Algorithms for clique-independent sets on subclasses of circular-arc graphs

Guillermo Durán [a,1] , Min Chih Lin [b,2],
Sergio Mera [b,2] and Jayme Luiz Szwarcfiter [c,3]

[a] *Departamento de Ingeniería Industrial, Facultad de Ciencias Físicas y Matemáticas, Universidad de Chile, Santiago, Chile.*

[b] *Departamento de Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, Buenos Aires, Argentina.*

[c] *Instituto de Matemática, NCE and COPPE, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brasil.*

Abstract

A circular-arc graph is the intersection graph of arcs on a circle. A Helly circular-arc graph is a circular-arc graph admitting a model whose arcs satisfy the Helly property. A clique-independent set of a graph is a set of pairwise disjoint cliques of the graph. It is NP-hard to compute the maximum cardinality of a clique-independent set for a general graph. In the present paper, we propose polynomial time algorithms for finding the maximum cardinality and weight of a clique-independent set of a $\overline{3K_2}$-free CA graph. Also, we apply the algorithms to the special case of an HCA graph. The complexity of the proposed algorithm for the cardinality problem in HCA graphs is $O(n)$. This represents an improvement over the existing algorithm by Guruswami and Pandu Rangan, whose complexity is $O(n^2)$. These algorithms suppose that an HCA model of the graph is given.

Key words: algorithms, circular-arc graphs, clique-independent sets, Helly circular-arc graphs

Email addresses: gduran@dii.uchile.cl (Guillermo Durán), oscarlin@dc.uba.ar (Min Chih Lin), smera@dc.uba.ar (Sergio Mera), jayme@nce.ufrj.br (Jayme Luiz Szwarcfiter).

[1] Partially supported by FONDECyT Grant 1050747 and Millennium Science Nucleus "Complex Engineering Systems", Chile and CNPq under PROSUL project Proc. 490333/2004-4, Brazil.

[2] Partially supported by UBACyT Grants X184 and X212, Argentina and CNPq under PROSUL project Proc. 490333/2004-4, Brazil.

[3] Partially supported by the Conselho Nacional de Desenvolvimento Científico e Tecnológico, CNPq, Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro, FAPERJ, and CNPq under PROSUL project Proc. 490333/2004-4, Brasil.

1 Introduction

The aim of this work is to describe algorithms for finding the clique-independence number for certain classes of graphs. We consider both the cardinality and weighted versions of the problem. The classes of graphs here considered are subclasses of circular-arc graphs: Helly circular-arc graphs and $\overline{3K_2}$-free circular-arc graphs.

Helly circular-arc graphs form an important class of circular-arc graphs. Some properties of interval graphs are captured more closely by Helly circular-arc graphs than by other classes of circular-arc graphs. On the other hand, the class of $\overline{3K_2}$-free circular-arc graphs contains that of Helly circular-arc graphs. Furthermore, the cliques of the $\overline{3K_2}$-free circular-arc graphs preserve some of the properties of those of the Helly circular-arc graphs.

The first NP-hardness result for clique-independent sets appears in [7]. The following are some classes of graphs admitting polynomial time algorithms for the problems of determining a maximum clique-independent set: strongly chordal graphs [5], [14]; chordal graphs with bounded clique size [14]; dually chordal graphs [4]; comparability graphs [1]; balanced graphs [3], [6]; distance hereditary graphs [16]; short-chorded graphs with no 3-fans nor 4-wheels [10]; Helly circular-arc graphs [14].

Let G be an undirected connected graph, $V(G)$ and $E(G)$ its vertex and edge sets, respectively, $|V(G)| = n$ and $|E(G)| = m$. For $v \in V(G)$, denote by $N(v)$ the set of neighbours of v, and $N[v] = N(v) \cup \{v\}$. Say that v is *universal* when $N[v] = V(G)$. A *complete set* of G is a set of pairwise adjacent vertices. A *clique* is a maximal complete set. An *independent set* is a set of pairwise non adjacent vertices. A *clique-independent set* is a set of pairwise disjoint cliques.

The *clique graph* $K(G)$ of G is the intersection graph of the cliques of G. Let $M_1, \ldots, M_k$ and $v_1, \ldots, v_n$ be the cliques and vertices of a graph G, respectively. We define A_G, a clique matrix of G, as a 0-1 matrix whose entry (i, j) is 1 if $v_j \in M_i$, and 0 otherwise.

A *circular-arc (CA) model* for G is a pair $(C, \mathcal{A})$, where C is a circle and $\mathcal{A}$ is a collection of arcs of C, such that each arc $A_i \in \mathcal{A}$ corresponds to a vertex $v_i \in V(G)$, and A_i, A_j intersect precisely when v_i, v_j are adjacent, $i \neq j$. A *circular-arc (CA) graph* is one admitting a CA model. When traversing the circle C, we will always choose the clockwise direction. If s, t are points of C, write (s, t) to mean the arc of C defined by traversing the circle from s to t. Call s, t the *extremes* of (s, t), while s is the *start* and t the *end* of the arc. For $A_i \in \mathcal{A}$, write $A_i = (s_i, t_i)$. Without loss of generality, all arcs of C are considered as open arcs, no two extremes of distinct arcs of $\mathcal{A}$ coincide and no single arc entirely covers C.

