LIST HOMOMORPHISMS
AND BIPARTITE CO-CIRCULAR ARC GRAPHS

by

Ali Ershadi

B.Sc. (Computing Science), Sharif University of Technology, 2009

A Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science

in the
School of Computing Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Fall 2012

APPROVAL

Name: Ali Ershadi

Degree: Master of Science

Title of Thesis: List Homomorphisms and Bipartite Co-Circular Arc Graphs

Examining Committee:

Dr. Ramesh Krishnamurti
Chair

Dr. Pavol Hell _____
Senior Supervisor
Professor

Dr. Arthur L. Liestman _____
Supervisor
Professor

Dr. Luisa Gargano _____
Examiner
Professor
University of Salerno

Date Approved: July 31, 2012 _____

Declaration of
Partial Copyright Licence

Abstract

In the study of the list homomorphisms, the class of bipartite co-circular arc graphs plays an important role in delineating easy (polynomial) cases and hard (NP-complete) cases of the list homomorphism problem. This class of graphs has many equivalent characterizations, and we present (at the beginning of Chapter 4) a new short proof of some of these equivalences. We then discuss possible approaches to the recognition problem for this class of graphs. First we discuss the linear time recognition algorithms of circular arc graphs by Eschen-Spinrad, McConnell and Nussbaum-Kaplan. These may be applied to the complementary graph, but the algorithm is no longer linear. The main new contributions contained in this thesis are efficient algorithms for the recognition of co-circular arc graphs in the special cases of trees, k-trees and bounded degree graphs (see Chapter 3). We also present new efficient algorithms for colouring, matching, and other similar problems on this class of graphs (see Chapter 4).

Acknowledgments

It is a pleasure to thank the people who made this thesis possible. Foremost, I would like to express my gratitude to my supervisor **Dr. Pavol Hell**, who has supported me throughout this thesis with his guidance, patience and knowledge.

I would also like to thank my committee Dr. Arthur Liestman, Dr. Louisa Gorgano and Dr. Ramesh Krishnamurti for their insightful comments and questions.

I am grateful to my friends for their constant support and encouragement throughout my work, specially Chakaveh Ahmadizadeh, Mojtaba Arvin and Aida Miri.

Finally, I wish to thank my parents, for their unconditional moral and financial support.

Contents

List of Figures

Chapter 1

List Homomorphisms

In this chapter, we introduce the list homomorphism problem and discuss various aspects of it.

1.1 Introduction

We first present the essential definitions for the list homomorphism problem, and present some theorems and techniques which are useful for treating the problem.

A graph is one of the richest abstract data types that may be used as a model for a relational property between a set of objects. Let G denote a graph. The graph G is defined by $V(G)$ as the set of vertices, also called nodes, and $E(G)$ as the set of edges between vertices. We denote by n the number of vertices and the number of edges by m. An edge $e \in E(G)$ is denoted by uv when u and v are the two vertices of e. By a graph we mean an undirected graph. We call a directed graph, a *digraph*. In a digraph, the edge uv is directed from u to v. Thus the edges uv and vu are distinct in digraphs. For any vertex v in a graph G, the *neighbourhood of* v, denoted $N(v)$, is a set of vertices in $V(G)$ to which v is adjacent. Let $N[v]$ denote the *closed neighbourhood* of v, equivalently $N[v] = N(v) \cup \{v\}$. The size of the neighbourhood of v is called degree of v and denoted $deg(v)$. The graph G is called a *complete* graph if all possible edges are in $E(G)$. We denote by K_n a complete graph on n vertices . An edge uu is called a *loop*, regardless of whether graph is directed or not. Each vertex may have a loop. The graph G is a *irreflexive* graph if it contains no loop. If each vertex of a graph G has a loop then G is a *reflexive* graph. A graph H is called a subgraph of G if $V(H) \subset V(G)$ and $E(H) \subset E(G)$. If all possible edges of G appear in H then H is an *induced subgraph* of G. An induced subgraph H of G is called a *clique* if H is complete.

Similarly, if H does not contain any edge the H is called an *independent set*. We denote by $\overline{G}$ *the complement of a graph* G. The graph $\overline{G}$ has the same vertex set as G and two vertices are adjacent in $\overline{G}$ if and only if they are not adjacent in G. A clique in a graph G is an independent set in the $\overline{G}$, and vice versa.

Definition 1.1.1. A vertex is called a *universal vertex* if it is adjacent to all other vertices of the graph.

A *module* M in a graph G is a set of vertices of G, i.e., $M \subset V(G)$, such that any vertex $v \in M$ is adjacent to the same set of vertices in $V(G) \setminus M$ (outside of M). A module is also called a *homogeneous set*.

Definition 1.1.2. A module S is called a *clique module*, also called a *complete homogeneous set*, if S is a clique.

In other words, any pair of vertices v and u in a clique module S have the same closed neighbourhood, i.e., $N[v] = N[u]$. Any pair of vertices in a clique module is called a *twin pair*.

In a graph G, a *path of length* k is a sequence of vertices $v_1, v_2, .., v_k$, such that v_i and v_{i+1} are adjacent, for $1 \leq i < k$. A path is an *induced path* if v_i is not adjacent to v_j with $j \neq i \pm 1$. We let P_k denote a graph that is a path of length k. Note that P_k has k vertices and $k-1$ edges. In a graph G, a *cycle of length* k is a path of length k in which v_k is adjacent to v_1. We let C_k denote a cycle of length k. For any cycle C_k of length greater than or equal to four, an edge $e \in E(G)$ is called a *chord* if e incident to two non-consecutive vertices in C_k. Similarly, in a digraph H we denote by $\overrightarrow{P_k}$ an induced directed path, and denote by $\overrightarrow{C_k}$ an induced directed cycle of length k.

Definition 1.1.3. A *chordless cycle* in a graph H is an induced cycle of length at least 4.

Definition 1.1.4. A graph G is *chordal* if G has no chordless cycle. Similarly, G is *weakly chordal* if it has no chordless cycle of length greater than 4.

A *bipartite* graph G is a graph in which $V(G)$ can be partitioned into two parts X and Y such that X and Y are independent sets in G. Equivalently, a graph is bipartite if and only if it has no odd cycle [65]. We denote by K_{n_1,n_2} the complete bipartite graph with parts of size n_1 and n_2. (See definition 1.1.4.)

Definition 1.1.5. A *chordal bipartite* graph G is a bipartite graph which is weakly chordal.

The following is the formal definition of an interval graph.

Definition 1.1.6. A graph H is an *interval graph* if there exists a family I of intervals on the real line and a one-to-one correspondence of $V(H)$ with the intervals in I so that two vertices are adjacent if and only if their corresponding intervals intersect. The family I of intervals is called an *interval representation* of H. Without loss of generality, we may assume no two intervals share any endpoints.

Equivalently, a graph H is an interval graph, if and only if it admits an interval representation I. Each interval $I_i \in I$ is represented by its right endpoint r_i and left endpoint ℓ_i, i.e., $I_i = (\ell_i, r_i)$. Any interval representation can be presented by its ordering of the right and left endpoints. We will discuss interval graphs in Chapter 2.

Definition 1.1.7. A graph H is a *circular arc graph* if there exists a family R of circular arcs around a fixed circle and a one-to-one correspondence of $V(H)$ with the circular arcs in R so that two vertices are adjacent if and only if their corresponding circular arcs intersect. Without loss of generality, we may assume no two circular arcs share any endpoints.

It is clear that a circular arc representation in which no arc goes through a region on the circle is equivalent to an interval representation. Thus every interval graph is a circular arc graph, and the class of circular arc graphs is a generalization of the class of interval graphs. Let A_i denote a circular arc in R. If we traverse the circle in the clockwise direction then the left endpoint of A_i, denoted by ℓ_i, precedes the right endpoint of A_i, denoted by r_i. Any circular arc representation R corresponds to a circular ordering of the endpoints in R. We will discuss circular arc graphs in Chapter 2.

The following is a basic definition for our purposes.

Definition 1.1.8. [30] A *homomorphism* of a graph G to a graph H, written as $G \to H$, is a vertex mapping f, $f : V(G) \to V(H)$, such that if v and u are two vertices of G with $uv \in E(G)$, then $f(u)f(v) \in E(H)$.

A homomorphism between two graphs preserves adjacency. The term came from abstract algebraic point of view, where homomorphism is a structure-preserving function. Note that we do *not* restrict the homomorphism to be a surjective mapping, an injective mapping or representing G as a subgraph or induced subgraph of H. We will discuss different examples of graph homomorphism later on.

A more general case of graph homomorphism is defined below.

Definition 1.1.9. Let G and H be two graphs. Suppose L is a family of lists, $L(v)$ for each $v \in V(G)$. A *list homomorphism* of G to H with respect to the lists L, is a homomorphism $f : G \to H$ such that $f(v) \in L(v)$ for each $v \in V(G)$.

Thus list homomorphisms may be viewed as homomorphisms with additional constraints which limit the choice of each vertex assignment to a specific list.

Now that we have defined our basic concepts, we can describe the problem we are considering in the following.

Definition 1.1.10. Suppose H is a fixed graph. The *H-homomorphism problem*, HOM H, asks whether or not there exists a homomorphism $f : G \to H$, of a given graph G to H.

- **Given**: A Graph G
- **Question** : Is there a homomorphism $G \to H$?

The *homomorphism problem* asks for the existence of a homomorphism $G \to H$ between two given graphs G and H, both parts of the input. This homomorphism problem is NP-complete [27]. The reason is that there are many NP-complete problems that can be viewed as a restriction of the homomorphism problem, for example the so-called k-colouring problem.

Definition 1.1.11. Given a graph G, the *k-colouring problem* asks whether or not there exists a colouring of the vertices of G with k colours such that two vertices with the same colour are not adjacent.

The k-colouring problem is equivalent to the homomorphism problem where H is the complete graph on k vertices, K_k. Thus, homomorphisms are a generalization of graph colourings. In fact, a graph homomorphism $G \to H$ is also called an *H-colouring*. The following result classifies the complexity of the HOM H problem in general.

Theorem 1.1.1. *[27] HOM H is NP-complete, if H is not a bipartite graph and does not contain a loop, and is polynomial time solvable otherwise.*

We also define the list homomorphism problem.

Definition 1.1.12. Suppose H is a fixed graph. Given a graph G and lists $L(v) \subseteq V(H)$ for all $v \in V(G)$, the list H-homomorphism problem, *L-HOM H*, asks whether or not there exists a homomorphism f of G to H such that $f(v) \in L(v)$, for each $v \in V(G)$.

- **Given**: A Graph G with a family of lists L
- **Question** : Is there a list homomorphism $G \to H$ with respect to L?

By fixing the graph H, also called the *host graph*, we view the graph G, also called the *guest graph*, with lists L as the input. It is natural to restrict the host graphs to a class of graphs C. There are different ways to describe C - a structural definition, a geometrical definition, etc.- but anyhow, C rather has two important properties. First, the class of graph C can be recognized efficiently, i.e., in polynomial time, and second, the L-HOM H problem where the host graph is a graph from C, has a polynomial time algorithm for any guest graph, or it is NP-complete. In this context, we define *easy cases* (polynomial time solvable) and *hard cases* (NP-complete) of the L-HOM problem.

The homomorphism problem, HOMH, can be viewed as a restriction (with lists) of the list homomorphism problem L-HOMH. The following theorem shows the relation between two problems.

Theorem 1.1.2. *The homomorphism problem is polynomial time reducible to the list homomorphism problem. In other words, HOM H $\preceq_P$ L-HOM H.*

Proof. Given an instance of HOMH problem with an input graph G. Suppose we create an instance of L-HOMH such that for every vertex v in G, the list $L(v)$ is set to $V(H)$. In other words, v can be assigned to any vertex in $V(H)$. It is clear that if there exists a homomorphism $f : G \to H$ then f respects the lists L. □

Every instance of HOMH can be viewed as an instance of L-HOMH. This motivates us to focus on the list homomorphism problem.

The list homomorphism problem naturally models many real-life problems. Here we introduce an example application of list homomorphism problem in the *job assignment problem*.

Example. Suppose we have a set of jobs J and a set of machines M. We want to assign each job $j \in J$ to a specific machine $m \in M$ so that certain conditions hold. Each machine m has a list of jobs $L(m) \subset J$ such that m is capable of performing a job $j \in L(m)$ and a list of machines $N(m)$ which are adjacent to it. Also we have some communication requirements for the jobs. If a job $j \in J$ is assigned to a machine $m \in M$ then m should be capable of performing j. Also every pair of jobs that need communication, must be assigned to two machines with communication. If we define a graph G with J as $V(G)$ such that any pair of jobs that needs communication are adjacent in G, and a graph H with M as vertices and for each vertex m, $N(m)$ is the adjacency list of m. It is easy to see the desired assignment of jobs to machines is exactly a solution for an instance of the list homomorphism problem from G to H.

1.2 The Role of Min Ordering

In this section, we want to introduce a property of digraphs which turns out to be useful for solving the list homomorphism problem.

Definition 1.2.1. A *min ordering* of a digraph H is an ordering $<$ of the vertices of H, such that if $uv \in E(H)$ and $u'v' \in E(H)$ are two edges such that $u < u'$, $v' < v$, then there also is an edge $uv' \in E(H)$.

Equivalently, $<$ is a min ordering of H if and only if the existence of $uv \in E(H)$ and $u'v' \in E(H)$ implies the existence of $min(u, u')min(v, v') \in E(H)$. This explains the reason for the name. The *min ordering*, also called *X-underbar enumeration property* in [47] [27]. It was named $\underline{X}$, by implication of the underbar if we had two crossed edges as in Figure 1.1.

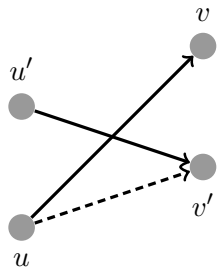


Figure 1.1: Minordering: The existence of uv and $u'v'$ entails uv'.

If H is a graph then the definition of min ordering can be applied ignoring the direction of edges in the definition 1.2.1.

For later purposes, we also specifically describe the min ordering property for reflexive and bipartite graphs. A reflexive graph H admits a min ordering $<$ if and only if for each edge $uv \in E(H)$ such that $u < v$, if a vertex w is placed between u and v in the min ordering (i.e., $u < w < v$), then the edge $uw \in E(H)$ also exists [20]. In other words, any vertex that appears between two adjacent vertices in the min ordering is adjacent to the minimum of the two. This turns out to mean that reflexive graphs with min ordering are exactly the interval graphs. We will talk about this in Section 1.3.1.

Let H be a fixed bipartite graph with parts X and Y. We shall call vertices in X *white* vertices, and vertices in Y *black* vertices. To validate the conditions of min ordering for a bipartite graph, we assume an imaginary direction for each edge, from white vertices to black vertices. A min ordering of H can be viewed as two orderings, one for each part of H: we order vertices in X and vertices in Y separately. If the edges x_1y_1 and x_2y_2 are in

$E(H)$ and $x_1 < x_2$ and $y_2 < y_1$ then the edge x_1y_2 is in $E(H)$. Note that since we ordered black and white vertices separately, we ignore the comparisons between two vertices with different colours.

Now we can explain the role of min ordering concerning the list homomorphism problem with the following theorem.

Theorem 1.2.1. *[24] If a digraph H admits a min ordering then L-HOMH is polynomial time solvable.*

Here we describe a polynomial time algorithm to solve L-HOMH using the min ordering.

Proof. Suppose H is a fixed digraph with min ordering $<$, and a digraph G together with the lists L is given. The objective is to find a mapping from G to H which preserves the adjacency and respects the lists L. In each step of the algorithm we choose an edge, $uv \in E(G)$, and we update $L(u)$ and $L(v)$. Let $x \in L(u)$ be the minimum vertex in the min ordering $<$ which admits a vertex $y \in L(v)$ with $xy \in E(H)$. Then let z be the minimum vertex of $L(v)$ in the min ordering $<$ such that x is adjacent to z. Now we remove the vertices of $L(u)$ that precede x and the vertices of $L(v)$ that precede z. There is no vertex less than x adjacent to a vertex in $L(v)$. Now suppose a vertex $x' > x$ is adjacent to a vertex $z' < z$, the min ordering of H implies x is adjacent to z'. Since z is the minimum vertex in the min ordering, such a z' does not exists. Thus the min ordering of H guarantees that the removed vertices can be ignored for the list homomorphism. In each step, if we remove a vertex from $L(u)$ then the edges adjacent to u must be checked again. In this process, we refine our lists step by step.

If eventually a list became empty, or there is no edge from $L(u)$ to $L(v)$ for two adjacent vertices $u, v \in V(G)$, then there is no list homomorphism from G to H. Otherwise, the updated L_v for each vertex $v \in V(G)$ is not empty. Also there is at least one edge from a vertex in $L(u)$ to a vertex in $L(v)$ for every $uv \in E(G)$. We map each vertex $v \in V(G)$ to the minimum vertex of $L(v)$ in the min ordering. The min ordering implies the minimum vertex of $L(v)$ and the minimum vertex of $L(u)$ are adjacent in H, if the edge uv is in $E(G)$. Thus this mapping is a homomorphism of G to H with respect to lists L.

Since H is fixed, each edge in $E(G)$ may rechecked at most $O(|V(H)|) = O(1)$ times. Also the minimum vertex can be found in constant time given the min ordering. Thus, the time complexity of the algorithm is linear time to size of G, $O(|E(G) + V(G)|$.

□

In the algorithm above, we enforced some conditions using the min ordering to reduce the size of the lists, thus reducing the search space. Such a technique is known as the

constraint propagation technique. It is used to solve many problems such as the constraint satisfaction problem, the network propagation problems, etc.

1.3 Undirected Graphs

1.3.1 Reflexive Case

Note that HOMH is trivial in the case of reflexive graphs. Even a single loop vv in $E(H)$ is sufficient to make HOMH trivial, since every graph G admits a homomorphism to H, by mapping all vertices of G to v. In the jobs assignment example in Section 1.1, it is natural to assume a machine is well connected to itself so we also obtain an instance of reflexive list homomorphism problem.

Here are some structures of H that play an important role in finding the line between hard and easy cases of the list homomorphism problem for reflexive graphs.

Theorem 1.3.1. *[17] If a reflexive graph H contains a chordless cycle then L-HOM H is NP-complete.*

This was proved independently by M. MacGillivray, as mentioned in [17]. Another interesting structure to explore is defined below.

Definition 1.3.1. An *asteroidal triple* of a graph H is a set of three non-adjacent vertices of H such that there is a path between each pair of vertices, not containing any neighbour of the third one.

Theorem 1.3.2. *[17] If a reflexive graph H contains an asteroidal triple then L-HOM H is NP-complete.*

This was proved in [17] using a reduction from not-all-equal 3-SAT without negated variable. Moreover, the following result was proved by Lekkerkerker and Boland.

Theorem 1.3.3. *[45] A graph H is an interval graph if and only if it is chordal and contains no asteroidal triple.*

This means the forbidden structures for interval graphs are precisely the NP-complete cases of the reflexive case of L-HOM, and it follows for reflexive graphs L-HOM H is NP-complete when H is not an interval graph. The following result proves that interval graphs yield polynomial time problems L-HOMH. This is called a dichotomy (for reflexive graphs) of the list homomorphism problem.

