. 10(c) represents the longest right forest of the interval model $\mathcal{M}' = (C, \mathcal{A} \setminus \mathcal{B})$, with $\mathcal{B} = \{A_6, A_7\}$.

The following lemma is clear.

Lemma 7.2 *Let* $\mathcal{M} = (C, \mathcal{A})$ *be an interval model,* $A_i \in \mathcal{A}$ *an arc of* $\mathcal{A}$*,* $p \in C$ *a point located at the right of* t_i*, and* $\mathcal{F}_{\mathcal{M}}$ *the longest right forest of* $\mathcal{M}$*. Then the overlap digraph of* $\mathcal{M}$ *has a path from* A_i *to some arc containing* p *if and only if* $\mathcal{F}_{\mathcal{M}}$ *has such a path.*

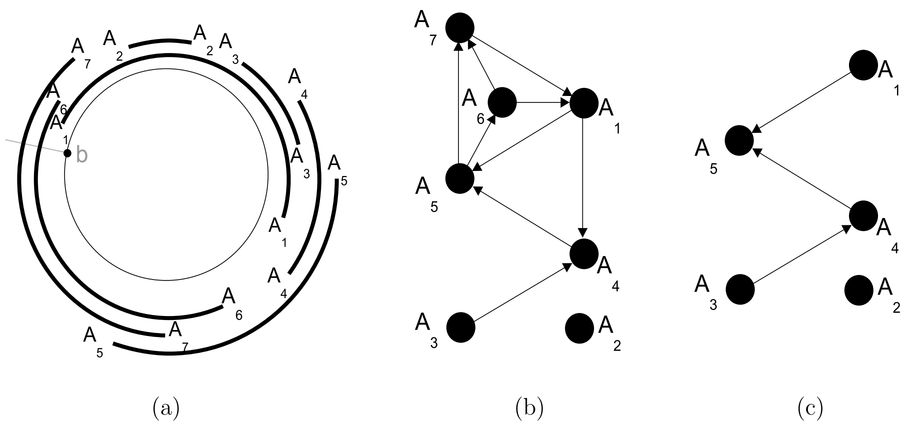


Fig. 10 An overlap digraph and a longest right forest

In the sequel, we characterize the types of covers. Let $\mathcal{M} = (C, \mathcal{A})$ be a CA model, b a point of C and $\mathcal{B} \subseteq \mathcal{A}$ the set of arcs containing b. Denote $\mathcal{A}' = \mathcal{A} \setminus \mathcal{B}$ and $\mathcal{M}' = (C, \mathcal{A}')$. Clearly, $\mathcal{M}'$ is an interval model. For an arc $B_i \in \mathcal{B}$, let x_i and y_i denote its left and right points, respectively. Finally, let $\mathcal{F}_{\mathcal{M}'}$, be the longest right forest of $\mathcal{M}'$.

Start with type 1 covers.

Theorem 7.1 *$\mathcal{M}$ has a type 1 cover if and only if for some $B_i \in \mathcal{B}$,*

$$ROOT_{\mathcal{M}'}(A_j) \cap B_i \neq \emptyset, \quad \textit{such that}$$

$$A_j = OR_{\mathcal{A}'}(B_i) \neq \emptyset.$$

Proof Let $\mathcal{C} = \{B_i, A'_1, \ldots, A'_l\}$ be a type 1 cover of $\mathcal{M}$, in circular ordering, $A_i \in \mathcal{A}'$, $B_i \in \mathcal{B}$. It follows that $A_j = OR_{\mathcal{A}'}(B_i) \neq \emptyset$, because $A'_1 \neq \emptyset$. Also, $l \geq 2$. In a minimal cover, any two consecutive arcs must overlap. Consequently, the overlap digraph of $\mathcal{M}'$ has a path from A'_1 to A'_l. On the other hand, B_i right overlaps A'_l, while A'_1 right overlaps B_i. The latter implies $x_i \in A'_l \setminus A'_1$. By applying Lemma 7.2, we conclude that $\mathcal{F}_{\mathcal{M}'}$ also has a path from A'_1 to some arc containing x_i. On the other hand, A_j either contains or right overlaps A'_1. In addition, $x_i \notin OR_{\mathcal{A}'}(B_i)$. Consequently, $\mathcal{F}_{\mathcal{M}'}$ must also contain a path from $OR_{\mathcal{A}'}(B_i)$ to some arc A'' containing x_i. Consequently, $A'' \cap B_i \neq \emptyset$. Furthermore, any ancestor A_k of A'' in $\mathcal{F}_{\mathcal{M}'}$ also intersects B_i, because $t_k \in x_i, b$ and y_i is at the right of b in $\mathcal{M}$. In particular $ROOT_{\mathcal{M}'}(A_j)$ also intersects B_i terminating the proof.