Circular-arc graphs have a variety of applications in a lot of fields (see for example [12]). The first characterization of circular-arc graphs is due to Tucker [20], who also gave an $O(n^3)$ algorithm for their recognition [21]. Recently, McConnell [17] improved this to $O(n+m)$. Different subclasses of circular-arc graphs have been studied in the literature, such as Helly circular-arc graphs, proper circular-arc graphs and unit circular-arc graphs.

A *Helly circular-arc (HCA) graph* G is a CA graph admitting a CA model whose arcs satisfy the Helly property. That is, every pairwise intersecting subfamily of arcs of $\mathcal{A}$ contains a common point. Such a model is called a *Helly circular-arc (HCA) model* for G. Gavril [11] has characterized HCA graphs as exactly those admitting a clique matrix having the circular $1'$s property for columns. This characterization leads to an algorithm for recognizing HCA graphs, which builds an HCA model in $O(n^3)$ time if that model exists.

We employ the following notation. Let G be a graph:

- $\alpha_c(G)$, maximum cardinality of a clique-independent set of G, *the clique-independence number*
- $\tilde{\alpha_c}(G)$, maximum weight of a clique-independent set of G
- $\alpha(G)$, maximum cardinality of an independent set of G, *the independence number*
- $\tilde{\alpha}(G)$, maximum weight of an independent set of G

In the present paper, we propose algorithms for solving the following problems. For Helly circular-arc graphs G, we describe algorithms for determining $\alpha_c(G)$ and $\tilde{\alpha_c}(G)$. The algorithm for the cardinality version of this problem requires $O(n)$ time, while for the weighted version it requires $O(n^2)$. For $\overline{3K_2}$-free circular-arc graphs, the proposed algorithm determines $\alpha_c(G)$ in $O(m)$ time and $\tilde{\alpha_c}(G)$ in $O(m \log\log n + n^2)$. Besides clique-independent sets, the method also leads to the description of an algorithm for constructing the clique graph of a Helly circular-arc graph. The proposed algorithm runs in $O(n^2)$ time. Clique graphs of Helly circular-arc graphs were considered in [8,2].

An algorithm for solving the cardinality problem on Helly circular-arc graphs has been previously described by Guruswami and Pandu Rangan [14]. The complexity of this algorithm is $O(n^2)$. Algorithms for the remaining above problems have not been reported so far, to our knowledge.

As usual for many algorithms on circular-arc graphs, we assume that the graph is given by its circular-arc model, with the extremes of the arcs circularly sorted. If they are not sorted we would need to add an extra $O(n \log n)$ time for the sorting. The Helly circular-arc graphs are assumed to be represented by a Helly model. All weights here considered are non negative real values.

Let G be a graph admitting a CA model $(C, \mathcal{A})$. For $A \in \mathcal{A}$, denote by $V(A)$

the vertex of G corresponding to A. Similarly, for $\mathcal{A}' \subseteq \mathcal{A}$, $V(\mathcal{A}') = \{V(A)|A \in \mathcal{A}'\}$. If $V(A)$ is a universal vertex then A is a *universal arc*. If an arc $A \in \mathcal{A}$ contains some point $p \in C$ then say that A is an arc of p. Denote by $\mathcal{A}(p)$ the collection of arcs of p. Clearly, $V(\mathcal{A}(p))$ is a complete set of G. For $p, p' \in C$ say that p *(properly) dominates* p' when $\mathcal{A}(p)$ (properly) contains $\mathcal{A}(p')$. When $\mathcal{A}(p) = \mathcal{A}(p')$ then p, p' are *equivalent*. Say that $p \in C$ is a *complete point* when no point of C properly dominates p. In addition when $V(\mathcal{A}(p))$ is a clique of G then p is a *clique point* of C. Such a clique is called a *Helly clique*. Clearly, G might contain cliques that are not Helly. However, if $(C, \mathcal{A})$ is a Helly model then all its cliques are Helly. In this case, there is a one-to-one correspondence between cliques of G and non equivalent clique points of C. On the other hand, any non Helly clique contains at least three vertices. Furthermore, among the arcs of $\mathcal{A}$ corresponding to the vertices of a non Helly clique there exist always three of them which together cover the entire circle.

The plan of the paper is as follows. In Section 2, we describe methods for determining sets of complete and clique points, which are employed in the proposed algorithms. Algorithms for clique-independence problems are considered in Section 3. Further remarks form the last section. Preliminary results of this work appear published in [9].

2 Intersection Segments

In this section, we describe a method for finding sets of complete points of a CA graph. These sets will be employed in the algorithms proposed in the next section. The following concepts are central for our methods.