Theorem 1.3.4. *[17] For any reflexive graph H, L-HOM H is polynomial time solvable if H is an interval graph and NP-complete otherwise.*

We shall first present a proof using the min ordering property. We prove that reflexive interval graphs have a min ordering, thus using Theorem 1.2.1 we see that L-HOMH is polynomial time solvable, for a reflexive interval graph H.

Theorem 1.3.5. *A reflexive interval graph H has a min ordering.*

Proof. Suppose H is a reflexive interval graph with an interval representation I. We show that if we define the ordering $<$ of the vertices of H by the left endpoints of their corresponding intervals in I, we obtain a min ordering.

Assume u and v are two vertices in $V(H)$ with corresponding intervals I_u and I_v in I. Suppose we have an edge uv in H such that $u < v$; then the left endpoint of I_u precedes the left endpoint of I_v. Also, the adjacency of u and v implies that the left endpoint of I_v precedes the right endpoint of I_u. Suppose w appears between u and v in the the left endpoint ordering, i.e., $\ell_u < \ell_w < \ell_v$. Then the edge uw is in $E(H)$, since the left endpoint of I_w precedes the left endpoint of I_v, hence it precedes the right endpoint of I_u. Thus, the left endpoint ordering is a min ordering. □

Second, we present an alternate geometric proof from [17] by using a polynomial time reduction to the 2-satisfiability problem.

Proof. Suppose H is an interval graph with n vertices, and given G is a graph with lists $L(v)$ for every $v \in V(G)$. We assume H is given with its interval representation I. There are $2n$ endpoints in I. We define a set of $2n+1$ points, $P = \{p_0, p_1, ..., p_{2n}\}$ such that p_0 precedes the leftmost endpoint of the intervals in I, and the rightmost endpoint of the intervals in I precedes p_{2n}. We assume each p_i for $i = 1, 2, ..., 2n-1$ is placed in the gap between two consecutive endpoints in I. We introduce variables $l_{v,p}$ and $r_{v,p}$ for every $v \in V(G)$ and every $p \in P$. These variables bound the interval image of every vertex v in $V(G)$ such that $l_{v,p} = 1$ means the left endpoint of the interval corresponding to the vertex to which v is mapped precedes p and $r_{v,p} = 1$ means p precedes the right endpoint of the interval corresponding to the vertex to which v is mapped.

Now we introduce the clauses as below. To assure the adjacency of the adjacent vertices of H, there are clauses $l_{u,p} \vee r_{v,p}$ for every edge $uv \in E(H)$ where the left endpoint of u is less than the left endpoint of v in I. There are clauses $l_{v,p} \vee r_{v,p}$ for every vertex $v \in V(G)$ and every $p \in P$ so that the left endpoint of each interval image of v precedes its right endpoint. To insert lists conditions, we add clauses $\bar{l}_{v,p} \vee \bar{r}_{v,q}$ for each $v \in V(G)$ and for

each $p, q \in P$ where (p, q) is not contained in any interval corresponding to a vertex from $L(v)$. The pairs of clauses $\{(l_{v,p_{2k}}) \wedge (r_{v,p_0})\}$, for every vertex $v \in V(G)$, guarantee that the intervals lay inside (p_0, p_{2k}).

It is clear that if there exists a list homomorphism, the clauses are satisfiable. Now, we show a satisfying truth assignment defines a list homomorphism as follows. For every $v \in V(G)$, there exists the leftmost point (the smallest index) $p \in P$ such that $l_{v,p} = 1$, and the rightmost point (the largest index) $q \in P$ such that $r_{v,q} = 1$. Thus, there exists an interval image of a vertex $u \in L(v)$ that contains (p, q). We map v to u. The clauses assure that the adjacent vertices in G are mapped to vertices corresponding intersecting intervals, and the reduction is polynomial time since H with its interval representation I is fixed.

□

1.3.2 Irreflexive Case

To continue the discussion, let us take a look at irreflexive graphs. The following result can be concluded directly.

Theorem 1.3.6. *[30] L-HOM H is NP-complete when H is not bipartite.*

Proof. We mentioned that HOM H is NP-complete when H is not bipartite (Theorem 1.1.1). If we apply Theorem 1.1.2 then L-HOM H is also NP-complete. □

Considering this, we focus on bipartite graphs and our purpose is to formulate the algorithm for L-HOM H.

Theorem 1.3.7. *[18] Let H be a bipartite graph. If the complement of H is a circular arc graph, then L-HOM H is polynomial time solvable.*

The following theorem characterizes the class of bipartite graphs whose complement is a circular arc graph. We call this graphs *bipartite co-circular arc graphs* and sometimes abbreviate this as *bipartite co-CA graphs.*

Theorem 1.3.8. *[29] A graph H is a bipartite co-circular arc graph if and only if it admits a min ordering.*

Proof. Suppose H is a bipartite co-circular arc graph with bipartition (X, Y). Let $\overline{H}$ denote the complement of H. The graph $\overline{H}$ has a circular arc representation R which contains two points called the *north pole* and the *south pole*, such that every circular arc corresponding to a vertex $v \in X$ covers the north pole but not the south pole, and vice versa [64] [28].

We now construct a min ordering $<$ for the vertices in X according to the right endpoint ordering of the corresponding circular arcs in the clockwise direction from the north pole to the south pole, and for the vertices in Y according to the right endpoint ordering of the corresponding arcs in the clockwise direction from the south pole to the north pole.

Let uv and $u'v'$ be two different edges in $E(H)$ such that $u, u' \in X$ and $v, v' \in Y$ satisfy $u < u'$ and $v' < v$. Consider the corresponding circular arcs A_u, A_v, $A_{u'}$ and $A_{v'}$. Since u precedes u' and $A_{u'}$ and A'_v do not intersect, the right endpoint of A_u also precedes the right endpoint of $A_{v'}$ in the clockwise direction from the north pole to the south pole. Similarly, since v' precedes v, and A_v and A_u do not intersect, the right endpoint of $A_{v'}$ precedes the right endpoint of A_v in the clockwise direction from the south pole to the north pole. Therefore A_u and $A_{v'}$ do not intersect, so u and v' are adjacent in H, i.e., $uv' \in E(H)$. Thus the right endpoint ordering of H is a min ordering. □

By applying Theorem 1.2.1 and 1.3.8, we can conclude L-HOMH is polynomial time solvable for bipartite co-circular arc graph.

We also present an alternate proof from [18], similar to the proof of Theorem 1.3.4. We describe a polynomial time reduction from L-HOMH to the 2-satisfiability problem. This time we use the circular arc representation of $\overline{H}$ to bound the variables.

Proof. Let H be a bipartite graph with parts X and Y, and $\overline{H}$ a circular arc graph. For every circular arc representation of $\overline{H}$ if A_v and A_u denote two circular arcs corresponding to $v, u \in V(H)$ then there is an edge uv in $E(H)$ if and only if A_v and A_u do not intersect. We assume R is a circular arc representation of $\overline{H}$ such that there is a point N (respectively S) on the circle which is contained in all circular arcs corresponding to the vertices in X (respectively Y) but is not contained in any circular arc corresponding to vertices in Y (respectively X) [64][28]. Thus N and S play the role of the north and south poles. The circle is partitioned into two parts, the *east side* and the *west side*, with respect to the north pole and the south pole. We may also assume G is a bipartite graph with parts A and B such that for each vertex v in A (respectively B), $L(v) \subset X$ (respectively $L(v) \subset Y$).

We define two sets of points on the circle, P_E and P_W. The set P_E contains N, S and the endpoints of the circular arcs in the east side, and the set P_W contains N, S and the endpoints of the circular arcs in the west side. We define the variables $w_{v,p}$ for every $p \in P_W$ and every $v \in V(G)$, and $e_{v,p}$ for every $p \in P_E$ and every $v \in V(G)$. These variables bound the circular arc image of a vertex in G. We intend for $e_{v,p} = 1$ ($w_{v,p} = 1$) to mean the circular arc in R corresponding the vertex to which v is mapped, does *not* contain the point p. We ensure that adjacent vertices of G are mapped to vertices with non-intersecting

corresponding circular arcs in R, by the clauses $e_{u,p} \vee e_{v,p}$, for each edge uv in $E(G)$ and each p in P_E, and the clauses $w_{u,p} \vee r_{v,p}$, for each edge uv in $E(G)$ and each p in P_W. The clauses $\overline{e_{v,p}} \vee \overline{w_{v,q}}$ for each $v \in A$, and for each $p \in P_E$ and each $q \in P_W$ are imposed with the following constraint: the portion of the circle from the next point of p to the previous point q in the clockwise direction does not contain any circular arc corresponding to a vertex in $L(v)$. Also, we add similar clauses for each $v \in B$. With imposing these clause, we ensure that there exists at least one vertex in $L(v)$ to which v can be mapped. Finally, we add clauses $\{e_{v,n} \wedge w_{v,n}\}$ for every $v \in A$, and the clauses $\{e_{u,s} \wedge w_{u,s}\}$ for every $u \in B$, so that at least one $e_{v,p}$ and at least one $w_{v,p}$ is true.

It is clear that a list homomorphism from $G \to H$ implies a truth assignment for the above clauses, by setting $e_{v,p} = 1$ and $w_{v,p} = 1$ for each point p not contained in the circular arc corresponding to the vertex to which v is mapped. Now suppose we have a truth assignment, we define the list homomorphism as follows. Suppose $v \in A$, let $p \in P_W$ be the nearest point to p such that $w_{v,c} = 1$, and let $d \in P_E$ be the nearest point to p such that $e_{v,d} = 1$. Now in the clockwise portion of the circle between c to d should be at least one circular arc A_u from $L(v)$, we map v to u. We use the similar mapping for the vertices in B. It is clear that adjacent vertices are assigned to the vertices corresponding to non-intersecting circular arcs with respect to the lists.

□

Moreover, L-HOM H is NP-complete if H contains certain structures. An odd cycle is one such structure mentioned earlier. Equivalently, L-HOM H is NP-complete if H is not a bipartite graph. The following result considers L-HOM H for the bipartite graphs.

Theorem 1.3.9. *[18] If a bipartite graph H contains a chordless cycle of length greater than four, then L-HOM H is NP-complete.*

The proof consists of a polynomial time reduction from the k-colouring problem to L-HOM H.

Definition 1.3.2. An *edge asteroid* is a structure consisting $2k+1$ disjoint edges u_0v_0,u_1v_1, ..., $u_{2k}v_{2k}$, and $2k+1$ paths $P_{0,1}$, $P_{1,2}$, ..., $P_{2k,0}$, with the following constraints.

- Each $P_{i,i+1}$ joins u_i and u_{i+1}, for $i = 0, 1, ..., 2k$.
- There is no edge between $\{u_i, v_i\}$ and $\{v_{i+k}, v_{i+k+1}\} \cup V(P_{i+k,i+k+1})$, for $i = 0, 1, ..., 2k$ and subscripts are modulo $2k+1$.

- There is no edge between $\{u_0, v_0\}$ and $\{v_1,\ v_2,\ ...,\ v_{2k}\} \cup V(P_{1,2}) \cup V(P_{2,3}) \cup\ ... \cup V(P_{2k-1,2k})$.

See Figure 1.2.

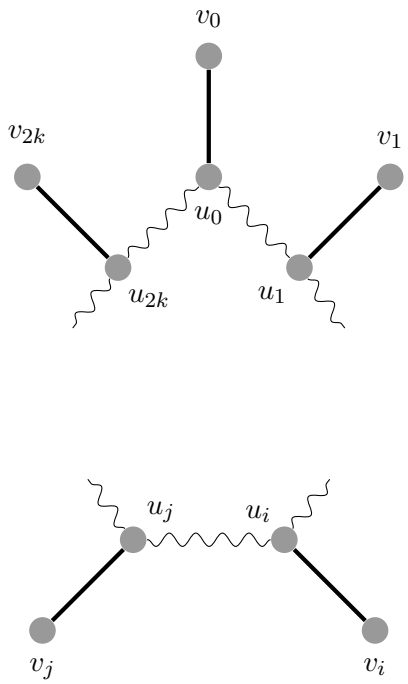


Figure 1.2: An edge-asteroid

The following theorem shows how an *edge asteroid* affect the list homomorphism problem.

Theorem 1.3.10. *[18] Let H be a bipartite graph. If H contains an edge-asteroid then L-HOM H is NP-complete.*

Hell and Huang proved the NP-completeness using a polynomial time reduction from the 3-colouring problem. For more details see [18].

Remark. L-HOM H is NP-complete when H contains the structures which are precisely the forbidden structures for co-bipartite circular arc graphs.

Theorem 1.3.11. *[18] A graph H is a bipartite co-circular arc graph if and only if H is chordal bipartite and contains no edge-asteroids.*

We can observe that this characterization of bipartite co-CA graphs is similar to Lekkerkerker and Boland's characterization of *interval* graphs in Theorem 1.3.3. This analogy extends to the dichotomy of the list homomorphism problems.

Theorem 1.3.12. *[18] For a irreflexive graph H, the problem L-HOM H is polynomial time solvable if H is a bipartite co-circular arc graph, and is NP-complete otherwise.*

1.3.3 General Case

A *general* graph is a graph where each vertex may or may not have a loop. In this section, we will show how to extend the complexity classifications from the previous sections (reflexive, irreflexive) to general graphs.

The distinction between hard and easy case of the list homomorphism problem for general graphs turns out to depend on the following definition.

Definition 1.3.3. Let C be a circle with two specified points N and S on C. A *bi-arc* is an ordered pair of circular arcs (A, B) on C such that A contains N but not S, and B contains S but not N. A graph H is a *bi-arc graph* if there is a family of bi-arcs $\{(A_x, B_x) : x \in V(H)\}$ such that for any $x, y \in V(H)$, not necessarily distinct, the following conditions hold:

- if x and y are adjacent then neither A_x intersects B_y nor A_y intersects B_x;
- if x and y are non-adjacent then both A_x intersects B_y and A_y intersects B_x.

We define an auxiliary graph as follows, for better understanding of the bi-arc graphs.

Definition 1.3.4. Given a graph H, the auxiliary bipartite graph H^* is defined with parts $X_H = \{x' : x \in V(H)\}$ and $Y_H = \{x'' : x \in V(H)\}$. The edge set of H^* consists of all edges $x'y''$ such that xy is an edge of H.

The following simple, yet useful observations describe how the auxiliary graph aids in the proofs.

Observation. *The graph H^* has no loops.*

The edges of H^* only join the vertices of X_H to the vertices of Y_H.

Observation. *A bi-arc representation of the graph H is precisely a circular arc representation of the auxiliary graph H^*.*

The following results from [19] show that the two graph classes we discussed earlier are bi-arc graphs: interval graphs from Section 1.3.1 and bipartite co-CA graphs in Section 1.3.2.

Theorem 1.3.13. *[19] Let H be a reflexive graph, Then H is a bi-arc graph if and only if H is an interval graph.*

Theorem 1.3.14. *[19] Let H be an irreflexive graph, Then H is a bi-arc graph if and only if H is a bipartite co-circular arc graph.*

Now, we wrap up the discussion of the list homomorphism problem for graphs with the following result which shows the dichotomy of list homomorphism problem for the general graphs and the importance of bi-arc graphs.

Theorem 1.3.15. *[19] Let H be a graph. The problem L-HOM H is polynomial time solvable if H is a bi-arc graph, and is NP-complete otherwise.*

1.4 Digraphs

In this section, we discuss a more general case of the problem, the list homomorphism problem for digraphs. Digraphs are more interesting from the point of view of homomorphisms and list homomorphisms, since the mappings must preserve both adjacency and the direction of the edges. Bulatov's theorem [5] classifies the tractable cases of the list homomorphism problems on very general structures including digraphs. For digraphs, there are more precise classification results discussed below which give a combinatorial classification.

1.4.1 Reflexive Case

We already discussed reflexive graphs, here we describe approaches to solve the list homomorphism problem of reflexive digraphs by examining different structures. The list homomorphism problem restricted to interval graphs is polynomial time solvable. For reflexive digraphs, we shall introduce a similar definition of *interval digraphs*.

Definition 1.4.1. An *interval digraph* is a digraph H which admits an interval pair representation, which is a family of pairs of intervals (I_v, J_v) for every $v \in V(H)$ such that $uv \in E(H)$ if and only if I_u intersects J_v.

Recall that due to the definition of interval graphs, they must be reflexive, but unlike interval graphs, interval digraphs may lack loops. Hell and Rafiei [31] introduced the so-called *adjusted interval graphs* which are reflexive.

Definition 1.4.2. [20] Let H be a interval digraph. If H admits a representation by intervals I_v, J_v, $v \in V(H)$, such that for each v, the intervals I_v and J_v share the same left endpoint, we call H an *adjusted interval digraph (in a short form, AID).*

It is clear that an adjusted interval digraph must be reflexive. Now we describe how this subclass of interval digraphs helped solving the list homomorphism problem for the reflexive digraphs.

Theorem 1.4.1. *[20] A reflexive digraph H is an adjusted interval digraph if and only if it admits a min ordering.*

We will show one direction of the proof, namely that an adjusted interval digraph has a min ordering. The complete proof is in [20].

Suppose H is an adjusted interval graph with a representation I. We show that if we define the ordering $<$ of the vertices of H by the ordering of the common left endpoints of their corresponding pairs of intervals in I, we obtain a min ordering.

Assume u and v are two vertices in $V(H)$ with the corresponding pairs of intervals I_u, J_u and I_v, J_v in I. Suppose we have an edge uv in H such that $u < v$; then the common left endpoint of I_u and J_u precedes the common left endpoint of I_v and J_v. Also, the adjacency of u and v implies that the left endpoint of J_v precedes the right endpoint of I_u. Suppose w appears between u and w in the the common left endpoint ordering. Then the edge uw is in $E(H)$, since the left endpoint of J_w precedes the left endpoint of I_v, hence it precedes the right endpoint of J_u. Thus, the common left endpoint ordering is a min ordering.

Corollary. *If a reflexive digraph H is an adjusted interval digraph, then LHOMH is polynomial time solvable.*

Interval graphs define the dichotomy of the list homomorphism problem for reflexive graphs. The following conjecture attempts to classify the dichotomy of list homomorphism for reflexive digraphs, in a similar way.

Conjecture 1.4.2. *[20] If a digraph H is not an adjusted interval digraph then L-HOMH is NP-complete.*

We note that, a polynomial time algorithm for recognition of an AID is described in [20].

1.4.2 General Case

In this section, we discuss the list homomorphism problem on digraphs in the general case. We defined an asteroidal triple as a forbidden structure for interval graphs according to [45]. That structure makes L-HOM H NP-complete on reflexive graphs. Here, we introduce a

similar variation of an asteroidal triple for digraphs, called a *digraph asteroidal triple* (in a short form, DAT).

Before we define a DAT, we should define some preliminaries concepts. Let H be a digraph. We call $uv \in E(H)$ a *forward edge* of H and vu is a *backward edge* of H. We say two walks $P = x_0, x_1, ..., x_n$ and $Q = y_0, y_1, ..., y_m$ in H are *congruent*, if $x_i x_{i+1}$ is a forward (respectively backward) edge if and only if $y_i y_{i+1}$ is a forward (respectively backward) edge. If P and Q are two congruent walks in H, P *avoids* Q, if there is no edge $x_i y_{i+1}$ with the same direction as $x_i x_{i+1}$.