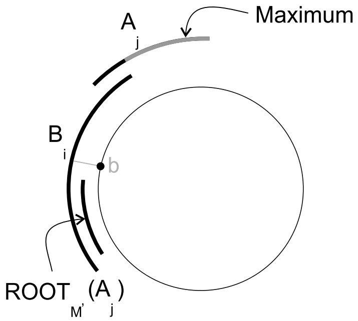
Conversely, by hypothesis $A_j = OR_{\mathcal{A}'}(B_i) \neq \emptyset$ and $ROOT_{\mathcal{M}'}(A_j) \cap B_i \neq \emptyset$. In addition, $A_j \setminus B_i$ is maximum. See Fig. 11. Clearly, $x_i \notin A_j$. Let A_k be the nearest ancestor of A_j in $\mathcal{F}_{\mathcal{M}'}$ containing x_i. Clearly, $A_k \neq A_j$, because $x_i \notin A_j$.

Consequently, the arcs in the path of $\mathcal{M}'$ from A_j to A_k, together with the arc B_i form a type 1 cover of $\mathcal{M}$. □

Next, we describe a characterization for type 2 covers.

Theorem 7.2 *$\mathcal{M}$ has a type 2 cover if and only if for some $A_i \in \mathcal{A}'$,*

Fig. 11 A case of Theorem 7.1



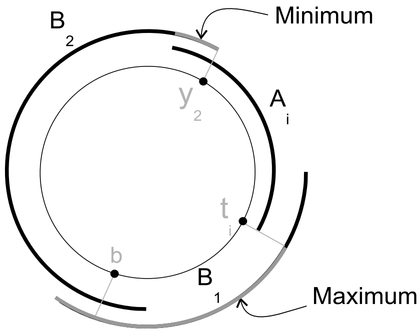


Fig. 12 A case of Theorem 7.2

$$y_1 \in \overline{A_i}, \quad \textit{where}$$
$$B_1 = OR'_{\mathcal{B}}(y_2, t_i) \neq \emptyset, \quad \textit{and}$$
$$B_2 = OR'_{\mathcal{B}}(\overline{A_i}) \neq \emptyset.$$

Proof Let A_i, B'_1, B'_2 be a type 2 cover, in circular ordering, $A_i \in \mathcal{A}'$ and $B'_1, B'_2 \in \mathcal{B}$. Then B'_2 right overlaps $\overline{A_i}$, implying that $B_2 = OR'_{\mathcal{B}}(\overline{A_i}) \neq \emptyset$. Also, B'_1 right overlaps y'_2, t_i meaning that it also right overlaps y_2, t_i, implying $B_1 \neq \emptyset$. Finally, $y'_1 \in \overline{A_i}$, which means that $y_1 \in \overline{A_i}$, completing the proof. □

Conversely, assume the stated conditions hold. Then A_i right overlaps B_2, and B_1 also right overlaps A_i. In addition, B_1 and B_2 do not intersect in A_i, because B_1 right overlaps y_2, t_i. Furthermore, B_2 and B_1 also overlap, because $B_1 \cap B_2 \neq \emptyset$, $s_i \in B_2 \setminus B_1$ and $t_i \in B_1 \setminus B_2$. We also know that $B_2 \setminus A_i$ is minimum, while $B_1 \setminus A_i$ is maximum, from the definitions of functions OR and OR', respectively. See Fig. 12. Consequently, A_i, B_1, B_2 cover C and no two of these arcs do.

Below is a characterization for type 3 covers of CA models that do not admit neither type 1 nor type 2 covers.

Theorem 7.3 *Let $\mathcal{M}$ be a CA model, admitting neither type* 1 *nor type* 2 *covers. Then $\mathcal{M}$ admits a type* 3 *cover if and only if for some $A_i \in \mathcal{A}'$,*

$$ROOT_{\mathcal{M}'}(A_j) \cap B_1 \neq \emptyset, \quad \textit{where}$$
$$A_j = OR_{\mathcal{A}'}(y_2, t_i) \neq \emptyset,$$
$$B_1 = OL_{\mathcal{B}}(x_2, s_i) \neq \emptyset, \quad \textit{and}$$
$$B_2 = OR'_{\mathcal{B}}(\overline{A_i}) \neq \emptyset.$$

Proof Suppose $\mathcal{M}$ has a type 3 cover $\mathcal{C}$ and no type 1 nor type 2 covers. Let $\mathcal{C}$ be formed by the arcs $B'_2, A'_1 \ldots, A'_l, B'_1$, in circular ordering, $B'_1, B'_2 \in \mathcal{B}$, with $A'_1 = A_i$, and $A'_k \in \mathcal{A}'$, $1 \leq k \leq l$ and $l \geq 2$. Because B'_2 left overlaps A'_1, we can conclude that $B_2 \neq \emptyset$. Since s_i, y_2 has minimum length, the left point of A'_2 is at the right of y_2, inside y_2, t_i. Consequently, $A_j = OR_{\mathcal{A}'}(y_2, t_i) \neq \emptyset$. Next, we discuss B_1.