Let G be a graph admitting a CA model $(C, \mathcal{A})$. Denote $\mathcal{A} = \{A_1, \ldots, A_n\}$ and $A_i = (s_i, t_i)$, $1 \leq i \leq n$. A *segment* is an arc of C formed by two consecutive extremes of the arcs of $\mathcal{A}$, when traversing C. Clearly, there are $2n$ segments, which exactly cover C, except for their extreme points. Also, each arc of $\mathcal{A}$ corresponds to a sequence of consecutive segments. All points belonging to a same segment are equivalent. An *intersection segment* is a segment of the type (s_i, t_j), that is, its start point is the start point of some arc arc $A_i \in \mathcal{A}$, while its end point is the end point of an arc $A_j \in \mathcal{A}$. Write $I_i = (s_i, t_j)$. A point $p_i \in I_i$ is called an *intersection point*. There are at most n intersection segments. The following theorem relates complete points to intersection segments.

Theorem 1 *: Every complete point is an intersection point.*

PROOF. Let G be a graph having a model $(C, \mathcal{A})$ and p a complete point of C. Denote $I = (x, y)$ as the segment of C which p belongs to. We show that

I must be an intersection segment.

Case 1: $x = t_i$, for some i
Clearly $\mathcal{A}(p)$ is formed by the arcs of $\mathcal{A}$ which contain I. Since $x = t_i$, no arc of $\mathcal{A}$ starts at x. Consequently, all arcs of $\mathcal{A}(p)$ also contain the segment I', which immediately precedes I in C. Let $p' \in I'$. Then $\mathcal{A}(p') \supseteq \mathcal{A}(p)$. Furthermore, $A_i \in \mathcal{A}(p') \setminus \mathcal{A}(p)$. That is, the inclusion is proper, meaning that p is not a complete point.

Case 2: $x = s_i$ and $y = s_j$, for some $i \neq j$
The situation is similar to Case 1, except that I' becomes the segment that immediately succeeds I in C.

Case 3: $x = s_i$ and $y = t_j$
Since neither Case 1 nor 2 can occur then Case 3 must apply.

Consequently, $I = (s_i, t_j)$, meaning that I is an intersection segment, i.e., p is indeed an intersection point. $\triangle$

The converse of the above theorem does not necessarily hold. In order to relate intersection points to complete points, we employ the following additional notation. An intersection segment $I_i = (s_i, t_j)$ is *simple* when $A_i \cup A_j \neq C$, and *universal* otherwise. That is, I_i is universal when A_i and A_j cover the entire circle. A point belonging to a simple segment is a *simple point*, whereas one inside a universal segment is a *universal point*.

Theorem 2 *Let $I_i = (s_i, t_j)$ be an intersection segment and $p \in I_i$. Then*

(2.1) If p is simple then p is complete.
(2.2) If p is universal then p is either complete or it is properly dominated by some intersection point $p' \in (s_j, t_i)$.

PROOF. We know that $A_i, A_j \in \mathcal{A}(p)$, because $I_i = (s_i, t_j)$ and $p \in I_i$. Suppose that (2.1) is not true. Then there exists a point $p' \in C$, such that $\mathcal{A}(p')$ properly contains $\mathcal{A}(p)$. Clearly, $p' \notin I_i$. If $i = j$, then $I_i = A_i$, implying that $A_i \in \mathcal{A}(p) \setminus \mathcal{A}(p')$, contradicting $\mathcal{A}(p') \supseteq \mathcal{A}(p)$. Consequently, $i \neq j$. The latter implies $A_i, A_j \in \mathcal{A}(p) \cap \mathcal{A}(p')$. In this situation, $A_i \cup A_j = C$, contradicting I_i to be simple. Then p is indeed a complete point and (2.1) holds.

In the sequel, we prove (2.2). By hypothesis, p is universal. If p is not complete then there exists $p' \in C$ which properly dominates p. Consequently, $A_i, A_j \in \mathcal{A}(p) \cap \mathcal{A}(p')$. Since $p' \notin I_i$, the latter implies $p' \in (s_j, t_i)$. We can choose p' to be maximal, i.e. complete. By Theorem 1, p' is an intersection point. $\triangle$

Next, we consider some special subsets of points of C which are of interest. Define the following four subsets. A *complete (simple) (universal) (clique) point representation of* C is a maximal set of complete (simple) (universal) (clique) non equivalent points of C. Represent these sets by P, S, U, Q, respectively. We describe how to construct them.

Let $P', P'' \subseteq C$ be two subsets of points of C. Then P', P'' are *isomorphic* when there exists a bijection f between these sets such that p' and $f(p')$ are equivalent, for all $p' \in P'$. Clearly, any two complete (simple) (universal) (clique) point representations are isomorphic. That is, P, S, U, Q are all unique, up to isomorphism. Consequently, we can write $P = S \cup U'$, for some $U' \subseteq U$. Also, $Q \subseteq P$, with $Q = P$ precisely when $(C, \mathcal{A})$ is a Helly model. Clearly, Q corresponds to the set of Helly cliques of G. Moreover, the Helly cliques can be further bipartitioned, as follows. Let M_i be a Helly clique of G and p_i the clique point of Q corresponding to M_i. Then M_i is a *simple clique* or *universal clique*, according whether p_i is a simple or universal point, respectively.