Definition 1.4.3. An *invertible pair* in H is a pair of vertices u, v, such that

- there exist congruent walks P from u to v and Q from v to u such that P avoids Q
- and there exist congruent walks P' from v to u and Q' from u to v such that P' avoids Q'

Definition 1.4.4. A *permutable triple* in H is a triple of vertices u, v, w together with six vertices $s(u), b(u), s(v), b(v), s(w), b(w)$, which satisfy the following condition.

For any vertex x from $\{u, v, w\}$, there exists a walk $P(x, s(x))$ from x to $s(x)$, and two walks $P(y, b(y))$ from y to $b(y)$ and $P(z, b(z))$ from z to $b(z)$, congruent to $P(x, s(x))$, such that $P(x, s(x))$ avoids both $P(y, b(y))$ and $P(z, b(z))$.

Now we can give the following essential definition.

Definition 1.4.5. A *digraph asteroidal triple* (DAT) is a permutable triple u, v, w with $s(u), b(u), s(v), b(v), s(w), b(w)$ such that for any vertex x from $\{u, v, w\}$, the pair of vertices $(s(x), b(x))$ is an invertible pair.

The presence of a DAT defines the dichotomy of the list homomorphisms problem for digraphs.

Theorem 1.4.3. *[31] If H contains a DAT, L-HOM H is NP-complete. If H is DAT-free, L-HOM H is polynomial time solvable.*

Although Bulatov [5] proved the existence of a dichotomy for all Constraint Satisfaction Problems, this is the first structural classification of the dichotomy. The structure *DAT* delineates the dichotomy of list homomorphisms for digraphs.

Chapter 2

Interval Graphs and Circular Arc Graphs

In this chapter, we discuss different geometric representations of graphs. Specifically, we investigate interval graphs and circular arc graphs.

2.1 Introduction

We have discussed what a graph is and how it is an abstraction of a relation amongst a set of objects. We start with answering a question, "What is a representation of a graph ?" or "How can we represent a graph?". Every graph can be represented by an adjacency matrix or by adjacency lists.

We focus on the types of representations of a graph which also describe a property of the graph. The graphs which can be represented in a certain way, make a graph class. There are many graph classes defined by limiting their representations.

Here we discuss a specific graph representation which is defined as follows.

Definition 2.1.1. Given a family of sets $S = \{S_1, S_2, ..\}$, we define the *intersection graph of* S, H as follows. Each set S_i corresponds to a vertex v_i in $V(H)$, and $v_iv_j \in E(H)$ if and only if $S_i \cap S_j \neq \emptyset$.

We say a graph H is an *intersection graph*, if there is a family of S such that H is the intersection graph of S.

It is clear that the subfamily of a family S that is represented by an intersection graph H, represents an induced subgraph of H. Thus we have the following remark.

Remark. Every induced subgraph of an intersection graph is an intersection graph.

The class of intersection graphs includes a wide range of graphs. There are different types of intersection graphs based on types of sets. Here are a few examples of intersection graphs created from a geometric representation of sets.

A graph H is *planar* if H has an embedding of vertices to the points on the plane such that two vertices are adjacent if and only if there is a continuous curve on the plane connecting these two points and no pair of curves crosses each other in any points other than their endpoints. Equivalently it can be shown [8], a planar graph H is the intersection graph of a family of line segments. It was Scheinerman's conjecture [56], proved by J. Chalopin and D. Gonalves [8]. Moreover, by the circle packing theorem of [42], planar graphs are also exactly the intersection graphs of families of non-crossing circles on the plane. The circles can only be tangent to one another and only two circle can be tangent in one point.

The two geometric intersection graphs which we mainly focus on are interval graphs and circular arc graphs. We discuss certain characterizations of interval graphs and circular arc graphs in the rest of the chapter. We discuss interval graphs in Section 2.2 and circular arc graphs with more details in Section 2.4.

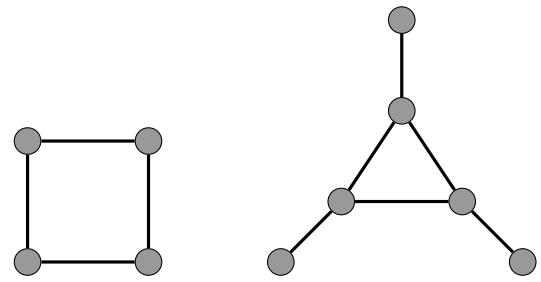
2.2 Interval Graphs

In this section, we will discuss a class of intersection graphs which has a rich combinatorial structure, namely interval graphs. Recall the definition of interval graphs in Section 1.1. (See Definition 1.1.6)

It is easy to see that the definition given in Section 1.1 mean that, interval graphs are the intersection graph of families of intervals on the real line. As for intersection graphs, any induced subgraph of an interval graph is also an interval graph. This property leads to the fact that interval graphs are characterized by forbidden structures. Recall Theorem 1.3.3, by which Lekkerkerker and Boland introduced the earliest characterization of interval graphs using forbidden structures, namely a chordless cycle of length greater than 3 and a asteroidal triple.

For example, the graphs in Figure 2.1 are two minimal forbidden structures for interval graphs. Consider a chordless cycle of length four in Figure 2.1(a). Any three vertices induce a path of length 2, which can only represented by 3 consecutive intervals. It is clear that there is no place for an interval intersecting the two interval at the end but avoiding the middle interval. The same argument applies to any induced cycle of length greater than 4. Now consider the graph *2-net* which is an asteroidal triple, depicted in Figure 2.1(b).

The three pairwise adjacent vertices in the center can only represented by three intervals sharing the same point. Each of these vertices has a neighbour non-adjacent to the other two. Intervals can only extend in two directions from the shared point, thus there is no place for the third interval.



(a) an induced C_4

(b) *2-net*

Figure 2.1: Examples of forbidden structures for interval graphs

There are different variations of interval graphs. For example, *proper interval graphs* are graphs that have an interval representation in which no interval is contained in another interval, and *unit interval graphs* are graphs with an interval representation such that all intervals have an equal length. These two definitions turn out to define the same class of graphs [55].

Interval graphs have interesting applications in many different research areas such as biology, archaeology and so on. Here are a few problems related to interval graphs.

One of the early applications interval graphs is the archaeological problem of finding '*chronological seriation*' of antique objects. Suppose we have a number of graves. The objects found in a grave co-existed in a same specific time period. The problem is finding the timeline of these antique objects. Each object's timeline may be considered as an interval on a historical timeline. Assume a graph H is the intersection graph of family of historical timelines of objects. Each grave corresponds to a clique in H. The interval representation of H is the chronological ordering of objects in time. More details on the problem can be found in [39].

Also Benzer [1], in molecular biology, proposed a *linear arrangement of genes* in the chromosome. In mutated versions of a chromosome, part of the genes may have changed. Two mutated version of the original chromosome may have similar properties if they both share a mutated gene. There are many questions that can be answered when the intersection

graph of mutated chromosomes is represented as an interval graph. For instance, finding the largest clique in the interval graph is equivalent to finding a mutated gene shared by the most mutations.

The *job scheduling problem* is another interesting application of interval graphs. Given a set of jobs with interval time periods of processing times, the problem asks for partitioning the jobs into a minimum number of sets such that each set of jobs can be done consecutively. In other words, no two jobs in each set conflict with each other. The intersection graph of the time periods is an interval graph. The job scheduling problem is exactly the colouring problem for the interval graph.

The *memory allocation problem* is defined as follows. A compiler is running different programs that may interfere with each other. Two programs may conflict when both needed a same memory unit at the same time. The problem is to find the total size of memory needed for the programs, given time intervals of each program. Let G be a graph with program units as vertices. Two program units are adjacent if they conflict. Each clique represents a set of running program units, accessing the memory units simultaneously. The problem is exactly finding the maximum clique size of G, given the time intervals and the memory requests of each program.

Since interval graphs have a natural representation by intervals, restricting to interval input graphs makes specific problems easier to solve. One of the interesting properties of interval graphs is the following.

Definition 2.2.1. An intersection graph G has the *Helly property* if for every clique H in G, all the sets in H share an element.

Interval graphs have the Helly property. The Helly property of an interval graph G implies that the intervals corresponding to a clique in G contain the same point. Using this fact, we can find the largest clique in an interval graph G. Suppose G was given by its interval representation. We traverse the interval representation of G. We set a variable x to zero and for each appearance of left endpoint increase x, and for each appearance of right endpoint decrease x. The maximum value of x is the maximum clique size of the graph. Thus problem of finding the maximum clique in an interval graph has a linear time algorithm, provided a representation by intervals is given.

This shows the importance of finding an interval representation for an interval graph. We discuss this in Section 2.3.

2.3 Recognition of Interval Graphs

In the previous section, we discussed how interval graphs allow efficient algorithms for certain problems. This is useful especially when we can recognize an interval graph and describe an interval representation for it in a reasonable time, especially linear time. Here, we define the interval graph recognition problem as follows.

```
INPUT: A graph H.
QUERY: Is the graph H an interval graph ?
If yes, describe the interval representation
```

This is slightly different from the decision problem of interval graphs recognition. In this version of the problem, not only we recognize the interval graphs, but also an interval representation is required.

Different approaches to the problem have been made. The earliest polynomial time recognition belongs to Gilmore and Hoffman [22]. They reduce the problem to an ordering of maximal cliques such that the maximal cliques containing a same vertex are consecutive in the ordering. Based on this, some 10 years later the first linear time recognition algorithm was introduced by Booth and Leuker [4]. The algorithm is very efficient but uses complex data structures (PQ-trees). Korte and Möhring [43] developed modified PQ-trees with a simpler update process, for recognizing interval graphs.

Later on, Hsu and Ma [34] gave a simpler linear time for the task using modular decomposition, not PQ-trees. Also Habib et.al. [25] introduced a partition refinement technique for ordering the vertices of a graph. It is much easier to implement than the previous algorithms. The most recent efficient algorithm to recognize interval graphs is based on the lexicographical breadth first search (in a short form, Lex-BFS or LBFS) [11].

Kratsch et. al. [44] modified Korte's and Möhring's algorithm [43] into the first certifying algorithm for the recognition of interval graphs. A *certifying algorithm* is an algorithm which not only solves the problem, but also provides a certification to prove its solution is correct. In this case, if the algorithm recognized an input graph G is not an interval graph then it provides an evidence, namely an asteroidal triple or a chordless cycle (cf. Theorem 1.3.3). With a certification, one be sure of the correctness of the result of the algorithm. The concept of certification algorithms was emphasized by McConnell cf. [48]. We will discuss a linear time algorithm for recognition of circular arc graphs in Section 2.5.1, which can be made certifying [49].

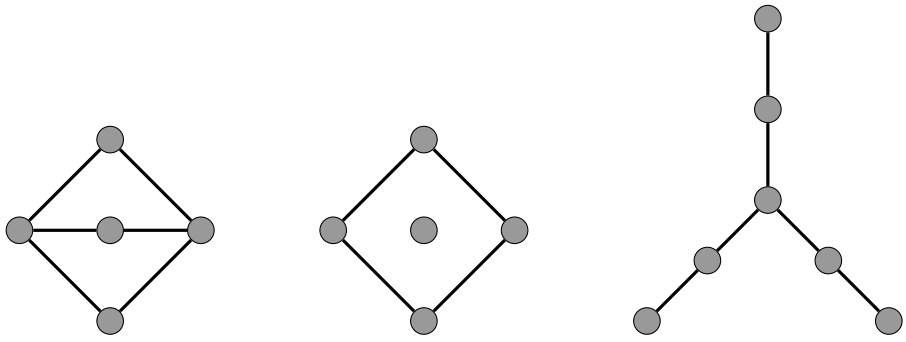
Note that the complements of interval graphs can also be recognized in linear time [50]. This fact motivates us to investigate the recognition of the complements of circular arc

graphs in Chapter 3.

2.4 Circular Arc Graphs

The class of circular arc graphs was introduced by Hadwiger, Debrunner and Klee in 1964 [40], cf. also Klee [26], and studied more specifically by Tucker [64]. Recall the definition of circular arc graphs from Chapter 1.1. (See Definition 1.1.7.)

We discuss the recognition of circular arc graphs in Section 2.5.1. As mentioned earlier, every interval graph is a circular arc graph. Thus, the class of circular arc graphs is a superclass of the class of interval graphs. The forbidden structures of circular arc graphs are a subset of the forbidden structures of interval graphs. For example, the graphs in Figure 2.2 are a few minimal forbidden structures of circular arc graphs [2]. For instance, consider the graph $K_{2,3}$ in Figure 2.2(a). The central vertices must correspond to three disjoint circular arcs around the circle. If a circular arc a intersects all three circular arcs then a must contain one. Because of this containment, any other circular arc b can not intersect the contained arc while avoiding a. The argument for the *bipartite claw* in Figure 2.2(c) is similar. For the graph C_4^* , if there is a circular arc representation for the graph C_4^* in Figure 2.2(b) then the portion of the circle not intersecting the circular arc corresponding to the single vertex in the center, is an interval representation for the graph C_4. Recall that the graph C_4 is not an interval graph, thus the graph C_4^* is not a circular graph.



(a) $K_{2,3}$ (b) C_4^* (c) bipartite claw

Figure 2.2: Examples of forbidden structures for circular arc graphs [2]

A circular arc representation R of a graph H is a set of arcs A on a circle. Each arc $a_i \in A$ is denoted by $a_i = (\ell_i, r_i)$ where ℓ_i and r_i are the endpoints of a_i. Thus we can

represent R by the clockwise circular ordering of the endpoints of arcs. We may assume no two arcs share an endpoint, otherwise we slightly perturb the endpoints [49].

Unlike interval graphs, circular arc graphs do not have the Helly property. On the real line, a K_3 can only be represented by three intervals intersecting in a region, as in Figure 2.3(a). On a circle, a K_3 can also be represented by three pairwise intersecting arcs covering the circle, as in Figure 2.3(b).

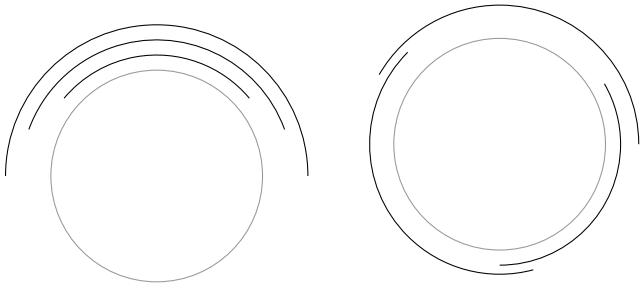


Figure 2.3: Two different circular arc model for K_3

Circular arc graphs model many situations and have many applications in different areas. Circular arc representation helps the analysis of these problems, just as interval representations of interval graphs do. Basically, every periodically repeated intervals have natural representation by circular arc graphs. Here we describe a few examples.

Example. [9] The *Periodic Allocation Problem* is a generalized version of the dynamic storage allocation problem, where items appear repeatedly. Assume a loop in a computer program, and the flow control of the loop, is described by a circle. The compiler controls the memory allocation of variables associated with the loop. In the register allocation problem exactly one of a set of memory registers has to be allocated to a variable within the loop for its lifetime. The objective is to minimize the size of the total allocated memory. The lifetimes of variables within a loop can be regarded as circular arcs, and the periodic allocation problem is precisely the k-colouring problem on circular arc graphs. An approximation algorithm for colouring a weighted circular arc graph representing an instance of the periodic allocation problem is given in [9].

Example. [63] The *Routing and Wavelength Problem* is defined as follows. Consider an optical ring network with a single ring. Assume there is a set of requests along the ring network. There are two possible routes for each request - each route can chose either the clockwise direction or the counter-clockwise direction on the ring. Every route can be

regarded as a circular arc on the ring. The intersection graph of the routes on the ring is a circular arc graph G. The routing problem is to find a routing with minimum size of the maximum clique in G. For a fixed routing, the sub-problem of finding maximum number of conflicting routes is exactly the maximum clique problem for circular arc graphs. An approximation algorithms for the routing problem can be found in [63].

2.5 Recognizition of Circular Arc Graphs

The recognition of circular arc graphs was at first conjectured to be NP-complete by Booth [3]. The earliest polynomial time circular arc graphs recognition algorithm belongs to Tucker [64]. Tucker solved the problem in time $O(n^3)$. Here we quote Spinrad's comment on Tucker's algorithm from Spinrad's book [27]:

> This algorithm has a reputation for being difficult to understand. In part, this is because the algorithm is complex, but the reputation of the algorithm has been exaggerated by a number of issues. Most importantly, Tucker himself talks about the need for simplification in the paper. [Tucker's statements from the paper include 'The real difficulty in our algorithm is not with speed but its length' and 'As mentioned earlier, simplification rather than greater speed is what really needed, since the constant in $O(n^3)$ is likely to be horrendous'. The author [Spinrad] disagrees slightly with the latter statement; I do not believe the constant in the running time is such a problem, but it is very difficult for someone who has not studied the paper very carefully to produce a correct implementation of the algorithm.]

Eschen and Spinrad [15] used the same approach as Tucker. However, studies of chordal bipartite graphs helped them to refine Tucker's algorithm, and reduce its complexity to $O(n^2)$. One of the important tasks in their approach is so-called neighbourhood containment. The neighbourhood containment problem is to determine for every pair of adjacent vertices u and v whether or not $N[v] \subset N[u]$. Eschen and Spinrad mentioned in [15] that neighbourhood containment was the bottleneck in Tucker's algorithm. Meanwhile, Hsu [35] introduced an $O(nm)$ algorithm for the task. Later on, McConnell [49] refined the neighbourhood containment technique and developed the first linear time algorithm to recognize a circular arc graph. McConnell's algorithm creates a matrix to characterize relations between the arcs, then constructs a so-called interval realizer from the matrix using modular decomposition and builds the circular arc model based on it. The most difficult part of McConnell's algorithm was to create an interval realizer with certain relations between arcs.

Recently, Kaplan and Nussbaum [38] introduced a simpler linear time algorithm mainly based on Eschen-Spinrad's algorithm with a tighter analysis.

In the next section, we explain how circular arc graphs can be recognized in linear time, based mostly on [38].

2.5.1 Linear-time Recognition of Circular arc Graphs

In this section, we provide some details of the Kaplan-Nussbaum's algorithm to solve the recognition problem for circular arc graphs in linear time. This algorithm carefully implements a compilation of Tucker's, Eschen-Spinrad's and McConnell's algorithm.

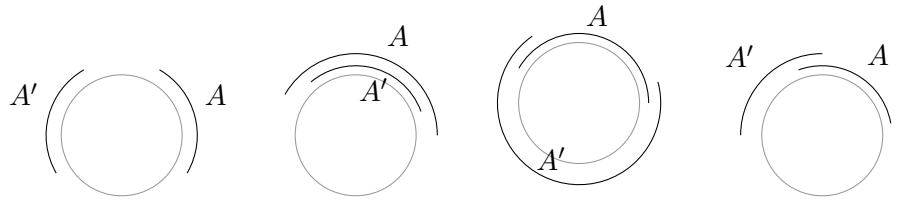
Given a graph G, we will recognize whether or not G is a circular arc graph. If the answer is positive, we will construct a circular arc representation R for G. The algorithm works from the assumption that G is a circular arc graph, and it tries to create a circular arc model for G. We develop a representation R step-by-step and when the algorithm halts with the circular arc representation R, the graph G is a circular arc graph if R realizes G, otherwise somewhere along the way we declare failure. We suppose G is given by its adjacency lists, so the size of the input is $O(n + m)$. We will show that every step of the algorithm runs in linear time, i.e., time $O(n + m)$.