First, suppose $B_1 = \emptyset$. In this situation, $B_2 \neq B'_2$ and $y'_1 \in B_2$. Consequently, $B_2 \supseteq B'_1$. The latter implies that some arc $A'_k \in \mathcal{C} \setminus \mathcal{B}$ left overlaps B_2, because $\mathcal{C}$

is a cover. Choose A'_k, having minimum k. If $k = 1$ then $A'_1 = A_i$ left overlaps and right overlaps B_2, a contradiction. For $k > 1$, $B_2, A'_1, \ldots, A'_k$ is a type 1 cover, again a contradiction. Then $B_1 = \emptyset$ can not occur. That is, $B_1, B_2, A_j \neq \emptyset$. It remains to examine the root of the tree of $\mathcal{F}'_{\mathcal{M}}$ containing A_j.

In the sequel, let $B_1 = B'_1$. Then the cover $B'_2, A'_1, \ldots, A'_l, B'_1$ implies that the overlap digraph of $\mathcal{M}'$ has a path from A'_x to B_1, $1 \leq x \leq l$. Consequently, the overlap digraph has a path from A_j to B_1. That is, $\mathcal{F}'_{\mathcal{M}}$ has a path from A_j to some arc $A_p \in \mathcal{A}'$, containing x_1.

Finally, let $B_1 \neq B'_1$, and we show that a similar fact holds. Compare the positions in C of x_1 and x'_1. First, suppose x'_1 is at the left of x_1. Then $B'_1 \cap B_2 = \emptyset$, otherwise x_1 is at the left of x'_1, because B_1 left overlaps B_2 and extends maximally to the left of x_2. However, B'_1 and B_2 intersect at b, which eliminates this alternative. Consequently, x_1 must be at the left of x'_1. Because $\mathcal{C}$ is a type 3 cover, $x'_1 \in A'_l$ and the overlap digraph of $\mathcal{M}'$ has a path from A'_x to A'_l, $1 \leq x \leq l$. If $A_j \cap B_1 \neq \emptyset$ then $\mathcal{F}'_{\mathcal{M}}$ clearly has a path from A_j to some arc $A_p \in \mathcal{A}'$ containing x_1. Otherwise, observe that since $\mathcal{C}$ is a cover, the left point of A'_2 belongs to the arc y_2, t_i. Consequently the overlap digraph of $\mathcal{M}$ also has a path from A_j to A'_l. By Lemma 7.2, $\mathcal{F}_{\mathcal{M}'}$ has a path from A_j to some arc $A_p \in \mathcal{A}'$ containing x_1.

Then $A_p \cap B_1 \neq \emptyset$. If $A_p = ROOT_{\mathcal{M}'}(A_j)$ the proof is complete. Otherwise, let A_q be a proper ancestor of A_p in $\mathcal{F}_{\mathcal{M}'}$. Since $b \in B_1$, $b \notin A_p$, A_q and $A_p \cap B_1 \neq \emptyset$, it follows $A_q \cap B_1 \neq \emptyset$. In particular, $ROOT_{\mathcal{M}'}(A_j) \cap B_1 \neq \emptyset$, as desired.

Conversely, by hypothesis $ROOT_{\mathcal{M}}(A_j) \cap B_1 \neq \emptyset$, with A_j, B_1, B_2 having the stated values. See Fig. 13, where the minimum and maximum follow from the definitions of functions OR', OL and OR. Let $\mathcal{C}$ be the following sequence of arcs, in circular ordering

$$B_2, A'_1, \ldots, A'_k, B_1,$$

where $A'_1 = A_i$, $A'_2 = A_j$ and $A'_2, \ldots, A'_k$ is the path in $\mathcal{F}_{\mathcal{M}'}$ from A'_2 to its nearest ancestor A'_k that intersects B_1, $k \geq 2$. By hypothesis, $ROOT_{\mathcal{M}'}(A_j) \cap B_1 \neq \emptyset$, which implies that $A'_2, \ldots, A'_k$ exists. Furthermore, for $1 \leq q \leq k-2$, $A'_q \cap A'_{q+2} = \emptyset$, otherwise $A'_{q+1} \neq OR_{\mathcal{A}'}(A'_q)$, a contradiction. Also, B_2 and A'_1 overlap by construction, so does A'_1 and A'_2. On the other hand, A'_k must overlap B_1, because of the minimality of k and considering that necessarily $A'_k \cap B_2 = \emptyset$, otherwise $B_2, A'_1, \ldots, A'_k$ is a type 1 cover, a contradiction. Consequently, $B_2, A'_1, \ldots, A'_k, B_1$ is a type 3 cover of $\mathcal{M}$. □