Theorems 1 and 2 lead to an algorithm for constructing a complete point representation P of C, given a CA model $(C, \mathcal{A})$ for a graph G. In fact, the algorithm constructs explicitly the simple point representation S and then finds $U' \subseteq U$, such that $P = S \cup U'$. The algorithm is divided into two steps. Step 1 constructs S and a set $U'' \supseteq U$, which contains U and possibly some additional equivalent points. Step 2 determines U' by including in it one universal point (the one with lowest index), for each collection of equivalent complete points. The algorithm employs a list L to contain this collection.

Algorithm 1 *CONSTRUCTING A COMPLETE POINT REPRESENTATION OF A CA GRAPH*

STEP 1: Identify the segments of C. Define $S = U'' = \emptyset$. For each segment (x, y) of C, if x is the start of some arc $A_i \in \mathcal{A}$ and y the end of $A_j \in \mathcal{A}$ then let p_i be a point of (x, y) and perform the following additional test: if $A_i \cup A_j \neq C$, include p_i in S, otherwise include p_i in U''.

STEP 2: Define $U' = \emptyset$. For each universal point $p_i \in U''$, let $I_i = (s_i, t_j)$ be its corresponding universal segment. For each $p_i \in U''$, apply the following procedure. Compute $\mathcal{A}(p_i)$. Define $L = \{i\}$. Traverse the arc $(s_j, t_i) \subseteq C$, segment by segment, in the order as they appear. In case of an intersection segment $(s_k, t_l) \subseteq (s_j, t_i)$, choose a point $p_k \in (s_k, t_l)$, compute $\mathcal{A}(p_k)$, and if $\mathcal{A}(p_i) = \mathcal{A}(p_k)$ then include k in L. After all segments contained in (s_j, t_i) have been traversed then include p_r in U' precisely in the case where p_i is not properly dominated by any p_k, and $r = min\{k \in L\}$. At the end, $P = S \cup U'$.

It is simple to verify that the algorithm is correct. Let I_i be a simple segment of C. By (2.1) of Theorem 2, $p_i \in I_i$ must be a complete point and by Theorem 1, I_i is an intersection segment. Furthermore, no two simple points in distinct

intersection segments can be equivalent. Because, if $I_i = (s_i, t_j)$ and I_k, $i \neq k$, are two simple segments, while $p_i \in I_i$ and $p_k \in I_k$ are equivalent simple points then A_i and A_j must cover the entire circle, contradicting I_i and I_k to be simple. Consequently, Step 1 correctly constructs S and therefore U''. For Step 2, assume that (s_i, t_j) is a universal segment, and $p_i \in (s_i, t_j)$. By (2.2) of Theorem 2, p_i is complete or there exists some $p_k \in (s_j, t_i)$ which properly dominates p_i. In the latter situation, $p_i \notin U'$, which corresponds exactly to the action taken by the algorithm. When p_i is complete, U' must contain exactly one among the set of (complete) points equivalent to p_i. By Theorem 1, all such complete points must be intersection points. Step 2 chooses the one with the lowest index, and correctly constructs U'.

Now, we determine the complexity of the algorithm. In Step 1, the operations of deciding if a segment is an intersection segment and partitioning the intersection segments into simple and universal are all straightforward and can be answered in constant time, per segment. Consequently, Step 1 requires $O(n)$ time. In Step 2, we analyse each iteration i. It takes $O(n)$ time to construct $\mathcal{A}(p_i)$ and $\mathcal{A}(p_k)$, for the first intersection segment $I_k \subseteq (s_j, t_i)$ reached in the traversal of (s_j, t_i). For the remaining ones, consider two consecutive segments (not necessarily intersection segments) $I_k, I_{k'}$ inside (s_j, t_i) and let p be the end point of I_k. Then, if $p = s_b$ for some arc $A_b \in \mathcal{A}$, construct $\mathcal{A}(p_{k'})$ from $\mathcal{A}(p_k)$, just by including in $\mathcal{A}(p_k)$ the arc A_b; otherwise, $p = t_b$, and obtain $\mathcal{A}(p_{k'})$ by removing A_b from $\mathcal{A}(p_k)$. Since each arc is manipulated at most twice in the entire traversal, we conclude that the construction of all $\mathcal{A}(p_k)$ can be done in $O(n)$ time. Checking the containments conditions and the minimality also require $O(n)$ time. Consequently, the complexity of Step 2 is $O(n^2)$. Therefore Algorithm 1 constructs S and U'' in $O(n)$ time and $U' \subseteq U$ in $O(n^2)$ time. Consequently, we require $O(n^2)$ time for constructing P.

For determining U, possibly we need to eliminate equivalent points from the subset U' constructed in Step 1. It can be easily performed in overall $O(n^2)$ time.

Finally, consider the determination of the clique point representation Q of C. To obtain $Q \subseteq P$, we need to remove from P those points $p \in P$, such that $V(\mathcal{A}(p))$ is not a clique. With this purpose, apply the following algorithm.