The basis of this approach is a classification of the intersection types of pairs of arcs. The different possible types of intersection between arcs A and A' in a circular arc representation R are listed below and illustrated in Figure 2.4.

- A and A' are *disjoint*, i.e., $A \cap A' = \emptyset$.
- A *contains* A' , i.e., $A' \subset A$. The arc A covers both endpoints of the other arc A' and A' is laid inside of A.
- A and A' *cover the circle* , i.e., $A \cup A' = C$. The two arcs will cover the circle together, so both endpoints of one arc are covered by the other one.
- A *overlaps* A' , i.e., $A \cap A' \neq \emptyset$, $A - A' \neq \emptyset$ and $A - A' \neq \emptyset$. They cover only one endpoint of each other.

For the purposes of constructing a circular arc representation of a graph G, it is not difficult to see that the anticipated types of intersection between arcs are closely related to the neighbourhood containment relationships between the corresponding vertices.

For example, let A_i, A_j, A_k be the corresponding circular arcs for vertices v_i, v_j, v_k. If A_i is contained in A_j, then every circular arc A_k that has a intersection with A_i also



(a) disjoint (b) contain (c) cover the circle (d) overlap

Figure 2.4: Different types of intersection between two arcs A and A'

intersects A_j. Thus every neighbour of v_i is a neighbour of v_j, i.e., $N(v_i) \subset N(v_j)$. Now suppose A_i and A_j cover the circle. Each circular arc A_k that intersects A_j but is disjoint from A_i, must be contained in A_j and vice versa. Thus every neighbour of v_k is also a neighbour of v_j, i.e., $N(v_k) \subset N(v_j)$.

We pre-process the input graph to find two types of vertices that can be ignored in the processing of the algorithm: *clique modules* and *universal vertices*. (See Definitions 1.1.2 and 1.1.1.)

A universal vertex in a graph G can be found in linear time and can be ignored, because if R is a circular arc representation of $G - v$, by adding a circular arc for v that goes almost all the way around the circle, we obtain a circular arc representation of G.

To find all clique modules in G, we start with the closed neighbourhood lists $N[v]$ for every vertex $v \in V(G)$. Assume the vertices are $1, 2, .., n$. We can sort each $N[v]$ using radix sort. Two vertices u and v are in a same clique module in G when $N[v] = N[u]$, i.e., their sorted closed neighbourhood lists are identical. McConnell [49] introduced the *radix partitioning* procedure which finds identical strings in a list of strings. The procedure is similar to radix sort. In i-th phase buckets contain strings that are identical on the first i characters. In each phase we keep track of non-empty buckets, thus we do not spend time on every bucket in each phase. The procedure can recognize these identical strings in $O(n+m)$ where n is number of strings and m is the total summation of strings lengths.

Let M be a clique module in G with vertices $\{u_0, u_1, .., u_k\}$. We keep one vertex from M, say u_0, and remove all other vertices of M from G, to obtain G_{new}. If G_{new} is a circular arc graph with a circular arc representation R_{new}, we construct a circular arc representation R for G as follows. Let a_0 denote the circular arc corresponding to u_0. We repeat a_0 for each circular arc a_i corresponding to u_i, so that each a_i intersects a_0 and the same arcs that a_0 intersecting. Therefore, we can ignore clique modules.

Now that the graph has been preprocessed, we continue with defining a specific type of representation so-called a *normalized* circular arc representation of the preprocessed graph.

Definition 2.5.1. Suppose H is a circular arc graph without a universal vertex or clique module. A circular arc representation R of H is a *normalized circular arc representation* of H if the following conditions hold. Suppose v and v' are two adjacent vertices in H, and A and A' are the corresponding arcs in R.

- A contains A' if v and v' are adjacent and $N[v'] \subset N[v]$.
- A and A' cover the circle if v and v' are adjacent, $N[v] \cup N[v'] = V(H)$, and for each vertex $u \in N[v]$, we have $N[u] \subset N[v]$ if u is not adjacent to v', and for each $u' \in N[v']$ we have $N[w'] \subset N[u]$ if u' is not adjacent to v'.
- otherwise A (singly) overlaps A'.

As we have already discussed, the conditions are based on the natural relations between arcs. The existence of a normalized representation for any circular arc graph with no universal vertex or clique modules was proved by Hsu [35].

Theorem 2.5.1. *[35] Let G be a circular arc graph with no universal vertex or clique module. Then there exists a normalized circular arc representation of G.*

To prove the theorem, we show that every circular arc representation of G can be made normalized. Our proof is simpler, since we only care about existence, not the time efficiency.

Proof. Since G is a circular arc graph, we assume R is an arbitrary circular arc representation of G. We will show how to adjust the ordering of endpoints in R with a two-step algorithm such that the new R is normalized and still represents G. Consider a pair of adjacent vertices v and v' in $V(G)$ and their corresponding arcs A and A' in R. Suppose $N[v'] \subset N[v]$ but A does not contain A'. Since G has no clique modules, $N[v'] \neq N[v]$, therefore A' does not contain A. Also G has no universal vertex, $N[v] \neq V(G)$, therefore A and A' do not cover the circle. Thus A can only overlap A'. If A contains the right (left) endpoint of A', we extend the left (right) endpoint of A up to the left (right) endpoint of A' so that A contains A'. We repeat this step until the second condition of a normalized circular arc representation is satisfied. It is clear during the changes, no new intersection was created since $N[v'] \subset N[v]$, thus the adjusted R is still represents G.

In the second step, suppose for a pair of vertices v and v' in $V(G)$, such that $N[v] \cup N[v'] = V(G)$, for each $u \in N[v] \setminus N[v']$ we have $N[u] \subset N[v]$, and for each for each $u' \in N[v'] \setminus N[v]$

we have $N[u'] \subset N[v']$, but their corresponding arcs A and A' do not cover the circle. Since G has no universal vertices, A does not contain, or is contained in, A'. Therefore, A overlaps A'. If A contains the right (left) endpoint of A' then we extend the right (left) endpoint of A' next to the left (right) endpoint of A so that A and A' cover the circle. Repeat this step, until there are no pairs of such vertices. It is clear that the adjusted R still represents G, since every arc corresponding to a non neighbour of v' is contained in A', so during the extension of A' no such arcs intersect A'.

□

We continue demonstrating how to find the neighbourhood containment and also check the conditions of normalized circular arc representation in linear time. Tucker [64] gave an $O(n^3)$ algorithm to do this. Later on Spinrad [15] gave another algorithm which runs in $O(n^2)$ and McConnell [49] slightly tighten the analysis of Spinrad algorithm to get linear time bound, $O(n + m)$. The following theorem was the first linear time neighbourhood containment recognition, introduced by McConnell.

Theorem 2.5.2. *[49] If G is circular arc graph, we can distinguish the following two cases of relations between neighbourhood of every two adjacent vertices v and u of G in time $O(n + m)$.*

- $N[v] \subset N[u]$.
- $N[v] \cup N[u] = V(G)$

To prove the first part of the theorem, we introduce two disjoint subset of $V(G)$. Suppose v_0 is a vertex with minimum degree in G, let $W = N[v_0]$ and $U = V(G) \setminus W$. Instead of looking for neighbourhood containment over G, we check the containments in these two disjoint subsets of $V(G)$. Specifically, $N[v] \subset N[u]$ if and only if $(N[v] \cap W) \subset (N[u] \cap W)$ and $(N[v] \cap U) \subset (N[u] \cap U)$. Also $N[v] \cup N[u] = V(G)$ if and only if $(N[v] \cap W) \cup (N[u] \cap W) = W$ and $(N[v] \cap U) \cup (N[u] \cap U) = U$. We build an auxiliary bipartite graph D with parts W' and $V(G)$ where W' is a disjoint copy of W. Two vertices $x \in V(G)$ and $y \in W'$ are adjacent in D, if and only if they are non-adjacent in G. In particular, each $x \in W \subset V(G)$ is non-adjacent to the corresponding $x' \in W'$.

The following lemma was proved by Spinrad [15].(See Definition 1.1.5.)

Lemma 2.5.3. *[49] D is chordal bipartite.*

Proof. Suppose D has a chordless cycle C of length $2k \geq 6$, $v_1 w_1 v_2 w_2 .. v_k w_k v_1$ such that $v_i \in V(G)$ and $w_i \in W'$. Consider the circular arc representation R of G, a pair of vertices

w_i and w_j, and their corresponding circular arcs A_i and A_j in R. For any v_i, Let B_i denote the circular arc corresponding to v_i in R including the vertex v_0 with minimum degree. The neighbourhood of w_i in G is not contained in w_j, thus A_i is not contained in A_j. Therefore, the left endpoint ordering of circular arcs corresponding to any w_i for $i = 1, 2, .., k$ is the same as their right endpoint ordering. We order every w_i according to the left endpoint ordering of their corresponding circular arcs in the clockwise direction from the right endpoint of B_0. Since v_i is adjacent to w_{i-1} and w_i in D, B_i avoids A_{i-1} and A_i. Also, B_i avoids any circular arc corresponding to a vertex w_j between w_i and w_{i-1} in the ordering, thus w_i and w_{i-1} are consecutive in the ordering. Without loss of generality, assume w_1 is the first vertex in the ordering. Since w_i and w_{i-1} are consecutive in the ordering, w_k is the last vertex in the ordering. We consider two cases for v_1, either v_1 is adjacent to v_0 in G or not.

If v_1 is adjacent to v_0 in G then the circular arc B_1 avoids A_1 and A_k, but it intersects the circular arc B_0 corresponding to v_0 and every A_i for $1 < i < k$. Thus B_1 must be contained in B_0, which contradicts the fact that v_0 is a minimum degree vertex in G, because there are no clique modules in G.

Now suppose v_1 is non-adjacent to v_0 in G. Since the circular arc B_1 corresponding to v_1 avoids A_1 ,A_k and B_0, it is impossible for B_1 to intersect any A_j for $1 < j < k$, while avoiding B_0. Therefore, D is chordal bipartite. □

Let v and u be two adjacent vertices in G. We now consider two cases with respect to our partitioning of $V(G)$ into U and W.

- $(N[v] \cap U) \subset (N[u] \cap U)$.

  To check this, we first test whether or not U is an interval graph. If U is not an interval graph, then G is not a circular arc graph. Otherwise, we can find an interval representation of U in linear time [44]. Let I be an interval representation of U. For any vertex w in G, we denote by ℓm_w the leftmost endpoint of the intervals of I corresponding to the vertices in $N[w] \cap U$. Similarly, we denote by rm_w the rightmost endpoint of such intervals. We can find neighbourhood containment of two vertices u and v, in U by checking the order of ℓm_u, ℓm_v, rm_u and rm_v in I. Finding these extreme endpoints for $v \in V(G)$ only take time $O(|N(v)|)$, by comparing the endpoints of each interval, since each comparison of the left and right endpoints of intervals given the interval representation of U will take constant time.

- $(N[v] \cap W) \subset (N[u] \cap W)$.

  Suppose G is a bipartite graph with parts X and Y. Spinrad [61] proved that either

finding neighbourhood containment of all pairs of vertices of G or determining G is not chordal bipartite, will take $O(n_1n_2 + k)$ time where there are k pair of adjacent vertices in G. We apply Spinrad's technique to find neighbourhood containment in D. This will solve the neighbourhood containment of two adjacent vertices in $V(G)$ with restriction to W. Since $|W| = deg(v_0) = O(m/n)$, it only takes $O(m + n)$ time.

We proved that if two vertices are adjacent in the graph, we can determine whether or not the neighbourhood of one is contained in the other. McConnell also proved that one can decide in linear time if two adjacent vertices corresponding arcs together cover the circle. The process starts with acquiring the list of pairs of vertices such that the union of their neighbourhoods cover the vertices of graph. Eschen [16] proved the time complexity of finding the neighbourhood disjointness in chordal bipartite graphs is $O(n_1n_2)$ where n_1 and n_2 are the sizes of each partitions. If two vertices neighbourhoods cover the graph, the complement of each vertex's neighbourhood should be disjoint. For every pair of adjacent vertices x and y that their neighbourhood cover the graph, McConnell explained how to find whether or not neighbourhood of every non neighbour of x is contained in y and vice versa is linear using so-called probe interval graphs. The detailed algorithm can be found in [49].

Now that we can recognize neighbourhood containments and cover the circle relations in linear time, we can start constructing the circular arc representation of G.

Creating a Circular Arc Model

Tucker introduced three different cases of graphs and for each case he proposed a different algorithm. Spinrad used the same classification of graphs. Case I are the co-bipartite graphs. These graphs are covered by two cliques, in other words, their complements are bipartite. Case IIa consists of graphs with a K_3 in their complement , and Case IIb of graphs with an induced C_{2k+1} in their complement with $k > 1$.

Here we present Eschen-Spinrad's algorithm for Case I and discuss Kaplan's and Nussbaum's analysis of the other cases.

Case I: Co-Bipartite Graphs

Suppose G is co-bipartite graph. Thus G is covered by two cliques. Since the size of one of them is at least $n/2$, G has $\Theta(n^2)$ edges. The Eschen-Spinrad algorithm will run in $O(n^2)$ time for graphs, but in this case, it will run in time linear with respect to the size of G, $O(m + n) = O(n^2)$.

Spinrad [60] introduced an algorithm to recognize a co-bipartite circular arc graph by reducing the problem to recognizing so-called two dimensional partial order. Here we discuss the reduction.

Let G be a co-bipartite graph which covered by two cliques X and Y. Assume G is a circular arc graph, the following theorem will be useful for discussing the circular arc representation of G.

Theorem 2.5.4. *[64] If G is co-bipartite circular arc graph with two cliques X and Y then there exist a circular arc representation of G with two points N, S such that every arc corresponding to a vertex in X cover S and not N and every arc corresponding to a vertex in Y cover N and not S.*

Recall that we usually call these two points on a circular arc representation of co-bipartite G the north pole and the south pole (cf. the proof of Theorem 1.3.8). These two poles partition the circle into two parts. We will again denote the clockwise portion of the circle from N to S, the *east side* and the counter-clockwise portion from N to S the *west side*. Each arc in the representation has exactly one endpoint in each side, more specifically every arc corresponding a vertex in X (Y) has a clockwise extreme endpoint in the east (west) side and a counter-clockwise extreme endpoint in the west (east) side. The circular arc representation of G will define two orderings of endpoints according to each side.

Definition 2.5.2. A *two dimensional partial order* is a combination of two linear orderings $<$ and $<'$ of the same objects. Let x and y be two objects then x precedes y in the two dimensional partial order if and only if $x < y$ and $x <' y$.

A directed graph G' with same vertex set of G will represent a two dimensional partial order with the following edges. The edge xy is an edge in $E(G)$ if and only if x precedes y in the two dimensional partial order, i.e., x precedes y in both orderings. If x precedes y in one ordering while y precede x in the other, x and y are non-adjacent.

Let R be a normalized circular arc representation of G. We define a two dimensional partial order of vertices of G based on the two orderings of the endpoints from the north pole to the south pole on the east side and the west side of R. Let v_1 be a vertex in X with the corresponding arc A_1 in R. For each vertex v_2 of G with the corresponding arc A_2 in R, the relation between A_1 and A_2 defines a two dimensional partial order of $V(G)$.

1 If A_1 and A_2 are disjoint then v_1v_2 is an edge in G'.

2 If A_1 contains A_2 then v_1v_2 is an edge in G'.

3 If A_1 is contained in A_2 then v_2v_1 is an edge in G'.

4 If A_1 and A_2 cover the circle then v_2v_1 is an edge in G'.

5 If A_1 overlaps A_2 then there is no edge between v_1 and v_2 in G'.

We create the directed graph G' that represents the two dimensional partial order as follows. The vertex set of G' is same as G. For each pair of vertices x_1 and x_2 in X, the edge x_1x_2 is in G', if neighbourhood of x_1 is contained in the neighbourhood of x_2. For each pair of y_1 and y_2 in Y, the edge y_1y_2 is in G', if the neighbourhood of y_2 is contained in the neighbourhood of y_1. For each $x \in X$ and $y \in Y$, an edge xy is in G', if x and y are non-adjacent. The edge yx is in G', if A_x and A_y, the arcs correspond to x and y, cover the circle. It is easy to see that there is a one-to-one correspondence between a normalized circular arc representation and G'.

Spinrad [62] showed that determining whether a directed graph is representing a two dimensional partial order will take $O(n^2)$ time. Feeding G' to Spinrad's two dimensional partial order recognizer, we can determine whether or not G is co-bipartite circular arc graph in time $O(n + m) = O(n^2)$.

Case *II*

Here we suppose that the graph G is not co-bipartite. Suppose v_0 is a minimum degree vertex in G. If $|N[v_0]| \geq n/2$ then $|E(G)| = \Theta(n^2)$ and the Eschen-Spinrad algorithm will run in time linear in the size of the graph. Otherwise, let M denote the set of non-neighbours of v_0, i.e., $M = V(G) \setminus N[v_0]$. Note that we assume $|M| > n/2$. Let v_1 be a vertex of M with the minimum degree in G. If $deg(v_1) \geq n/2$ then each vertex in M is adjacent to at least $n/2$ vertices. Thus, $|E(G)| = \Theta(n^2)$ and the Eschen-Spinrad algorithm will run in linear time with respect to size of the graph, i.e. $O(n + m)$. Otherwise, there is a vertex v_2 of M which is non-adjacent to both v_0 and v_1. We choose $v_2 \in M$ among the vertices non-adjacent to v_1 with the minimum degree in G.

Here we discuss how to recognize a circular arc graph by creating a circular arc representation, if G has at least three independent vertices. We briefly overview the steps of the recognition and more details can be found in [38] [16] [52].

We start by building a maximal independent set I based on the vertices v_0, v_1 and v_2. Since we chose these vertices with the minimum degree, their corresponding arcs are not contained any other arc. We construct I by choosing the next independent vertex with minimum degree in G. If no other vertex can be added to I then we verify the following

condition. If two non-adjacent vertices u, v are adjacent to exactly one vertex w in I, we replace w by u and v. Also if there are more than two non-adjacent vertices such that they are only adjacent to w in I then we reject G as a circular arc graph. Because no arc is contained in A_w, the vertex w has at most two non-adjacent neighbours such that the the arc corresponding to each covers a different endpoint of A_i.

After the maximal independent set I is created, we arrange the arcs corresponding to vertices in I around the circle. Let $J = V(G) \setminus I$ and A_I the set of the arcs corresponding to vertices in I, i.e., $A_I = \{I_i | I_i \textit{ is the corresponding arc for } v_i\}$. To find a circular ordering for A_I, we may use the following.