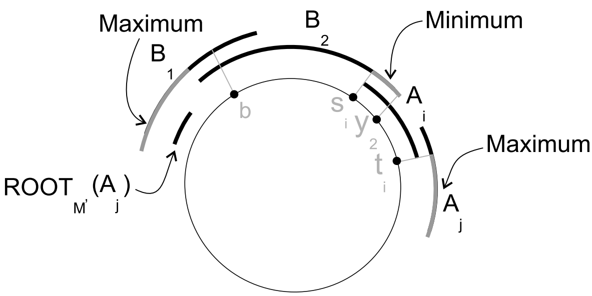


Fig. 13 A case of Theorem 7.3

The algorithm for deciding if a given CA model $\mathcal{M}$ contains a minimal cover of size ≥ 3 consists of applying Theorems 7.1, 7.2 and 7.3, in this order, for verifying if $\mathcal{M}$ contains a type 1, type 2 or type 3 cover. For recognizing if a given model $\mathcal{M}$ is HCA, apply this algorithm to the complement model $\overline{\mathcal{M}}$ of $\mathcal{M}$. Then $\mathcal{M}$ is HCA precisely when $\overline{\mathcal{M}}$ does not contain covers of any types.

We describe the algorithm for recognizing the cover types. Given $\mathcal{M} = (C, \mathcal{A})$, choose a point $b \in C$ and construct the set $\mathcal{B} \subseteq \mathcal{A}$ of the arcs containing b. Let $\mathcal{A}' = \mathcal{A} \setminus \mathcal{B}$. Compute all the OR, OL and OR' functions involved in Theorems 7.1, 7.2 and 7.3, using the algorithms of Sect. 5. Construct the longest right forest $\mathcal{F}_{\mathcal{M}'}$ of $\mathcal{M}' = (C, \mathcal{A}')$. Then, for each $B_i \in \mathcal{B}$, apply Theorem 7.1, looking for type 1 cover. Afterwards, for each $A_i \in \mathcal{A}'$ apply Theorem 7.2 for type 2 covers. If no cover has been found so far then apply Theorem 7.3, for each $A_i \in \mathcal{A}'$.

As for the complexity, first observe that there are $O(n)$ values of functions OR, OL and OR' to be computed. By Sect. 5, all these values can be computed in $O(n)$ time. The construction of $\mathcal{F}_{\mathcal{M}'}$ also takes $O(n)$ time. Finally, each application of Theorems 7.1, 7.2 or 7.3 can be done in $O(n)$ time.

As for finding negative certificates, it follows from Corollary 3.1 that the complement of any of the cover types is a minimal violation for $\mathcal{M}$ being Helly. Consequently, it represents a negative certificate, which can also be displayed in $O(n)$ time. The (converse) proofs of Theorems 7.1, 7.2 and 7.3 provide the details of the algorithm for producing such negative certificates.

The validation of this certificate can be easily done in $O(n)$ time, by finding the complement $\overline{\mathcal{M}}$ of $\mathcal{M}$ and checking if either the arcs of $\overline{\mathcal{M}}$ form a cycle of length 3, or a chordless cycle of length > 3.

8 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 [9, 17] to recognize whether G is a CA graph. In the affirmative case, let $\mathcal{M}$ be the model constructed by any of the algorithms. Otherwise terminate the algorithm (G is not HCA).
2. Transform G into a stable model, applying the algorithm of Sect. 6.
3. Verify if $\mathcal{M}$ is a HCA model, applying the algorithm of Sect. 7. Then terminate the algorithm (G is HCA if $\mathcal{M}$ is HCA, and otherwise G is not HCA).

The correctness of the algorithm follows directly from Theorem 4.1 and from the correctness of the algorithms of Sects. 6 and 7.

Step 1 requires $O(n+m)$ time, and Steps 2 and 3 terminate within $O(n)$ time.

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 induced subgraph of G, that is, an obstacle. Such a forbidden induced subgraph can be obtained in $O(n)$ time from the negative certificate of the stable model of G not being Helly, and following the proof $(b) \Rightarrow (c)$ of Theorem 4.1.