Algorithm 2 *CONSTRUCTING A CLIQUE POINT REPRESENTATION OF A CA GRAPH*

Define $Q := P$. For each complete point $p \in P$, perform the following operations. Denote by (s_i, t_j) the intersection segment corresponding to p. Define $s_o := s_j$. Traverse the arc (t_j, t_i), identifying the extreme points q of the arcs $A_k \in \mathcal{A}$, such that q is the first extreme of A_k, in the traversal. For each such extreme q, do the following: if $q = s_k$ and $t_k \in (s_0, s_i)$ then $Q := Q \setminus \{p\}$ and

terminate the iteration corresponding to p (p not a clique point*); if $q = t_k$ and $s_k \in (s_0, s_i)$ then assign $s_0 := s_k$. At the end, Q is the required clique point representation.*

Theorem 3 *The set Q constructed by Algorithm 2 is a clique point representation.*

PROOF. Let $p \in P$ be a complete point of C. We prove that Algorithm 2 excludes p from Q precisely when p is not a clique point of C. Let (s_i, t_j) be the intersection segment corresponding to p. Label as *white* the arcs of $\mathcal{A}$ containing (s_i, t_j), and as *black* the remaining arcs. Observe that p is not a clique point if and only if there exists a black arc intersecting all white ones. Furthermore, in a traversal of C, starting from t_j, and arc (s_k, t_k) is white if t_k precedes s_k in the traversal, and black otherwise.

Suppose p is a clique point. Then for any black arc (s_k, t_k), where $s_k \in (t_j, t_i)$, there is a white arc (s_h, t_h) not intersecting A_k. Consequently, $t_h \in (t_j, s_k)$ and $s_h \in (t_k, s_i)$, meaning that t_h precedes s_k in the traversal. When the algorithm considers s_k, the extreme s_0 defined by the algorithm belongs to $[s_h, s_i)$. Consequently, $t_k \notin (s_0, s_i)$, implying that the algorithm would not exclude p from Q during the computation of s_k. Therefore, after considering the last extreme of the arc (t_j, t_i), p is still in Q.

Alternatively, suppose p is not a clique point. Then there exists a black arc A_k intersecting all white ones. Clearly, $s_k \in (t_j, t_i)$. When the algorithm considers the extreme s_k, it follows that $t_k \in (s_0, s_i)$, otherwise there exists a white arc (s_h, t_h) satisfying $s_0 = s_h$, and such an arc does not intersect A_k, a contradiction. Therefore, the algorithm terminates the iteration corresponding to p and excludes it from Q. $\triangle$

To determine the complexity of the algorithm, observe that for each complete point $p \in P$, the number of steps performed is at most the number of extremes contained in the arc (t_j, t_i), which is less than twice the degree of the vertex of the graph corresponding to arc A_i. Consequently, it takes $O(n)$ time to decide if a complete point $p \in P$ is a clique point. Furthermore, given P the algorithm constructs Q in $O(m)$ time.

3 Algorithms for clique-independent sets

In this section, we describe algorithms for finding the maximum cardinality and weight of a clique-independent set of a $\overline{3K_2}$-free CA graph. Also, we apply

the algorithms to the special case of an HCA graph. First, we introduce some additional notation.

Let G be a graph admitting a CA model $(C, \mathcal{A})$. Let $I_i = (s_i, t_j)$ be an intersection segment and $p_i \in I_i$. The *arc reduction of* p_i is the arc $R_i = (s_i, t_a) \subseteq C$, where $(p_i, t_a) \supseteq (p_i, t_l)$, for all $A_l \in \mathcal{A}(p_i)$. That is, for $I_i = (s_i, t_j)$, select the arc $A_a = (s_a, t_a) \in \mathcal{A}(p_i)$, such that t_a is as far as possible from t_j, when traversing C. Then $R_i = (s_i, t_a) \subseteq A_a$. Let Q be the clique point representation of C, and $Q' \subseteq Q$. Denote by $G_R(Q')$ the intersection graph of the arc reductions of the clique points of Q'. On the other hand, denote by $\hat{K}(Q')$ the intersection graph of the family of subsets of arcs $\{\mathcal{A}(p_i) | p_i \in Q'\}$. Clearly, $\hat{K}(Q')$ is an induced subgraph of the clique graph $K(G)$ of G. The following theorem relates $G_R(Q')$ with $\hat{K}(Q')$.

Theorem 4 *Let G be a graph admitting a CA model $(C, \mathcal{A})$, Q the clique point representation of C and $Q' \subseteq Q$. Then $G_R(Q') \cong \hat{K}(Q')$.*

PROOF. Let $p_i, p_k \in Q'$ be non equivalent clique points of C, with $I_i = (s_i, t_j)$, $I_k = (s_k, t_l)$ the intersection segments containing p_i, p_k, and R_i, R_k their corresponding arc reductions, respectively. We show that $R_i \cap R_k \neq \emptyset$ if and only if $\mathcal{A}(p_i) \cap \mathcal{A}(p_k) \neq \emptyset$, implying the theorem.