Lemma 2.5.5. *[64] If G is circular arc graph, there exists a circular arc representation R for G in which the circular ordering of A_I satisfies the following:*

- *For each vertex $v \in J$ with the corresponding arc A_v, the arcs corresponding to vertices in $N[v] \cap I$ are ordered consecutively around the circle. More precisely, A_v overlaps at most two arcs A_1 and A_2 in A_I. Then A_1 and A_2 are at the ends of consecutive ordering of $N[v] \cap I$ and the arcs contained in A_v are consecutive in the consecutive ordering of $N[v] \cap I$.*

- *For each pair of adjacent vertices u and v in J such that $(N(v) \cap N(u)) \cap I = \emptyset$, the arcs contained in either A_v or A_u are consecutive around the circle.*

To determine a circular ordering of A_I with the above conditions, Kaplan and Nussbaum construct an auxiliary $0, 1$-matrix M. The columns of M are corresponding to arcs in A_I. For the first condition, we add a row for every vertex v in J with entry "1" for columns corresponding to each A_i in A_I such that the arc corresponding to v contains A_i. We also add two extra rows with entry 1 in columns corresponding to the arcs in $A_i \cup \{A_i | A_i$ is contained in $A_v\}$ for each $i = 1, 2$. If G is circular arc graph then there is a circular ordering of the columns such that the entry 1s in each row are consecutive. This property of the $0, 1$-matrix M is called *circular ones property of M* and can be verified in time linear in the number of entries "1" in M [36], i.e., $O(m + n)$.

Assume the ordering of the independent arcs found in the previous step. Let $p = |I|$. By putting the arcs in A_I around the circle, $2p$ sections are created on the circle. Specifically, S_{2i} for each arc $A_i \in A_I$, and S_{2i+1} for the gap between two consecutive arcs A_i and A_{i+1}. We denote by S_i the ith section. The arc A_x corresponding to $x \in J$ in a circular representation of G has at most one endpoint in each section, since any arc in A_I contains no other arc. Using the types of relation between the arc A_x and the arcs in A_I we can determine the section for each of the endpoints. Consider the following cases.

- A_x overlaps just an arc A_i in A_I.
- A_x contains just an arc A_i in A_I. In this case, we put ℓ_x in S_{2i-1}, and r_x in S_{2i+1}.
- A_x overlaps two consecutive arcs A_i and A_{i+1} in A_I. In this case, we put ℓ_x in S_{2i} and r_x in S_{2i+2}.
- A_x does not intersect all arcs in A_I. In this case, we can determine which arc in A_I it meets first or which arc it meets last using the ordering we acquired in the previous step, and determine its endpoint's sections depending on whether it contains or overlaps these extreme arcs.
- A_x intersects all arcs in A_I and only overlaps one arc A_i in A_I.

This classification covers any possible relation between arcs in a circular arc representation of G. Except in the first and the last case, in all other cases the section of the endpoints are clear. For the case where A_x is either contains all other arcs in A_I and overlaps one or just overlaps one, full details can be found in [16] [38].

Finally, the last step is to arrange the endpoints in each section S_i with more than one endpoint. Note that the arrangement of the endpoints in S_i only affect the adjacency between the vertices corresponding to the arcs with an endpoint in S_i. The set of vertices corresponding to any circular arc with the left endpoint in S_i is a clique in G, since their corresponding circular arcs cover an endpoint of the circular arc $A_{i/2}$. Similarly, the set of vertices corresponding to any circular arc with the right endpoint in S_i is a clique in G. Let D_i denote the subgraph on the set of vertices with any endpoints in S_i, and $n_i = |D_i|$. The graph D_i is covered with two cliques, thus D_i is co-bipartite, and it has at least $\frac{n_i}{2}(\frac{n_i}{2} - 1)$ edges. We split the endpoints in each S_i into ordered disjoint subsections $S_{i,1}, S_{i,2}, .., S_{i,k}$, so that for each $S_{i,j}$, there is a section S_l contains the other endpoints of the arcs with an endpoint in $S_{i,j}$. Nussbaum described how this can be done in $O(n_i^2)$ [52]. For each subsection $S_{i,j}$ containing more that one endpoint, we determine the circular arc representation of an induced subgraph $D_{i,j}$ of D_i on the vertices with an endpoint in $S_{i,j}$. The graph $D_{i,j}$ is also covered with two cliques, thus we can determine a normalized circular arc representation for $D_{i,j}$ using the Case I of the algorithm. Since $O(\Sigma n_i^2) = O(m)$, the algorithm runs in time linear with respect to the size of G.

Chapter 3

Co-Circular Arc Graphs

In the previous chapter, we discussed the recognition of circular arc graphs in linear time. An interesting question that arises is the following. Given a graph G, can we recognize whether or not the complement of G is a circular arc graph. The question is particularly interesting if G has very few edges. For example, consider a bipartite graph G. Since the co-bipartite graph $\overline{G}$ is covered with two cliques, it contains $\Theta(n^2)$ edges. The linear time circular arc recognition algorithm from the previous chapter will take $\Theta(n^2)$ steps. But can we recognize if G is a co-circular arc graph in time linear with respect to the size of G (which could have fewer than $\Theta(n^2)$ edges)? We discuss co-bipartite graphs in Chapter 4.

In this chapter we study the following problem. Given a graph H, we want to find out if $\overline{H}$ is a circular arc graph. Using a direct approach to find a circular arc representation for $\overline{H}$ by the algorithm from the previous chapter will take $\Theta(n^2)$ time.

The close relationship of the bipartite co-CA graphs and interval graphs (see Theorems 1.3.4 and 1.3.12) and the fact that interval graphs can be recognized in linear time suggests that the answer to the question could be "yes".

One of our early observations was when H has $\Theta(n^2)$ edges, the basic algorithm will run in linear time, $O(n + m)$. This fact suggests we should look at graphs with small number of edges, like trees, k-trees and bounded degree graphs.

The results in this chapter are new, unless stated otherwise.

3.1 Recognition of Co-CA Trees

Trees have a linear number of edges (in terms of n). In this section, we assume H is a tree. We determine whether or not the complement of H is a circular arc graph. Since H is a tree,

H is bipartite. Thus, we may use Trotter's and Moore's classification of forbidden structures for bipartite co-CA graphs [37]. The following theorem will characterize the intersection of two graph classes: trees and co-CA graphs. The graph G_1 is described in Figure 3.1.

Theorem 3.1.1. *Let H be a tree. Then the complement of H is a circular arc graph if and only if G_1 is not an induced subgraph of H.*

Proof. Let H be a co-circular arc tree and R a circular arc representation of $\bar{H}$. Suppose H contains G_1 as an induced subgraph. We use the labels from Figure 3.1. An arc a_i corresponds to the vertex v_i. Consider the set, $S = \{a_2, a_5, a_8\}$. The arcs in S pairwise intersect, and each arc in S avoids a_1. Thus, the arcs in S share a point on the circle. Each arc a_i in S avoids the arc a_{i+1} intersecting the other two ones from S, thus no two arcs in S are contained in each other. We may assume the endpoints of the arcs in S are ordered and some $a_m \in S$ is contained in $\bigcup_{a_i \in S} a_i$. The arc a_{m+1} avoids a_m but intersects the arcs in $S \setminus a_m$, thus a_{m+1} contains a_1. The arc a_{m+2} avoids a_{m+1} but intersects a_1. This contradicts the containment relation between a_1 and a_{m+1}. Thus, H does not contain G_1 as an induced subgraph.

In Trotter's and Moore's characterization of bipartite co-circular arc graphs, the graph G_1 is the only forbidden tree structure. Thus, if G_1 is not an induced subgraph of H, the complement of H is a circular arc graph. □

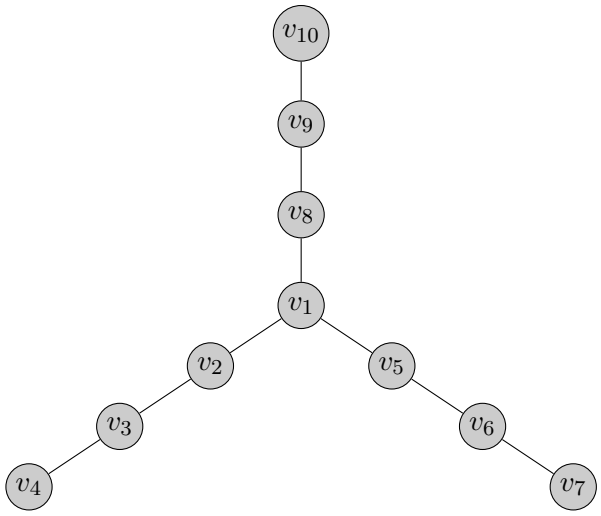


Figure 3.1: Graph G_1

Now that the only obstruction is described, we introduce a linear time algorithm to find G_1 in a given tree, if it is present.

INPUT: A tree T.
QUERY: Does T contain an induced G_1?

Algorithm

Let T be a tree. Our algorithm checks for every node, whether or not it can play the role of the vertex v_1 in a subgraph of T isomorphic to G_1.

First, we root T from an arbitrary vertex r_T in T. Throughout the algorithm, for each node x in T, we define a three-dimensional vector $v(x) = (v(x)_1, v(x)_2, v(x)_3)$ such that $v(x)_1 \geq v(x)_2 \geq v(x)_3 \geq 0$. Assume there is an orientation on edges from parents to children. The vector $v(x)$ encodes the lengths of the (up to) three longest directed paths from x through its distinct children. Specifically, we define, and compute the vector $v(x)$ as follows:

1 If u is a leaf then $v(u) = (0, 0, 0)$.

2 If u is a node with one child x then $v(u) = (v(x)_1 + 1, 0, 0)$.

3 If u is a node with two children x and y then $v(u) = (v(x)_1 + 1, v(y)_1 + 1, 0)$ (assuming $v(x)_1 \geq v(y)_1$, and similarly if $v(y)_1 > v(x)_1$).

4 If u is a node with more than two children then $v(u) = (v(x)_1 + 1, v(y)_1 + 1, v(z)_1 + 1)$ when x, y and z are children of u such that three longest paths among children of u starts from x, y and z, and $v(x)_1 \geq v(y)_1 \geq v(z)_1$.

The leaves of T do not have any children, thus we initialized them with $(0, 0, 0)$. We start with a queue Q of nodes initialized with the leaves of T. In each step, we dequeue a node u from Q and update $v(parent(v))$. After the update, we enqueue $parent(v)$ to S.

The tree T contains G_1 as an induced subgraph, if one of the following conditions is satisfied for some node u in T.

- $v(u)_1 \geq 3$ and $v(u)_2 \geq 3$ and $v(u)_3 \geq 3$.
- $v(u)_1 \geq 3$ and $v(u)_2 \geq 3$ and $v(parent(u))_2 \geq 2$
- $v(u)_1 \geq 3$ and $v(u)_2 \geq 3$ and $v(parent(parent(u)))_2 \geq 1$
- $v(u)_1 \geq 3$ and $v(u)_2 \geq 3$ and $parent(parent(v))$ has a parent.

The following theorem proves the correctness of the algorithm.

Theorem 3.1.2. *The class of co-CA trees can be recognized in linear time.*

Proof. Let T be a given tree. We may apply the algorithm above to T. We show that the algorithm is correct and runs in linear time. It is clear that for a node v if one of the conditions was satisfied then T contains an induced G_1. Thus due to Theorem 3.1.1, T is not a co-CA tree.

Now suppose T is not a co-CA tree, we show that one of the conditions will be satisfied. Due to Theorem 3.1.1, T must contains an induced G_1. Consider v_1 in G_1, either v_1 is the parent of its three neighbours in G_1 or v_1 is the child of one and the parent of the other two neighbours. The former case satisfies the first condition, and the latter case satisfies one of the other conditions depending on parent-child relations of $parent(v_1)$ with its other neighbour.

The algorithm runs for $\Theta(n)$ steps and each step takes constant time, also finding the set of leaves for a tree takes linear time. Thus, the algorithm recognizes a co-CA tree in linear time. □

Note that the same forbidden structure will characterize co-CA forests. Let F be a forest. We can apply the algorithm to each connected component of F.

Corollary. *The class of co-CA forests can be recognized in linear time.*

3.2 Recognition of Co-CA k-Trees

In the previous section, we discussed recognition of co-CA trees. Here, we expand our results to another class of graphs, namely k-trees.

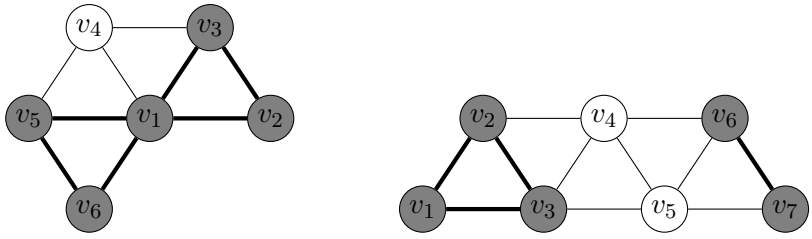
Definition 3.2.1. A graph G is a *k-tree* if either G is a K_{k+1} or if the graph resulting from eliminating a vertex whose neighbours induced a complete graph on k vertices, is a k-tree.

According to this recursive definition of k-tree, we may consider the construction process of a k-tree, G. In the construction process, we start from a complete graph on k vertices. By adding a vertex adjacent to a complete subgraph of size k in G, the resulting graph is also a k-tree.

A 1-tree is a tree. We recognized the class of co-CA trees in the previous section. Here we start by describing the recognition of co-CA 2-trees.

We define the following two graph classes of 2-trees which are useful for characterizing 2-tree co-circular arc graphs.

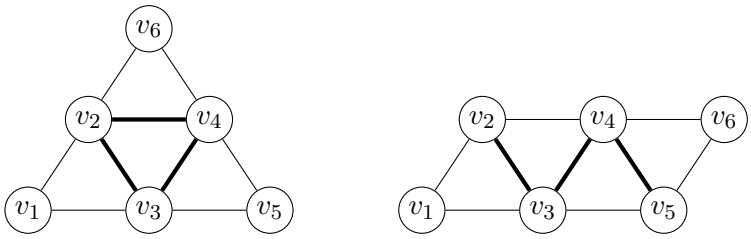
Definition 3.2.2. Consider the graphs A and B in Figure 3.3.



(a) Graph X (b) Graph Y

Figure 3.2: The forbidden structures of co-CA 2-trees

- A-class: all 2-trees obtained from the graph A (or an induced subgraph of A) by adding any number of vertices adjacent to any of the marked edges.
- B-class: all 2-trees obtained from the graph B (or an induced subgraph of B) by adding any number of vertices adjacent to any of the marked edges.



(a) Graph A (b) Graph B

Figure 3.3: Graph A and B

Lemma 3.2.1. *Let G be a 2-tree. If G does not contain X or Y from Figure 3.2 as an induced subgraph, then G is a graph from the A-class or the B-class.*

Proof. Let G be a 2-tree which does not contain an induced copy of X or Y. Consider the construction process of G, where we construct a changing graph G' along each step of the construction process. We initialize G' with K_3. In each step of the process, an edge e is chosen from G' and a number of vertices adjacent to e are added to G'. Without loss of generality, we assume vertices adjacent to the same edge are added consecutively. After two steps, the resulting graph is isomorphic to a subgraph on vertices $\{v_1, v_2, v_3, v_4, v_5\}$ from

A (or B), with marked edges v_2v_3 and v_3v_4 in Figure 3.3. Now, the construction process continues. For the next step, a vertex u' is added to H' such that u' is adjacent to an edge e' from H'. If $e' = v_2v_4$ then G' is from the A-class, if $e' = v_4v_5$ or $e' = v_1v_2$ then G' is from the B-class. If $e' = v_1v_3$ or $e' = v_3v_5$ then it is clear that G' would contain an induced subgraph X.

If G' is from the A-class then adding a vertex adjacent to an unmarked edge induces a subgraph X in G'. If G' is from the B-class then adding a vertex adjacent to v_1v_2 or v_4v_5 induces a subgraph Y in G'. Also adding a vertex adjacent to v_1v_3, v_3v_5, or v_2v_4 induces a subgraph X in G'. It is clear that adding vertices adjacent to a marked edge does not create an induced copy of X or Y in G'. (See Figure 3.3). The construction process of G may continue with vertices adjacent to the marked edges. Thus G is an induced subgraph of a graph from the A-class or the B-class, if G does not contain an induced subgraph of X or Y.

□

The following result gives several equivalent characterizations of 2-tree co-CA graphs. It also implies a linear time recognition algorithm, by considering the sequence of vertex additions for constructing the 2-tree.

Theorem 3.2.2. *The following are equivalent for a 2-tree G.*

1 G is a co-CA graph.

2 There is no induced $K_{2,3}$ or C_4^* in $\overline{G}$.

3 There is no induced X or Y in G.

4 G is from the A-class or B-class.

5 There is no induced P_5 in G.

Proof. 1 $\Rightarrow$ 2. See Figure 2.2 in Section 2.4.

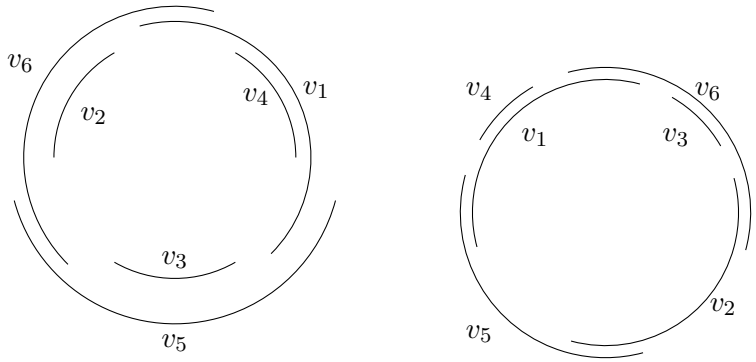
2 $\Rightarrow$ 3. See the marked vertices in Figure 3.2, X contains $\overline{C_4^*}$ and Y contains $\overline{K_{2,3}}$.

3 $\Rightarrow$ 4. Lemma 3.2.1.

4 $\Rightarrow$ 5. It is easy to check that a graph from A-class or B-class does not contain an induced P_5.

5 $\Rightarrow$ 3. Suppose G contains X or Y. Since both X and Y contain an induced P_5, there is an induced P_5 in G.

$4 \Rightarrow 1$. We show the complements of A and B are circular arc graphs by describing the circular arc representation of $\overline{A}$ and $\overline{B}$; this is done in Figure 3.4. A graph G from the A-class or the B-class, has sets of independent vertices only adjacent to a marked edge. There are three marked edges in each class. For each marked edge e, there already is a vertex v adjacent to e. Each independent set S_v of vertices only adjacent to e will be treated as a clique module with v in the complement of G. We describe how to arrange these sets according to the circular arc representation in Figure 3.4 to obtain a circular arc representation for $\overline{G}$. For each arc a_s corresponding to a vertex s in S_v, we arrange the endpoints of a_s next to endpoints of a_v so that a_s contains a_v. Thus a_u intersects a_v and the arcs intersecting a_v, and avoids the arcs avoiding a_v. Therefore, the circular arc corresponding to u is the same as the corresponding arc for v. □



(a) A (b) B

Figure 3.4: Circular arc representations of $\overline{A}$ and $\overline{B}$

The general case of k-trees is more difficult; we at least consider 3-trees. Suppose C is a graph in Figure 3.5, and let the C-class consist of graphs obtained by adding any number of vertices adjacent to any of the marked copies of K_3 in graph C, or in an induced subgraph of C (see Figure 3.5). The following two lemmas will describe the class of 3-tree co-circular arc graphs.