The validation of this certificate follows from the validation of its corresponding non Helly stable model, before described. In fact, we can exhibit both the forbidden subgraph and the non Helly submodel and in linear time confirm that the latter is a model for the subgraph.

9 Conclusions

We have described new characterizations and a linear-time algorithm for recognizing Helly circular-arc graphs. In case the given graph G is indeed a HCA graph, the algorithm produces a HCA model for it. Otherwise, if G is a CA graph, but no HCA, then the algorithm exhibits a certificate of this fact, in terms of a forbidden induced subgraph. The complexity of the algorithm is $O(n+m)$. However, if the input already consists of a CA model of G, the complexity reduces to $O(n)$.

However, except for its linear-time recognition and model construction, the same as above is so far not known for the general class of circular-arc graphs. So, the following open problems would be of interest.

1. Describe a characterization by forbidden induced subgraphs for circular-arc graphs.
2. Describe an algorithm for finding a certificate for a graph not to be a circular-arc graph.
3. Describe a linear-time algorithm for solving isomorphism of circular-arc graphs.

References

1. Benzer, S.: On the topology of the genetic fine structure. Proc. Natl. Acad. Sci. USA **45**, 1607–1620 (1959)
2. Booth, S., Lueker, S.: Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. J. Comput. Syst. Sci. **13**, 335–379 (1976)
3. Deng, X., Hell, P., Huang, J.: Linear time representation algorithms for proper circular-arc graphs and proper interval graphs. SIAM J. Comput. **25**, 390–403 (1996)
4. Gabow, H.N., Tarjan, R.E.: A linear-time algorithm for a special case of disjoint set union. J. Comput. Syst. Sci. **30**, 209–221 (1985)
5. Gavril, F.: Algorithms on circular-arc graphs. Networks **4**, 357–369 (1974)
6. Golumbic, M.C.: Algorithmic Graph Theory and Perfect Graphs, 2nd edn. Academic Press, New York (2004)
7. Habib, M., McConnell, R., Paul, C., Viennot, L.: Lex-bfs and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. Theor. Comput. Sci. **234**, 59–84 (2000)
8. Itai, A.: Linear time restricted union/find. Manuscript (2006)
9. Kaplan, H., Nussbaum, Y.: A simpler linear-time recognition of circular-arc graphs. In: The Tenth Scandinavian Workshop on Algorithm Theory (SWAT'06). Lecture Notes in Computer Science, vol. 4059, pp. 289–300. Springer, Berlin (2006)
10. Kaplan, H., Nussbaum, Y.: Certifying algorithms for recognizing proper circular-arc graphs and unit circular-arc graphs. In: Proceedings of the 32nd Workshop on Graph Theoretic Concepts in Computer Science (WG'06). Lecture Notes in Computer Science, vol. 4271, pp. 289–300. Springer, Berlin (2006)
11. Kleinberg, J., Tardos, E.: Algorithm Design. Addison-Wesley, Boston (2006)
12. Lekkerker, C.G., Boland, J.C.: Representation of finite graphs by a set of intervals on the real line. Fund. Math. **51**, 45–64 (1962)
13. Lin, M.C., Szwarcfiter, J.L.: Characterizations and linear time recognition of Helly circular-arc graphs. In: Proceedings of the 12th Annual International Conference on Computing and Combinatorics (COCOON'06). Lecture Notes in Computer Science, vol. 4112, pp. 73–82. Springer, Berlin (2006)

14. Lin, M.C., Szwarcfiter, J.L.: Efficient construction of unit circular-arc models. In: Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'06), pp. 309–315 (2006)
15. Lin, M.C., Szwarcfiter, J.L.: Unit circular-arc graph representations and feasible circulations. SIAM J. Discrete Math. **22**, 409–423 (2008)
16. McConnell, R.M.: Linear-time recognition of circular-arc graphs. In: Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science (FOCS'01), pp. 386–394 (2001)
17. McConnell, R.M.: Linear-time recognition of circular-arc graphs. Algorithmica **37**(2), 93–147 (2003)
18. Roberts, F.S.: Graph Theory and its Applications to Problems of Society. Society for Industrial and Applied Mathematics, Philadelphia (1978)
19. Spinrad, J.: Efficient Graph Representations. Am. Math. Soc., Providence (2003)
20. Tucker, A.: Characterizing circular-arc graphs. Bull. Am. Math. Soc. **76**, 1257–1260 (1970)
21. Tucker, A.: An efficient test for circular-arc graphs. SIAM J. Comput. **9**, 1–24 (1980)