Suppose that $\mathcal{A}(p_i) \cap \mathcal{A}(p_k) \neq \emptyset$. Examine the arc reductions R_i, R_k and compare them to an arc $A_c \in \mathcal{A}(p_i) \cap \mathcal{A}(p_k)$. Clearly, in C the extremes of I_i and I_k appear in the circular ordering s_i, t_j, s_k, t_l, s_i. Include the extremes of A_c in this ordering. Because $A_c \in \mathcal{A}(p_i) \cap \mathcal{A}(p_k)$, there are two possibilities: $s_i, t_j, t_c, s_c, s_k, t_l, s_i$ or $s_i, t_j, s_k, t_l, t_c, s_c, s_i$. In the first situation, $(s_i, t_c) \subseteq R_i \cap R_k$ and in the second $(s_k, t_c) \subseteq R_i \cap R_k$. Consequently, $R_i \cap R_k \neq \emptyset$, as required.

Conversely, suppose $R_i \cap R_k \neq \emptyset$. Let $R_i = (s_i, t_a)$ and $R_k = (s_k, t_b)$. Since $R_i \cap R_k \neq \emptyset$, there are again two possibilities. Either $I_i \subseteq R_i \cap R_k$ or $I_k \subseteq R_i \cap R_k$. In the former alternative, $A_b \in \mathcal{A}(p_i) \cap \mathcal{A}(p_k)$, while in the latter $A_a \in \mathcal{A}(p_i) \cap \mathcal{A}(p_k)$. Consequently, $\mathcal{A}(p_i) \cap \mathcal{A}(p_k) \neq \emptyset$, completing the proof. $\triangle$

We observe that all the arc reductions can be easily computed, as follows. First, remove all arcs of $\mathcal{A}$ which are properly contained in some other arc of $\mathcal{A}$. Afterwards, choose an arbitrary arc $A_k \in \mathcal{A}$ and traverse C starting from s_k. In the traversal, each time we meet an intersection segment $I_i = (s_i, t_j)$, the arc reduction R_i is precisely the arc (s_i, t_a), where A_a is the last arc of $\mathcal{A}$ which started before I_i. Clearly, the above procedure can be implemented in $O(n)$ time.

Let $\mathcal{M}'$ be the set of cliques of G corresponding to $Q' \subseteq Q$, and let $K(G)$

be the clique graph of G, that is, the intersection graph of the cliques of G. We remark that $G_R(Q')$ is also isomorphic to the induced subgraph of $K(G)$ corresponding to the cliques of $\mathcal{M}'$. This observation together with Theorem 4 imply a simple algorithm for constructing the clique graph of an HCA graph G, given by its HCA model $(C, \mathcal{A})$, as follows. First, we find Q and then all arc reductions $R_i \subseteq C$. Let $\mathcal{R} = \cup\{R_i\}$. It is clear that $(C, \mathcal{R})$ is a CA model of the clique graph of G, after adjusting the extremes of the arcs of $\mathcal{R}$, which may have possibly coincided. As for the complexity, we need $O(n^2)$ time for the construction of Q, while all the arc reductions require $O(n)$ time. Therefore the construction of $K(G)$, the clique graph of G, takes $O(n^2)$ time.

There is a straightforward relation between clique-independent sets of a graph G and independent sets of its clique graph.

Theorem 5 *Let G be a CA graph, Q its clique point representation, $Q' \subseteq Q$, $V' \subseteq V(\hat{K}(Q'))$ and $\mathcal{M}'$ the set of cliques of G corresponding to those clique points of Q' associated to the vertices of V' in $\hat{K}(Q')$. Then $\mathcal{M}'$ is a clique-independent set of G if and only if V' is an independent set of $\hat{K}(Q')$.*

For the purpose of the problem of clique-independence in CA graphs, it would be useful to know whether two cliques could possibly be disjoint. The next theorem describes conditions which would force the cliques to intersect.

Theorem 6 *Let G be a graph with a CA model $(C, \mathcal{A})$, and M, M' cliques of it. If M, M' satisfy any of the conditions below then $M \cap M' \neq \emptyset$.*

(6.1) M is Helly and M' is universal
(6.2) M is non Helly and M' is Helly
(6.3) M and M' are both non Helly and G is $\overline{3K_2}$-free

PROOF. (6.1): Since M' is universal, it contains two vertices whose corresponding arcs in $\mathcal{A}$ cover the entire circle. Then any Helly clique contains at least one of these vertices. Consequently, $M \cap M' \neq \emptyset$.

(6.2): Because M' is a Helly clique, there is a clique point $p' \in C$ representing it. Since M is non Helly, it contains three vertices whose corresponding arcs cover the entire circle. Consequently, one of these arcs must contain p'. That is, $M \cap M' \neq \emptyset$.