Lemma 3.2.3. *If G is a 3-tree from the C-class then G is co-CA 3-tree.*

Proof. We show that C is a co-circular arc graph by describing a circular arc representation of $\overline{C}$; we do this in Figure 3.6. Let G be any graph from the C-class, it may contain a vertex u adjacent to one of the copies of K_3, $v_1v_2v_3$ or $v_1v_2v_5$. In the former case, the vertex u has

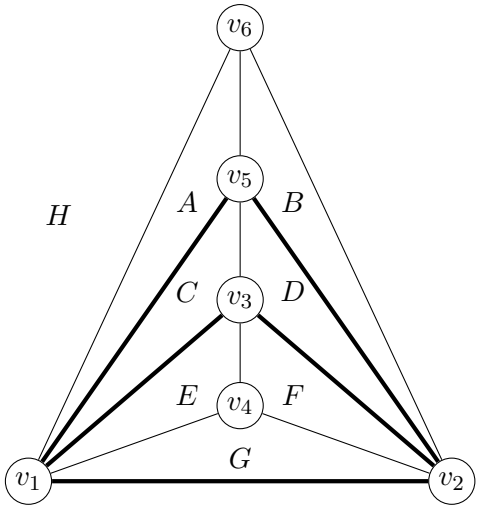


Figure 3.5: Graph C

the same neighbourhood as v_4, and in the latter case, u has the same neighbourhood as v_6. Thus, the corresponding arc for u is either the same as the corresponding arc for v_4, or the same as the corresponding arc for v_6. □

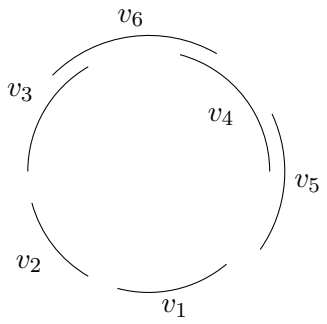
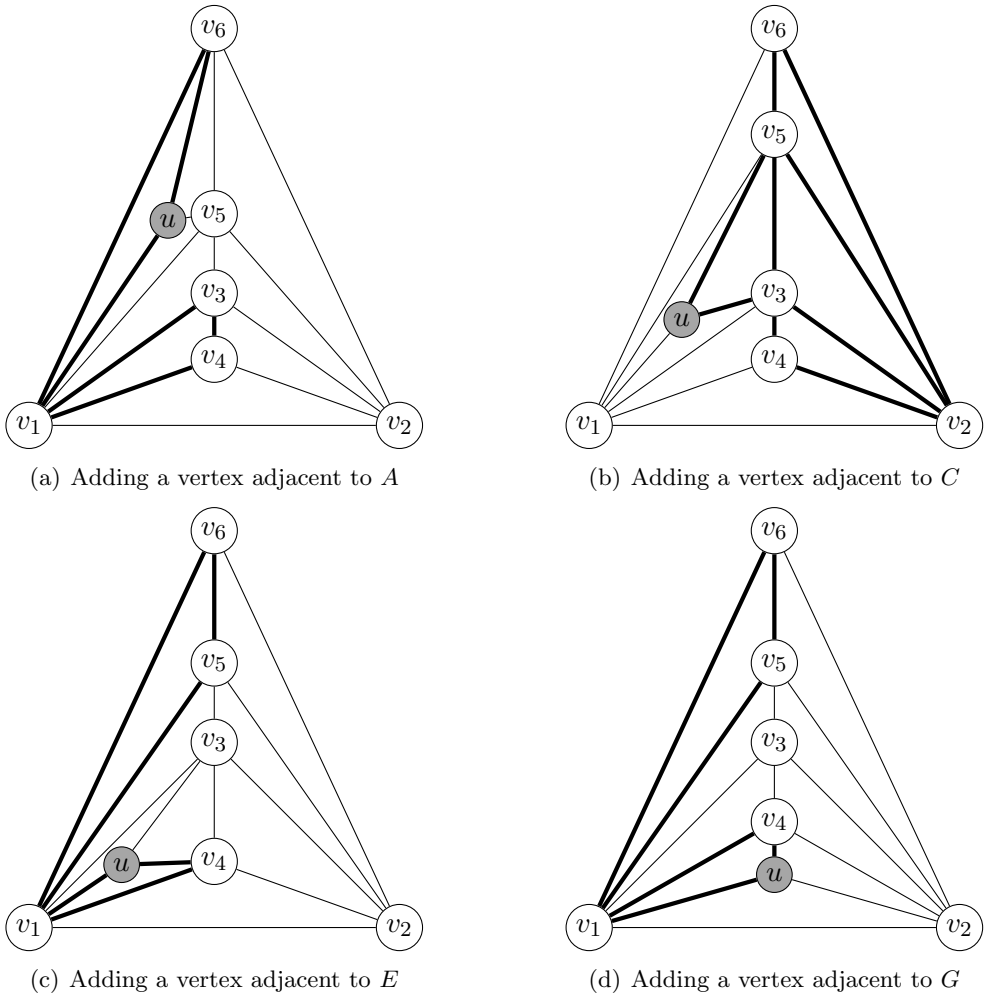


Figure 3.6: Circular arc representations of $\overline{C}$

Lemma 3.2.4. *Let G be any graph obtained by adding a vertex adjacent to any unmarked K_3 in the graph C. Then G is not a co-circular arc graph.*

Proof. Suppose we want to add a new vertex u to C. Let A, B, .. ,G, H denote the possible copies of K_3 for u to be adjacent to as in Figure3.5. Due to the symmetry of C, we only discuss A, C, E and G. In cases of A, E and G, an induced subgraph isomorphic to $\overline{C_4^*}$ is

marked in Figure 3.7(a) 3.7(c) 3.7(d), implying G is not a co-CA graph. In case of C, the marked edges in Figure 3.7(b) form an induced subgraph isomorphic to $\overline{2-net}$. The graph $2-net$ is a well known non-interval graph (See Figure 2.1(b)). The vertex v_1 is adjacent to every vertex of $\overline{2-net}$. Therefore, $\overline{2-net^*}$ is not a co-circular arc graph. □



(a) Adding a vertex adjacent to A

(b) Adding a vertex adjacent to C

(c) Adding a vertex adjacent to E

(d) Adding a vertex adjacent to G

Figure 3.7: 3-trees obstruction graphs

We conclude with the following characterization of co-circular arc 3-trees.

Theorem 3.2.5. *If G is a 3-tree, then G is a co-CA 3-tree if and only if G is a graph from the C-class.*

Proof. Let G be a 3-tree. Consider the construction process of G. We apply each step of the construction process to a graph G' to obtain G. We initialize G' with K_4. After

the first step, G' is consists of at least two independent vertices adjacent to a K_3. The construction may continue with adding a vertex adjacent to a new K_3. The new K_3 has exactly one vertex from the independent set and two vertices from the first K_3. Thus, the resulting graph after 2 steps is from the C-class. Due to Lemma 3.2.4, if the construction process continues with adding vertices to an unmarked K_3, then H', and hence H, is not a co-CA graph. It is also easy to see that every graph from the C-class is co-CA graph, by duplicating suitable arcs from Figure 3.6. □

It follows that 3-tree co-CA graphs can be recognized in linear time from their construction sequence.

3.3 Recognition of Co-CA Graphs with Bounded Degree

Trotter and Moore described forbidden structures of bipartite co-circular arc graphs. In this section we use their description to obtain an algorithm to recognize bipartite co-circular arc graphs with maximum degree 3.

Among the forbidden structures of bipartite co-circular arc graphs, the chordless cycles of length greater than 4 and the graphs T_1, G_1 and G_3 from Figure 3.8 are the only forbidden structures with degree bounded by 3.

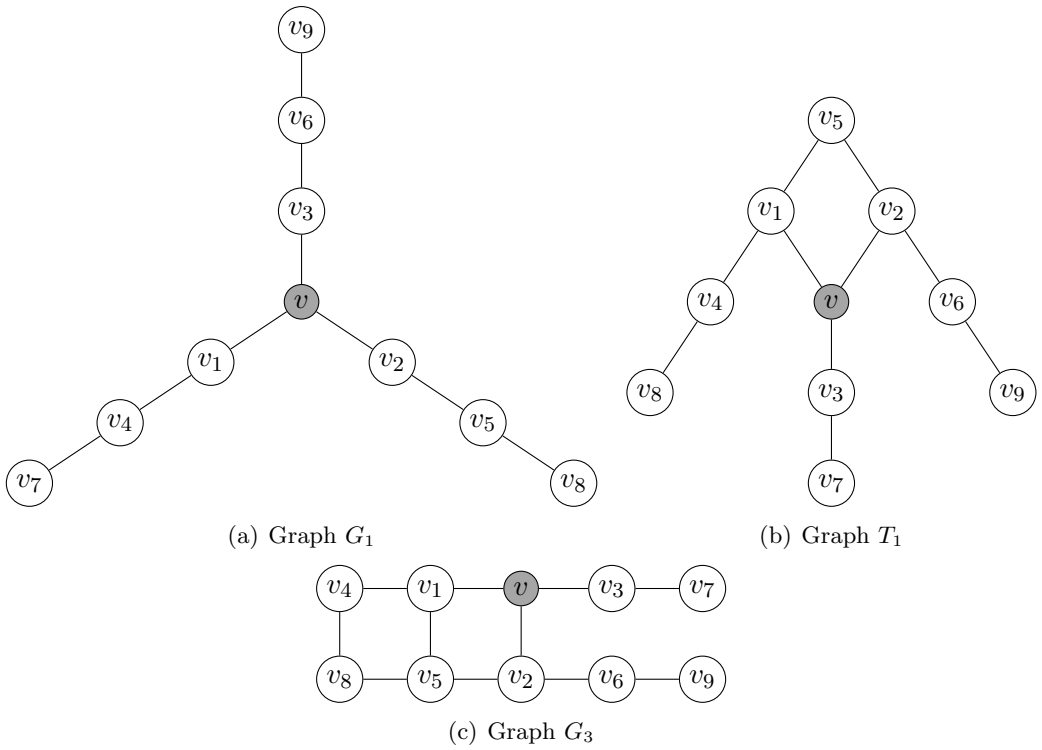
To guarantee that the given graph does not contain any induced cycle of length greater than 4, we use the recognition algorithm for chordal bipartite graphs. An elegant recognition algorithm for chordal bipartite graphs (see Definition 1.1.5) due to Lubiw [46] runs in $O(m \log n^2)$, and the time was improved by Paige and Tarjan [54] to $O(m \log n)$. In our case, $m \leq \frac{3n}{2}$ thus the Paige and Tarjan recognition will take $O(n \log n)$.

Next, we decide if a given graph G with maximum degree 3 contains an induced copy of T_1, G_1 or G_3.

```
INPUT: A bipartite graph G with degree bounded by 3.
QUERY: Is G a co-CA graph?

Algorithm
```

Suppose given G is a bipartite graph with degree bounded by 3. We test whether any vertex in G can play the role of the *marked vertex* (the center vertex) in Figure 3.8 for these forbidden structures.



(a) Graph G_1 (b) Graph T_1

(c) Graph G_3

Figure 3.8: The graphs T_1, G_1 and G_3

The maximum distance from the center vertex in all of these three obstruction graphs is 3. Since the degrees are bounded, there are bounded number of neighbours that we have to explore, and for this reason the rest of the algorithm will run in linear time.

Let v be a vertex in G with 3 neighbours. (Note that the center has 3 neighbours.) Each of the neighbours has at most 2 other neighbours. Let $L_1(v)$ denote the set of neighbours of v. Since G is bipartite no two vertices of $L_1(v)$ are adjacent. Let $L_2(v)$ denote the set of neighbours of the vertices in $L_1(v)$ different from v. Let $L_3(v)$ denote the set of neighbours of of vertices of L_2 not in L_1. No two vertices of $L_2(v)$ or $L_3(v)$ can be adjacent. Let $L_1(v)$ be $\{u_1, u_2, u_3\}$. For a vertex w in $L_2(v)$, we label w with the subset of $L_1(v)$ that w is adjacent to. For a vertex w in $L_3(v)$, we label w with the union of the labels from its neighbours in $L_2(v)$.

Now we check for graphs G_1, G_3 and T_1 as follows.

G_1 : G has G_1 as an induced subgraph, if and only if for a vertex $v \in V(G)$, $L_2(v)$ and $L_3(v)$ has 3 singly labelled vertices.

G_3 : G has G_3 as an induced subgraph, if and only if for a vertex $v \in V(G)$, $L_2(v)$ has 3 singly labelled vertices and a doubly labelled vertex with label, say $\{u_1, u_2\}$, and $L_3(v)$ has a doubly labelled vertex with label $\{u_1, u_2\}$, and a singly labelled vertex with label either $\{u_1\}$ or $\{u_2\}$.

T_1 : G has T_1 as an induced subgraph, if and only if for a vertex $v \in V(G)$, $L_2(v)$ has 3 singly labelled vertices and a doubly labelled vertex with label, say $\{u_1, u_2\}$, and $L_3(v)$ has two singly labelled vertices with labels $\{u_1\}$ and $\{u_2\}$.

It is easy to see that each singly labelled vertex in $L_i(v)$ is equivalent to a disjoint path of length i starting from v in G, thus G has G_1 as an induced subgraph with the center vertex v if and only if $L_3(v)$ has 3 singly labelled vertices.

If G has T_1 as induced subgraph then there are two paths of length 3 and a path of length 2 starting from the center vertex v. This is the reason $L_2(v)$ has 3 singly labelled vertices, and $L_3(v)$ has two singly labelled vertices with labels $\{u_1\}$ and $\{u_2\}$. Also $L_2(v)$ has a doubly labelled vertex with label $\{u_1, u_2\}$ which plays the role of v_5 in Figure 3.8(b).

If G contains G_3 then $L_2(v)$ has 3 singly labelled vertices, playing the roles of v_4, v_6 and v_7. Also $L_2(v)$ has a doubly labelled vertex w with label $\{u_1, u_2\}$ which plays the role of v_5 in Figure 3.8(c), and $L_3(v)$ has a doubly labelled vertex w' with the same label as w, so w' plays the role of v_8 in Figure 3.8(c). Note that if v_8 was non-adjacent to v_4 but was adjacent to v_2 then the resulting graph is still an induced G_3. Therefore, G has an induced G_3 if and only if the conditions above satisfied.

In conclusion, the recognition of bipartite co-circular arc graphs with degree bounded by three will take time $O(n \log n)$, with the bottleneck being finding induced cycles of length greater than four. If we know the input graph is chordal bipartite, then we have a linear time algorithm for determining whether the input graph is a co-CA graph.

Chapter 4

Bipartite Co-Circular Arc Graphs

In the study of list homomorphism problems for bipartite graphs, the class of bipartite complements of circular-arc graphs turned out to delineate the dichotomy (polynomial - NP-complete) for the problem (see Theorem 1.3.12). Furthermore, the complement of this class was singled out by Trotter and Moore as the central case for the recognition of circular arc graphs [37]. For this reason, we focus on the characterization of bipartite co-circular arc graphs. We also discuss solving various problems on this class of graphs. We call these graphs *bipartite co-CA* graphs.

In the previous chapter, we discussed the recognition of co-CA graphs in linear time for some classes of bipartite graphs namely, for trees, forests and bounded degree bipartite graphs. Now we investigate this class of graphs in general. We will also discuss some polynomial time algorithms on this class, for problems that are NP-complete in general, or problems that are polynomial time solvable in general, but can be solved more efficiently for this class. These results are new, unless stated otherwise. In some cases, we give a new proof to an existing result (cf. Theorem 4.1.3).

4.1 Characterization

We call G a *bipartite co-CA* graph, if G is bipartite graph with parts X and Y, and $\overline{G}$ has a circular arc representation, say R. Since G is bipartite, $\overline{G}$ is covered with two cliques on X and Y. The circular arc representation, R is representing $\overline{G}$, thus in R two arcs are disjoint if and only if their corresponding vertices in G are adjacent.

This class has various characterizations. The earliest characterization of the class belongs to Trotter and Moore [37]: they proved that a graph G is a bipartite co-CA graph if and

only if G does not contain an induced cycle of length greater than 4 or an induced subgraph from the list in Figure 4.1.

The following definitions are useful.

Definition 4.1.1. The *line graph* $L(G)$ of G is a graph on $E(G)$ such that e_1 and e_2 in $E(G)$ are adjacent in $L(G)$ if and only if e_1 and e_2 share a common vertex in G.

Definition 4.1.2. Let G be a graph. The *square of* G denoted by G^2 is the graph on $V(G)$ such that two vertices are adjacent if and only if their distance is at most two in G.

We may use the following results in future discussions. (See Definitions 1.1.5 and 1.1.4)

Lemma 4.1.1. *A bipartite co-CA graph G is chordal bipartite.*

Proof. A bipartite co-CA G can not contain any induced cycles of length greater than four, according to [37]. Thus, G is chordal bipartite. □

Lemma 4.1.2. *Let G be a bipartite co-CA graph. Then $L^2(G)$ is weakly chordal.*

This follows directly from Lemma 4.1.1 in view of the fact that $L^2(G)$ of a chordal bipartite graph G is weakly chordal [7].

We first state the following theorem which summarizes different equivalent characterizations of the class. (See Definitions 1.2.1, 1.3.2 and 1.4.3.)

Theorem 4.1.3. *([29] [37] [18] [57]) Let H be any graph. The following are equivalent:*

1 H is a bipartite co-circular arc graph.

2 H admits a min ordering.

3 H has no induced cycles of length greater than four and no edge asteroid.

4 H has no invertible pairs.

5 H has no induced subgraph from Figure 4.1 and no induced cycles of length greater than 4.

6 H has a 2-directional orthogonal ray representation.

The concept of 2-directional orthogonal ray representation is explained below. We contribute a direct proof of the equivalence of the 6th statement and the 1st statement.

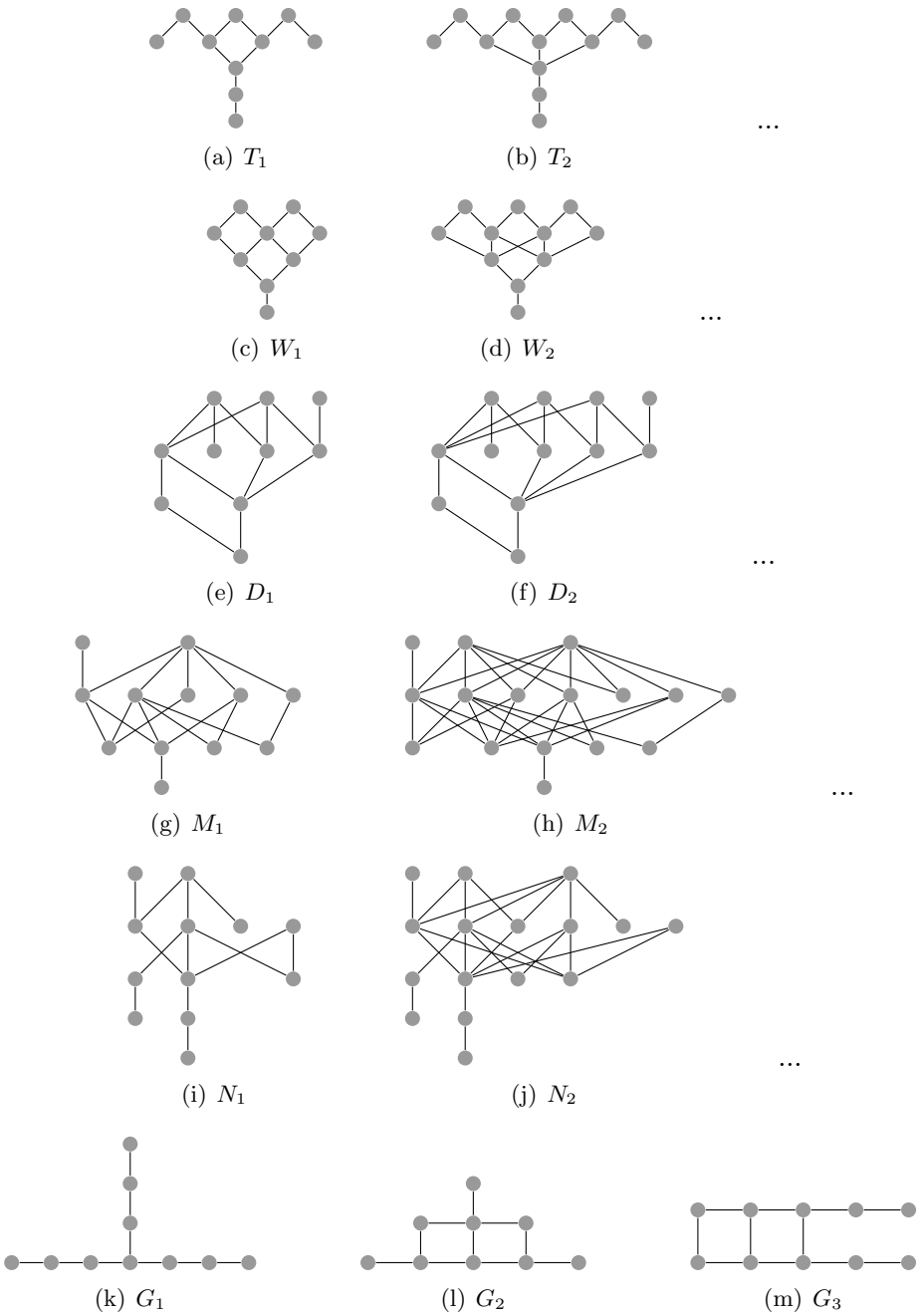


Figure 4.1: Forbidden structures of bipartite co-circular arc graphs.

4.1.1 Orthogonal Ray Graphs

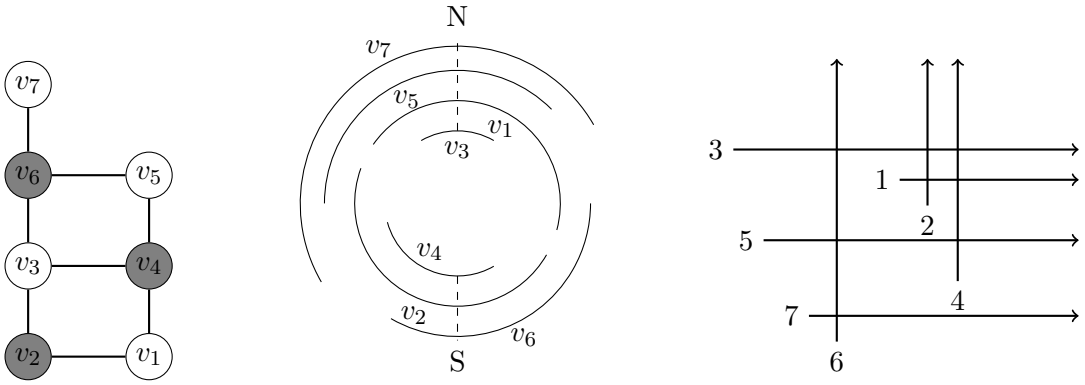
Here, we introduce another geometric characterization of bipartite co-CA graphs from [57].

Definition 4.1.3. A bipartite graph G with bipartition (X, Y) is called a *two directional orthogonal ray graph* if there exists a family of non-intersecting horizontal half-lines (rays) for every vertex in X and a family of non-intersecting vertical half-lines for every vertex in Y, such that for any $x \in X$ and $y \in Y$, $xy \in E(G)$ if and only if the horizontal ray of x and the vertical ray of y intersect.

This class of intersection graphs is also called the class of 2-directional orthogonal ray graphs [57]. The rays are illustrated on xy-plane in Figure 4.2. The following result clarifies the importance of this graph class.

Theorem 4.1.4. *[57] A bipartite graph G is a 2-directional orthogonal ray graph if and only if $\overline{G}$ is a circular arc graph.*

In [57], this correspondence is proved indirectly by the inability of representing edge-asteroids and even cycles of greater than four with 2-directional orthogonal rays (thus showing 6 is equivalent to 3 in Theorem 4.1.3). In this thesis, we present a direct proof of the equivalence of 6 and 1 of Theorem 4.1.3.



(a) Graph G (b) R_c of $\overline{G}$ (c) R_o of G

Figure 4.2: The correspondence between bipartite co-CA graphs and orthogonal ray graphs

Proof. Let G be a 2-directional orthogonal ray graph. We show there is a one-to-one correspondence between 2-directional orthogonal ray models of G and circular arc graph representations of $\overline{G}$. Let R_o be a 2-directional orthogonal ray model of G. We may assume no

two rays start with same vertical or horizontal position. If there exists a pair of such rays, we could slightly move one so that R_o still represent G.

We now describe the circular arc representation R_c for $\overline{G}$. Since G is bipartite with parts X and Y, $\overline{G}$ is covered with two cliques. Let N and S denote two arbitrary points on the circle in R_c. According to Theorem 2.5.4, there is a circular arc representation of $\overline{G}$ in which the northern arcs denote the arcs corresponding to the vertices in X cover N but not S and the southern arcs denote the arcs corresponding to vertices in Y cover S but not N. (These two points represent the north pole and the south pole respectively in R_c.) Each ray has an initial point (x, y). Horizontal rays keep the y fixed, and vertical ray keep x fixed. We sort all the initial points of the rays (regardless of type) according to the x coordinates. We call this the *horizontal ordering* of the rays. Also we sort all the initial points of the rays (regardless of type) according to the y coordinates and call this the *vertical ordering* of the rays. (Note that each ordering involves both horizontal and vertical rays.) We arrange the left endpoints of the northern arcs and the right endpoints of the southern arcs in the clockwise direction from N to S according to the horizontal ordering of the rays. For example, in Figure 4.2, the horizontal ordering of the rays is $\{3, 5, 7, 6, 1, 2, 4\}$ which yields the ordering $\{r_3, r_5, r_7, \ell_6, r_1, \ell_2, \ell_4\}$ of the endpoints in the clockwise direction from N to S. Similarly, in the clockwise direction from S to N, we arrange the right endpoints of northern arcs and the left endpoints of southern arcs according to the vertical ordering of rays. In the same example in Figure 4.2, the vertical ordering of the rays is $\{3, 1, 2, 5, 4, 7, 6\}$ which yields the ordering $\{\ell_3, \ell_1, r_2, \ell_5, r_4, \ell_7, r_6\}$ in the clockwise direction from N to S.

Let ray_1 be a horizontal ray in R_o corresponding to $v_1 \in X$ and ray_2 a vertical ray in R_o corresponding to $v_2 \in Y$. Then v_1 and v_2 are adjacent if and only if ray_1 precedes ray_2 in the horizontal ordering and ray_2 precedes ray_1 in the vertical ordering. The arc $a_1 = (\ell_1, r_1)$ corresponds to v_1 and the arc $a_2 = (\ell_2, r_2)$ corresponds v_2, in the circular arc representation R_c. Then r_1 precedes ℓ_2 in the clockwise direction from N to S, and r_2 precedes ℓ_1 in the clockwise direction from S to N if and only if a_1 and a_2 are disjoint.

Conversely, let G be a bipartite co-circular arc graph with parts X and Y. Suppose R_c is a circular arc representation of $\overline{G}$ with the north and south poles N and S such that the northern arcs corresponding vertices in X cover N but not S, and the southern arcs corresponding to vertices in Y cover S but not N. These poles divide the circle into two parts. We now describe a 2-directional ray representation R_o for G.

Consider an $n \times n$ grid in the xy-plane. The circular arc representation R_c involves $2n$ points (two endpoints for each circular arc). There are exactly n endpoints in each part of the circle, since each arc is either a northern arc or a southern arc. We order the x

coordinates of the initial points of rays according to the ordering of the endpoints of the arcs in the clockwise direction from N to S. Similarly, we order the y coordinates of the initial points of rays according to the ordering of the endpoints of the arcs in clockwise direction from S to N. Note that no two rays (regardless of type) share a column or a row. Every vertex in X (respectively Y) corresponding to a horizontal ray (respectively vertical ray).

Two vertices $v_1 \in X$ and $v_2 \in Y$ are adjacent in G if and only if the arcs $a_1 = (\ell_1, r_1)$ corresponding to v_1 and $a_2 = (\ell_2, r_2)$ corresponding to v_2 are disjoint. Since a_1 is a northern arc, and a_2 is a southern arc, r_1 precedes ℓ_2 in the clockwise direction from N to S, and r_2 precedes ℓ_1 in the clockwise direction from S to N. Let ray_1 with initial point (x_1, y_1) denote the horizontal ray corresponding to v_1 and ray_2 with initial point (x_2, y_2) denote the vertical ray corresponding to v_2. Thus according to the ordering of the rays x_1 precedes x_2, and y_1 precedes y_2 if and only if ray_1 and ray_2 intersect. □

It is an interesting fact that both the class of bipartite co-CA graphs and the class of their complements are intersection graphs, with different kinds of intersection representations.

Moreover, here we proved every 2-directional orthogonal ray model for a graph G exactly corresponds to a circular arc model for the complement of G. We observed that the horizontal ordering and the vertical ordering of the starting points of the rays define two partial order on vertices of G. Note that G represents a two dimensional partial order created from these two orderings. We mentioned how Spinrad used this fact to recognize bipartite co-CA graphs in Section 2.5.1.

4.2 Algorithms

In the previous sections, we studied the recognition of circular arc and co-circular arc graphs. Here we contribute polynomial time algorithms (and even linear time algorithms in some cases) on bipartite co-CA graphs and their complements, for a few well-known problems.

4.2.1 Colouring, Matching, and Covering

We discussed the graph k-colouring problem in Section 1.1 and showed how it can be interpreted as a list homomorphism problem. When $k > 2$, the graph k-colouring problem is NP-complete even when restricted to circular arc graphs [21].

The k-colouring problem is trivial for bipartite co-CA graphs, and we focus on the k-colouring problem for co-bipartite CA graphs. The graph k-colouring problem is equivalent

to the clique covering problem on the complement of the graph. In other words, we consider covering a bipartite co-CA graph with minimum number of cliques. Note that in a bipartite co-CA graph each clique is either K_1 or K_2.

Definition 4.2.1. A *matching* M for graph G is a set of edges in G such that no pair of edges in M share a vertex. A vertex does not belong to an edge of M is said to be *unmatched*.

The clique covering problem is exactly the maximum matching problem in bipartite graphs. Specifically, a bipartite graph G has a matching with k edges if and only if it has a covering with $n - k$ cliques. The maximum matching problem for bipartite graphs can be solved in time $O(n^{\frac{5}{2}})$ or $O(m\sqrt{n})$ [33].

We give a more efficient algorithm using the fact that co-CA graphs admit a min ordering. The problem we consider is this:

```
INPUT : A bipartite co-CA graph G with a given min ordering <.
QUERY : Find the maximum matching of G.
```

```
Algorithm
```

Suppose G is a bipartite co-circular arc graph with parts X and Y, and a given min ordering $<$. We construct a set of edges M_{alg} which is a matching on G. Initially $M_{alg} = \emptyset$. Let m_1 denote the last unmatched vertex from X in the min ordering. Let m_2 be the last unmatched neighbour of m_1 in the min ordering. We add the edge m_1m_2 to M_{alg}, if such a m_2 exists. Otherwise, we skip m_1. We stop when all the vertices in X is either matched or skipped.

Lemma 4.2.1. *The set M_{alg} is a maximum matching in G.*

Proof. Suppose M_1 is a maximum matching in G. Assume m_1m_2 is not in M_1. The edge m_1m_2 cannot be added to M_1. Thus at least one of m_1, m_2 is matched in M_1. If just one of m_1 or m_2 is matched in M_1, say $m_iv \in M_1$, then we replace m_iv by m_1m_2 and obtain a new maximum matching M_2 for G which includes m_1m_2. If both m_1 and m_2 are matched in M_1, then there are in M_1 edges m_1v_2 and m_2v_1 for some $v_1 \in X$ and $v_2 \in Y$. The vertex m_2 is the last neighbour of m_1 thus $v_2 < m_2$ and $v_1 < m_1$. Because of the min ordering and the existence of the edges m_2v_1 and m_1v_2 in G, the edge v_1v_2 is in G. Now, if we replace m_2v_1 and m_1v_2 by v_1v_2 and m_1m_2, we obtain a new maximum matching M_2 for G which includes m_1m_2.

Suppose that M_{alg} and M_i share i edges in i-th step of the algorithm. Now let $m'_1m'_2$ be the edge added to M_{alg} in the i-th step of the algorithm. Assume $m'_1m'_2$ is not in M_i. The edge $m'_1m'_2$ cannot be added to M_i. Thus again at least one of m'_1, m'_2 is matched in M_i. If just one of m'_1 or m'_2 is matched in M_i, say $m'_iv \in M_i$, then we replace m'_iv by m_1m_2 in M_i and obtain a new maximum matching M_{i+1} for G which includes $m'_1m'_2$. If both m'_1 and m'_2 are matched in M_i, then there are in M_i edges m'_1v_2 and $v_1m'_2$ for some $v_1 \in X$ and $v_2 \in Y$. Since m'_1 is the last unmatched vertex in the min ordering with at least one unmatched neighbour, every vertex $v'_1 > m'_1$ is either matched in M_i or does not have an unmatched neighbour in M_i. Thus, $v_1 < m'_1$. Also m'_2 is the last unmatched neighbour of m'_1 in the min ordering, thus $v_2 < m'_2$. Since edges m'_1v_2 and $v_1m'_2$ exist, the min ordering entails the existence of v_1v_2. Now, if we replace $v_1m'_2$ and m'_1v_2 by v_1v_2 and $m'_1m'_2$ in M_i, we get a new maximum matching M_{i+1} for G which includes $m'_1m'_2$. The M_{alg} has exactly $|M_1|$ edges and is therefore a maximum matching.

□

Theorem 4.2.2. *The maximum matching problem for a bipartite co-CA graph can be solved in linear time.*

Proof. Suppose a bipartite co-CA graph G with a min ordering on its vertices was given by its adjacency lists. We sort each adjacency list, using bucket sort, in linear time. Thus the algorithm above finds the last vertex in a min ordering, and finds the last neighbour of a vertex in constant time. Since each step of the algorithm match two vertices in G, the algorithm runs for $O(n)$ time. The bucket sort will take $O(n + m)$ where n is the number of vertices and m is the number of edges [10]. This shows that the algorithm above runs in $O(n + m)$. □

Moreover, the *Vertex Cover* problem asks for a minimum number of vertices in G such that every edge in G is incident to at least one of them. In bipartite graphs, due to König's Theorem [65], the size of the maximum matching is equal to size of the minimum vertex cover.

Corollary. *The Vertex Cover, Minimum Clique Cover and Maximum Matching for a bipartite co-CA graph G can be solved in linear time, provided a min ordering $<$ of G is given.*

Corollary. *The k-colouring problem for a co-bipartite CA graph G can be solved in linear time, provided a min ordering $<$ of G is given.*

Definition 4.2.2. An *induced matching* (IM) for graph G is a matching in G such that any two of its edges induce a copy of $2K_2$.

The *Maximum Induced Matching* (MIM) problem is defined as follows.

```
INPUT: A graph G.
QUERRY: Find the largest induced matching on G.
```

The MIM problem is NP-complete for general graphs [21], even for bipartite graphs with bounded degree 4, and for planar graphs [14]. Golumbic and Laskar showed that the MIM problem is polynomial time solvable for circular arc graphs and linear time solvable for interval graphs [23].

The MIM problem for co-bipartite CA graphs follows from the above known results, thus we focus on bipartite co-CA graphs.

Theorem 4.2.3. *[6] The size of the maximum induced matching for a graph G is exactly the size of the maximum independent set in $L^2(G)$.*

Since bipartite co-circular arc graph is chordal bipartite, by Theorem 4.1.1, the square of its line graph is weakly chordal. A polynomial time algorithm for finding a maximum independent set on weakly chordal graphs can be found in [7].

4.2.2 Oriented Chromatic Number

We discussed the k-colouring problem earlier. The minimum k such that a graph G has k-colouring is called the *chromatic number* $\chi(G)$. The chromatic number of a graph G is equivalently regarded as the minimum k such that $G \to K_k$. In other words, K_k has the minimum size vertex set among graphs to which G is homomorphic.

To extend the concept of k-colouring to digraphs, let us consider an irreflexive digraph H without opposite arcs (uv and vu are *opposite arcs* for u and v in $V(H)$). Such a digraph is called an *oriented graph*. The *oriented colouring* of H is a homomorphism of H to another oriented graph H'. An *oriented k-colouring* of H is a homomorphism to any H' with $|V(H')| = k$. Equivalently, an oriented k-colouring of H is a partition of $V(H)$ into k disjoint sets such that all edges between two sets are in a same direction and no two adjacent vertices belong to the same set. The *oriented chromatic number* $\chi_o(H)$ is the minimum number k such that H has an oriented k-colouring. Unlike k-colouring, H' does not need to contain all possible edges. Note that opposite edges in H violate the first condition, and loops in H violate the second condition, for an oriented colouring of H.

The concept of an oriented colouring was first introduced by B. Courcelle [12]. The oriented chromatic number was formally introduced by Sopena in [58]. One of the important applications of oriented colouring arose in link scheduling for a wireless sensor network [32]. Most of the results on oriented chromatic numbers construct bounds for different classes of graphs (see [53] [51]). Klostermeyer and MacGillivray studied the complexity of the oriented colouring in [41]. They showed the decision problem of whether H has an oriented k-colouring for $k \geq 4$ is NP-complete, and for $k \leq 3$ is polynomial time solvable. Culus and Demange in [13] showed this result is also true for bipartite digraphs. For more details, see a survey on oriented colouring [59].

In this thesis, we find the oriented chromatic number for bipartite co-CA oriented graphs with a given min ordering, and present a linear time algorithm for the task. Here we assume H is a digraph with a min ordering $<$ such that the underlying graph is bipartite co-CA. However we do not assume all edges go from one part of the bipartition to the other part, i.e., we do not just have two orderings on each part.

A directed edge uv in $E(H)$ is called an *increasing edge* (a *decreasing edge*), if $u < v$ ($v < u$) according to our fixed min ordering. Each edge in $E(H)$ is either an increasing edge or a decreasing edge. The following lemmas are useful for characterizing the structure of a bipartite oriented graph H with min ordering.