(6.3): By hypothesis, G is $\overline{3K_2}$-free and M, M' are non Helly cliques. Suppose $M \cap M' = \emptyset$. No arc associated to some vertex $v \in M$ can contain an arc of M'. Otherwise $v \in M'$, contradicting $M \cap M' = \emptyset$. Since M is non Helly, there is a subset $\{A_1, A_2, A_3\}$ of three arcs associated to M, covering the entire circle. For each of these arcs A_i, there is a corresponding arc A'_i of M' satisfying $A_i \cap A'_i = \emptyset$, otherwise the vertex v_i corresponding to A_i would be

part of M', a contradiction. Examine A'_1, with respect to A_2 and A_3. Because $A'_1 \cap A_1 = \emptyset$, A'_1 must intersect A_2 or A_3. In addition, A'_1 can not be contained in A_2, nor A_3. Hence $A'_1 \cap A_2 \neq \emptyset$ and $A'_1 \cap A_3 \neq \emptyset$. Similarly, there exists A'_2, A'_3 corresponding to vertices of M', such that $A'_2 \cap A_2 = A'_3 \cap A_3 = \emptyset$, while $A'_2 \cap A_1, A'_2 \cap A_3, A'_3 \cap A_1, A'_3 \cap A_2 \neq \emptyset$. In addition, A'_1, A'_2, A'_3 pairwise intersect, because M' is a clique. Therefore the subgraph of G induced by the vertices corresponding to the arcs $A_1, A_2, A_3, A'_1, A'_2, A'_3$ is a $\overline{3K_2}$, contradicting the hypothesis. Hence $M \cap M' \neq \emptyset$. $\triangle$

The above results lead to the following equation for computing the clique-independence number of a $\overline{3K_2}$-free CA graph.

Theorem 7 *Let G be a $\overline{3K_2}$-free CA graph, S and Q the simple and clique point representations, relative to some CA model of G, respectively. Then*

$$\alpha_c(G) = \begin{cases} 1, & \text{if } S \cap Q = \emptyset \\ \alpha(G_R(S \cap Q)), & \text{otherwise} \end{cases}$$

PROOF. Assume G contains two disjoint cliques M, M'. By applying the conditions (6.3),(6.2) and (6.1), we conclude that M and M' must be simple cliques. If $S \cap Q = \emptyset$ there are no simple cliques and therefore no disjoint cliques can exist. Consequently, $\alpha_c(G) = 1$. Consider $S \cap Q \neq \emptyset$. By Theorem 6, $\alpha_c(G)$ equals the cardinality of a maximum set of pairwise disjoint simple cliques. By Theorem 5, $\alpha_c(G) = \alpha(\hat{K}(S \cap Q))$ and by Theorem 4, $\alpha(\hat{K}(S \cap Q)) = \alpha(G_R(S \cap Q))$. $\triangle$

The proposed algorithm for determining $\alpha_c(G)$ corresponds to the computation of the equation given by Theorem 7. Let G be a $\overline{3K_2}$-free graph with a given CA model.

Algorithm 3 *CLIQUE-INDEPENDENCE NUMBER OF A $\overline{3K_2}$-FREE CA GRAPH*

First, construct the simple representation S. Then we can generate $S \cap Q$ applying Algorithm 2 to $S \subseteq P$. If $S \cap Q = \emptyset$ then $\alpha_c(G) = 1$. Otherwise, find all arc reductions and construct a CA model for the graph $G_R(S \cap Q)$. Finally, find the maximum independent set of $G_R(S \cap Q)$, as $\alpha_c(G) = \alpha(G_R(S \cap Q))$.

Next, we determine the complexity of the algorithm. Finding S requires $O(n)$ time by Algorithm 1, but the determination of $S \cap Q$ (by Algorithm 2) takes $O(m)$ time. The construction of all arc reductions and finding the CA model

of $G_R(S \cap Q)$ can be done in $O(n)$ time. Finally, the maximum independent set of $G_R(S \cap Q)$ can also be computed in $O(n)$ time, using the algorithms given in [13] or [15]. Therefore the overall complexity is $O(m)$.

In particular, if the given model is an HCA model then $S \cap Q = S$ and S can be found $O(n)$ time, then the complexity of the algorithm reduces to $O(n)$.

The solution of the weighted clique-independence problem is similar. Let G be a $\overline{3K_2}$-free CA graph, where there is a weight assigned to each vertex. For a clique M of G, define its weight $\tilde{M}$ as the sum of the weights of the vertices which form M. The equation below determines the maximum weight of a clique-independent set of G.

Theorem 8 *Let G be a $\overline{3K_2}$-free CA graph, S and Q the simple and clique point representations, relative to some CA model of G, respectively. Denote by $\tilde{M}_1$ the maximum weight of a single clique of G. Then*

$$\tilde{\alpha_c}(G) = \begin{cases} \tilde{M}_1, & \text{if } S \cap Q = \emptyset \\ max\{\tilde{M}_1, \tilde{\alpha}(G_R(S \cap Q))\}, & \text{otherwise} \end{cases}$$

PROOF. It is analogous to that one of Theorem 7. The only remark we have to do is that in this case, by Theorem 6, $\tilde{\alpha_C}(G)$ equals the maximum between the maximum weight of a set of pairwise disjoint simple cliques and the maximum weight of a single clique (which could be a non simple clique).$_\triangle$

The algorithm for computing the above formula is as follows. Let G be a $\overline{3K_2}$-free graph with a given CA model.

Algorithm 4 *MAXIMUM WEIGHT OF A CLIQUE-INDEPENDENT SET OF A $\overline{3K_2}$-FREE CA GRAPH*

Select the clique M_1 of G having maximum weight $\tilde{M}_1$. Construct S and $S \cap Q$. If $S \cap Q = \emptyset$ then $\tilde{\alpha_c}(G) = \tilde{M}_1$. Otherwise, construct the CA model of $G_R(S \cap Q)$ and $\tilde{\alpha_c}(G) = max\{\tilde{M}_1, \tilde{\alpha}(G_R(S \cap Q))\}$.

Finally, we evaluate the complexity of the algorithm. The maximum weight clique problem on circular-arc graphs can be solved in $O(n \log n + m \log \log n)$ time [18]. The weighted independent set of a CA graph can be determined in $O(n^2)$ time [19]. The remaining operations require $O(m)$ time. The overall complexity is therefore $O(m \log \log n + n^2)$.

When G admits an HCA model, G has at most n cliques. Furthermore, the maximum weight $\tilde{M}_1$ among all cliques can be determined in linear time (using

the same idea of Step 2 of Algorithm 1) , as $\tilde{M}_1$ equals to the maximum weight among the sets of arcs corresponding to every intersection segment of G. The dominating operation is that of determining $\tilde{\alpha}(G_R(S \cap Q))$. Consequently, the complexity of the algorithm is $O(n^2)$.

4 Conclusions

The table below summarizes the problems that have been considered in this paper, together with the complexities of the corresponding proposed algorithms.

Problem	Graph Class	Version	Proposed alg.	Prev. alg.
Clique-independence number	HCA	cardinality	$O(n)$	$O(n^2)$ [14]
		weighted	$O(n^2)$	-
	$\overline{3K_2}$-free CA	cardinality	$O(m)$	-
		weighted	$O(m \log\log n + n^2)$	-

In all cases, the algorithms determine the cardinality or the weight of the corresponding maximum clique-independent set. There is no difficulty to modify them so as to compute the actual maximum sets.

It remains open to determine whether the clique-independence number of a general CA graph can be found in polynomial time.

References

[1] V. Balachandhran, P. Nagavamsi, and C. Pandu Rangan. Clique transversal and clique independence on comparability graphs. *Information Processing Letters*, 58:181–184, 1996.

[2] F. Bonomo. Self-clique Helly circular-arc graphs. In *Proceedings of the 34th Argentine Conference on Informatics and Operational Research*, Rosario (Argentina), 2005. To appear.

[3] F. Bonomo, G. Durán, M. C. Lin, and J. L. Szwarcfiter. On balanced graphs. *Mathematical Programming B*, 2005. To appear.

[4] A. Brandstädt, V. D. Chepoi, F. F. Dragan, and V. I. Voloshin. Dually chordal graphs. *SIAM Journal on Discrete Mathematics*, 11:437–455, 1998.

[5] G. J. Chang, M. Farber, and Zs. Tuza. Algorithmic aspects of neighbourhood numbers. *SIAM Journal on Discrete Mathematics*, 6:24–29, 1993.

[6] E. Dahlhaus, P. D. Manuel, and M. Miller. Maximum h-colourable subgraph problem in balanced graphs. *Information Processing Letters*, 65:301–303, 1998.

[7] D. Duffus, H. A. Kiersted, and W. T. Trotter. Fibers and ordered set coloring. *Journal of Combinatorial Theory A*, 58:158–164, 1991.

[8] G. Durán and M. C. Lin. Clique graphs of Helly circular-arc graphs. *Ars Combinatoria*, 60:255–271, 2001.

[9] G. Durán, M. C. Lin, S. Mera, and J. L. Szwarcfiter. Clique-independent sets of Helly circular-arc graphs. *Electronic Notes in Discrete Mathematics*, 18:41–46, 2004.

[10] G. Durán, M. C. Lin, and J. L. Szwarcfiter. On clique transversals and clique independent sets. *Annals of Operations Research*, 116:71–77, 2002.

[11] F. Gavril. Algorithms on circular-arc graphs. *Networks*, 4:357–369, 1974.

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

[13] M. C. Golumbic and P. Hammer. Stability in circular arc-graphs. *Journal of Algorithms*, 9:314–320, 1988.

[14] V. Guruswami and C. Pandu Rangan. Algorithmic aspects of clique transversal and clique-independent sets. *Discrete Applied Mathematics*, 100:183–202, 2000.

[15] W.-L. Hsu and K.-H. Tsei. Linear time algorithms on circular-arc graphs. *Information Processing Letters*, 40:123–129, 1991.

[16] C.-M. Lee, M.-S. Chang, and S.-C. Sheu. The clique transversal and clique independence of distance hereditary graphs. In *Proceedings of the 19th Workshop on Combinatorial Mathematics and Computation Theory*, pages 64–69, Taiwan, 2002.

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

[18] W. K. Shih and W. L. Hsu. An $O(n \log n + m \log \log n)$ algorithm for finding a maximum weight clique in circular-arc graphs. *Infor. Process. Letters*, 32:129–134, 1989.

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

[20] A. Tucker. Characterizing circular-arc graphs. *Bull. Amer. Math. Soc.*, 76:1257–1260, 1970.

[21] A. Tucker. An efficient test for circular-arc graphs. *SIAM J. Comput.*, 9:1–24, 1980.