Lemma 4.2.4. *Let H be a bipartite digraph with a given min ordering $<$. If uv and vw are two edges of H then either both uv and vw are increasing edges or both are decreasing edges.*

Proof. Suppose two edges uv and vw are in $E(H)$. If uv is an increasing edge and vw is a decreasing edge then $u < v$ and $w < v$. Thus, the min ordering of H implies uw in H. The underlying K_3 on u,v and w violates the bipartiteness of H. Otherwise, if uv is a decreasing edge and vw is a decreasing edge then $v < u$ and $v < w$. Again, the min ordering of H implies v has a loop which contradicts our assumption. □

In an oriented graph H, an *oriented walk* is a sequence of vertices $v_0, v_1, ..., v_k$ such that v_{i-1} and v_i are adjacent in H for each $i = 1, 2, ..k$, i.e., either $v_{i-1}v_i$ or v_iv_{i-1} is in $E(H)$. An oriented walk is *closed* if v_0 and v_k are adjacent. An edge $v_{i-1}v_i$ in $E(H)$ is called a *forward edge* of the walk, and an edge v_iv_{i-1} in $E(H)$ is called a *backward edge* of the walk. The *net length* of a walk is the difference between the number of forward and backward edges in the walk.

Lemma 4.2.5. *Let H be an oriented bipartite graph which admits a min ordering. Then the net length of any closed walk of length 4 in H is zero.*

Proof. Let H be an oriented bipartite graph with a min ordering $<$, and W a closed walk of length four with the sequence of vertices $w_1w_2w_3w_4$ in H. Since W is a closed walk, we may assume w_1 is the first vertex in $<$. We assume $w_2 < w_4$ in the min ordering, otherwise consider the reversal oriented closed walk of W. It is easy to see that the net length of a closed walk and its reversal are opposite of each other. Equivalently, the net length of W is 0 if and only if the net length of the reversal of W is 0. By Lemma 4.2.4, either the edges w_1w_2 and w_1w_4 or the edges w_2w_1 and w_4w_1 are in $E(H)$. Assume the first case, then there are three possible cases for w_3.

1. $w_3 < w_2 < w_4$: By Lemma 4.2.4, w_3w_2 and w_3w_4 are in $E(H)$. So the net length of W is zero. (See Figure 4.3(a).)

2. $w_2 < w_3 < w_4$: By Lemma 4.2.4, w_3w_4 is in $E(H)$. If w_2w_3 is in $E(H)$ then the min ordering and the edges w_2w_3 and w_1w_{2k} imply the edge w_1w_3. Thus w_1, w_3 and w_2k induce a K_3 in the underlying graph of H, which contradicts the bipartiteness of H. So w_3w_2 is in $E(H)$ and the net length of W is zero. (See Figure 4.3(b).)

3. $w_2 < w_4 < w_3$: By Lemma 4.2.4, either w_3w_2 and w_3w_4 are in $E(H)$, or w_2w_3 and w_4w_3 are in $E(H)$. So the net length of W is zero. (See Figure 4.3(c).)

Now assume edges w_2w_1 and w_4w_1 are in $E(H)$. The proof is similar.

□

Lemma 4.2.6. *Let H be an oriented bipartite co-CA graph which admits a min ordering. Then the net length of every closed walk in H is zero.*

Proof. Since the underlying graph of H is a bipartite graph, it does not contain any odd cycle. Thus, every oriented closed walk of H is an even-length oriented closed walk. Assume W is the shortest oriented closed walk with vertices $w_1, w_2, ..., w_{2k}$ in H, such that the net length of W is not zero. If $w_i = w_j$ for some $i < j$ then the subsequence $w_i, w_{i+1}, .., w_{j-1}$ is an oriented closed walk W_s in H. Note that both W_s and $W - W_s$ are oriented closed walks in H. It is clear that the net length of W is the summation of the net lengths of W_s and $W - W_s$. Since the net length of W is not zero, so the net length of either W_s or $W - W_s$ is not zero. This contradicts our assumption that W is the shortest oriented walk with net length zero. Thus, $w_i \neq w_j$ for any i and j. This shows that an oriented walk W with length $2k$ has an underlying graph C_{2k} in H with possible chords.

The underlying graph of W is a C_{2k}. Since the underlying graph of H is a bipartite co-CA graph, every cycle of length greater than 4 has a chord. Thus, there is a subsequence of W

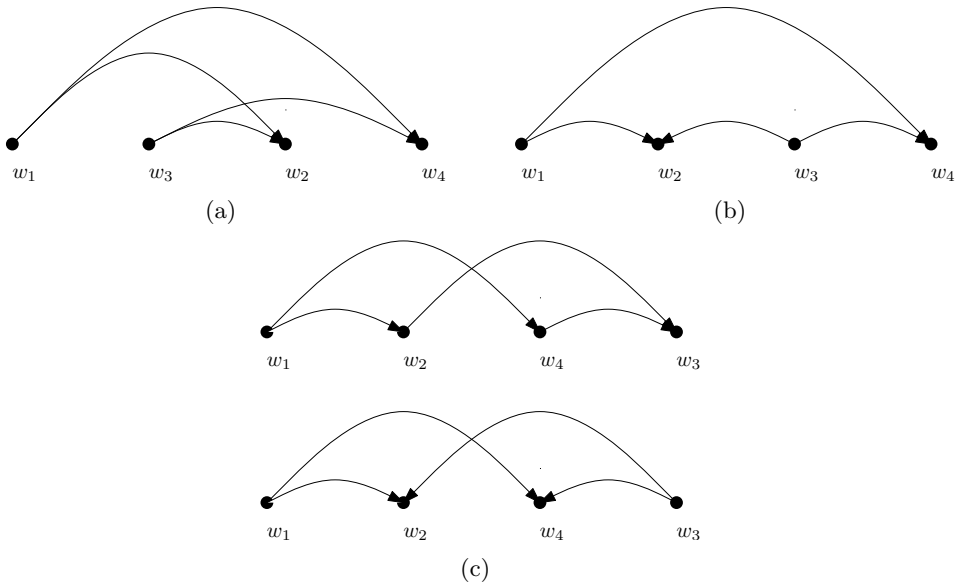


Figure 4.3: Different orientations of C_4 with a min ordering (from left to right)

with vertices $w_iw_{i+1}w_{i+2}w_{i+3}$ with a chord w_iw_{i+3}. We proved that the oriented closed walk C_4 with vertices $w_iw_{i+1}w_{i+2}w_{i+3}$ has the net length zero. Thus, the net length of the walk $w_iw_{i+1}w_{i+2}w_{i+3}$ is equal to the net length of the walk w_iw_{i+3}. Now in the shortest closed walk W with the net length zero, we may replace $w_iw_{i+1}w_{i+2}w_{i+3}$ by w_iw_{i+3} producing a closed walk W'. The oriented closed walks W' and W have the same net length. But W' has a smaller length, which contradicts the fact that W is the shortest oriented closed walk with non-zero net length.

□

Definition 4.2.3. An oriented graph G is *balanced*, if every closed walk in G has net length zero.

Lemma 4.2.7. *[27] A balanced oriented graph G has a homomorphism to a directed path $\overrightarrow{P_k}$ for some $k \geq 1$.*

Lemma 4.2.8. *A directed path of length k is homomorphic to the directed cycle $\overrightarrow{C_3}$.*

Proof. Let $\overrightarrow{P_k}$ be a directed path of length k with the vertices $v_1v_2..v_k$ such that v_i is adjacent to v_{i+1} for $i = 1, 2, .., k-1$, and $\overrightarrow{C_3}$ a directed cycle with the vertices $u_0u_1u_2$ such that u_0u_1, u_1u_2 and u_2u_0 are in $\overrightarrow{C_3}$. We map v_i to u_{imod3}. It is easy to see that the images of two

vertices v_i and v_{i+1}, namely $u_{i \mod 3}$ and $u_{(i+1) \mod 3}$, are adjacent. The mapping preserves the adjacency, thus $\overrightarrow{P_k}$ is homomorphic to $\overrightarrow{C_3}$. □

Corollary. *Let H be an oriented bipartite co-CA graph with a min ordering $<$ then H is homomorphic to $\overrightarrow{C_3}$ and $\chi_o(H) \leq 3$.*

We showed that $\chi_o(H) \leq 3$ for every oriented bipartite co-CA graph H with a min ordering $<$. Here we describe an algorithm for oriented colouring of H.

`Algorithm`

If H has no edge then $\chi_o(H) = 1$. Also, if H has no induced P_2 then $\chi_o(H) = 2$, since the start vertex of each edge of H can be coloured 1 and its end vertex is coloured 2. Otherwise, if H contains an induced P_2 then $\chi_o(H) = 3$. We describe the algorithm for oriented colouring of H with $\overrightarrow{C_3}$. We pick a vertex v of H. We can choose any arbitrary colour for v. If a vertex is coloured by i then we will colour the vertex u by $i + 1$ (modulo 3) if u is adjacent to v with an edge vu, and we colour u by $i - 1$ (modulo 3) if u is adjacent to v with an edge vu. It is easy to see that no vertex coloured with two different colours, since H is balanced.

Remark. The above algorithm is the same algorithm used for oriented colouring of trees [27].

Remark. Oriented chordal bipartite graphs with min ordering $<$ has an oriented 3-colouring.

Bibliography

[1] Seymour Benzer. On the topology of the genetic fine structure. *Proc Natl Acad Sci U S A.*, 45(11):1607 – 1620, 1959.

[2] Flavia Bonomo, Guillermo Durán, Luciano N. Grippo, and Martín Darío Safe. Partial characterizations of circular-arc graphs. *Journal of Graph Theory*, 61(4):289 – 306, 2009.

[3] Kellogg S. Booth. *Pq-tree algorithms.* PhD thesis, 1975.

[4] Kellogg S. Booth and George S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using pq-tree algorithms. *J. Comput. Syst. Sci.*, 13(3):335 – 379, 1976.

[5] Andrei A. Bulatov. Tractable conservative constraint satisfaction problems. In *Logic In Computer Science*, pages 321 – 330, 2003.

[6] Kathie Cameron. Induced matchings in intersection graphs. *Discrete Mathematics*, 278(1-3):1 – 9, 2004.

[7] Kathie Cameron, R. Sritharan, and Yingwen Tang. Finding a maximum induced matching in weakly chordal graphs. *Discrete Mathematics*, 266(13):133 – 142, 2003. The 18th British Combinatorial Conference.

[8] Jérémie Chalopin and Daniel Gonçalves. Every planar graph is the intersection graph of segments in the plane: extended abstract. In *41st ACM Symposium on Theory of Computing*, pages 631 – 638, 2009.

[9] Giuseppe Confessore, Paolo Dell'Olmo, and Stefano Giordani. An approximation result for a periodic allocation problem. *Discrete Applied Mathematics*, 112(13):53 – 72, 2001. Combinatorial Optimization Symposium, Selected Papers.

[10] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.

[11] Derek G. Corneil, Stephan Olariu, and Lorna Stewart. The ultimate interval graph recognition algorithm? (extended abstract). In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 175 – 180, 1998.

[12] Bruno Courcelle. The monadic second order logic of graphs vi: on several representations of graphs by relational structures. *Discrete Applied Mathematics*, 54(23):117 – 149, 1994.

[13] Jean-François Culus and Marc Demange. Oriented coloring: complexity and approximation. In *Proceedings of the 32nd conference on Current Trends in Theory and Practice of Computer Science*, pages 226 – 236, Berlin, Heidelberg, 2006. Springer-Verlag.

[14] W. Duckworth, D. Manlove, and M. Zito. On the approximability of the maximum induced matching problem. *Journal of Discrete Algorithms*, 3:79 – 91, 2000.

[15] Elaine M. Eschen and Jeremy Spinrad. An $o(n^2)$ algorithm for circular-arc graph recognition. In *SODA*, pages 128 – 137, 1993.

[16] E.M. Eschen. *Circular-arc Graph Recognition and Related Problems*. Vanderbilt University, 1997.

[17] Tomás Feder and Pavol Hell. List homomorphisms to reflexive graphs. *J. Comb. Theory, Ser. B*, 72(2):236 – 250, 1998.

[18] Tomás Feder, Pavol Hell, and Jing Huang. List homomorphisms and circular arc graphs. *Combinatorica*, 19(4):487 – 505, 1999.

[19] Tomas Feder, Pavol Hell, and Jing Huang. Bi-arc graphs and the complexity of list homomorphisms. *J. Graph Theory*, 42:61 – 80, January 2003.

[20] Tomás Feder, Pavol Hell, Jing Huang, and Arash Rafiey. Adjusted interval digraphs. *Electronic Notes in Discrete Mathematics*, 32(0):83 – 91, 2009.

[21] M. R. Garey, D. S. Johnson, Gary L. Miller, and C. H. Papadimitriou. The complexity of coloring circular arcs and chords. *SIAM J. Alg. Disc. Meth.*, 1(2):216 – 227, June 1980.

[22] P.C. Gilmore and A.J. Hoffman. *A Characterization of Comparability Graphs and of Interval Graphs*. Defense Technical Information Center, 1962.

[23] Martin Charles Golumbic and Renu Laskar. Irredundancy in circular arc graphs. *Discrete Applied Mathematics*, 44(1-3):79 – 89, 1993.

[24] Wolfgang Gutjahr, Emo Welzl, and Gerhard J. Woeginger. Polynomial graph-colorings. *Discrete Applied Mathematics*, 35(1):29 – 45, 1992.

[25] Michel Habib, Ross M. McConnell, Christophe Paul, and Laurent Viennot. Lex-bfs and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. *Theor. Comput. Sci.*, 234(1-2):59 – 84, 2000.

[26] H. Hadwiger and H. Debrunner. *Combinatorial Geometry in the Plane: Translated by Victor Klee*. Athena series. Selected topics in mathematics. Holt, Rinehart and Winston, 1964.

[27] P. Hell and J. Nesetřil. *Graphs and homomorphisms*. Oxford lecture series in mathematics and its applications. Oxford University Press, 2004.

[28] Pavol Hell and Jing Huang. Two remarks on circular arc graphs. *Graphs and Combinatorics*, 13(1):65 – 72, 1997.

[29] Pavol Hell, Monaldo Mastrolilli, Mayssam Nevisi, and Arash Rafiey. Approximation of minimum cost homomorphism. 2012.

[30] Pavol Hell and Jaroslav Nesetril. On the complexity of h-coloring. *J. Comb. Theory, Ser. B*, 48(1):92 – 110, 1990.

[31] Pavol Hell and Arash Rafiey. The dichotomy of list homomorphisms for digraphs. In *SODA*, pages 1703 – 1713, 2011.

[32] Ted Herman, Imran Pirwani, and Sriram Pemmaraju. Oriented edge colorings and link scheduling in sensor networks. In *SENSORWARE 2006: 1st International Workshop on Software for Sensor Networks*, 2006.

[33] J. Hopcroft and R. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4):225 – 231, 1973.

[34] W. Hsu and T. Ma. Fast and simple algorithms for recognizing chordal comparability graphs and interval graphs. *SIAM Journal on Computing*, 28(3):1004 – 1020, 1998.

[35] Wen-Lian Hsu. O(m*n) algorithms for the recognition and isomorphism problems on circular-arc graphs. *SIAM J. Comput.*, 24(3):411 – 439, 1995.

[36] Wen-Lian Hsu and Ross M. McConnell. Pc trees and circular-ones arrangements. *Theor. Comput. Sci.*, 296(1):99 – 116, March 2003.

[37] William T. Trotter Jr. and John I. Moore Jr. Characterization problems for graphs, partially ordered sets, lattices, and families of sets. *Discrete Mathematics*, 16(4):361 – 381, 1976.

[38] Haim Kaplan and Yahav Nussbaum. A simpler linear-time recognition of circular-arc graphs. *Algorithmica*, 61(3):694 – 737, November 2011.

[39] David G. Kendall. Incidence matrices, interval graphs and seriation in archeology. *Pacific J. Math.*, 28(3):565 – 570, 1969.

[40] Victor Klee. What are the intersection graphs of arcs in a circle? *The American Mathematical Monthly*, 76(7):810 – 813, 1969.

[41] William Klostermeyer and Gary MacGillivray. Homomorphisms and oriented colorings of equivalence classes of oriented graphs. *Discrete Mathematics*, 274(1-3):161 – 172, 2004.

[42] Paul Koebe. Kontaktprobleme der konformen abbildung. *Ber. Sächs. Akad.Wiss. Leipzig, Math.-Phys.*, 88:141 – 164, 1936.

[43] Norbert Korte and Rolf H. Möhring. An incremental linear-time algorithm for recognizing interval graphs. *SIAM J. Comput.*, 18(1):68 – 81, 1989.

[44] Dieter Kratsch, Ross M. McConnell, Kurt Mehlhorn, and Jeremy P. Spinrad. Certifying algorithms for recognizing interval graphs and permutation graphs. In *SIAM J. COMPUT*, pages 158 – 167, 2006.

[45] C. G. Lekkerkerker and J. C. Boland. Representation of a finite graph by a set of intervals on the real line. *Fund. Math.*, 51:45 – 64, 1962.

[46] Anna Lubiw. Doubly lexical orderings of matrices. *SIAM J. Comput.*, 16(5):854 – 879, 1987.

[47] Hermann A. Maurer, Ivan Hal Sudborough, and Emo Welzl. On the complexity of the general coloring problem. *Information and Control*, 51(2):128 – 145, 1981.

[48] R.M. McConnell, K. Mehlhorn, S. Nher, and P. Schweitzer. Certifying algorithms. *Computer Science Review*, 5(2):119 – 161, 2011.

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

[50] Ross M. McConnell and Jeremy P. Spinrad. Linear-time transitive orientation. In *Proceedings of the eighth annual ACM-SIAM symposium on Discrete algorithms*, SODA '97, pages 19 – 25, Philadelphia, PA, USA, 1997. Society for Industrial and Applied Mathematics.

[51] Jarik Nesetril, Andrea Raspaud, and Eric Sopena. Colorings and girth of oriented planar graphs. *Discrete Mathematics*, 165 - 166(0):519 – 530, 1997. Graphs and Combinatorics.

[52] Yahav Nussbaum. *Recognition of circular-arc graphs and some subclasses*. PhD thesis, 2007.

[53] Pascal Ochem, Alexandre Pinlou, and ric Sopena. On the oriented chromatic index of oriented graphs. *Journal of Graph Theory*, 57(4):313 – 332, 2008.

[54] Robert Paige and Robert Endre Tarjan. Three partition refinement algorithms. *SIAM J. Comput.*, 16(6):973 – 989, 1987.

[55] F.S. Roberts. Indifference graphs. Proof Tech. Graph Theory, Proc. 2nd Ann Arbor Graph Theory Conf. 1968, 139-146, 1969.

[56] Edward R. Scheinerman. Characterizing intersection classes of graphs. *Discrete Mathematics*, 55(2):185 – 193, 1985.

[57] A.M.S. Shrestha, S. Tayu, and S. Ueno. On two-directional orthogonal ray graphs. In *ISCAS*, pages 1807–1810, 2010.

[58] Eric Sopena. The chromatic number of oriented graphs. *Journal of Graph Theory*, 25(3):191 – 205, 1997.

[59] Eric Sopena. Oriented graph coloring. *Discrete Mathematics*, 229(13):359 – 369, 2001.

[60] Jeremy Spinrad. Circular-arc graphs with clique cover number two. *J. Comb. Theory, Ser. B*, 44(3):300 – 306, 1988.

[61] Jeremy Spinrad. Doubly lexical ordering of dense 0 - 1 matrices. *Inf. Process. Lett.*, 45(5):229 – 235, 1993.

[62] Jeremy Spinrad and Jacobo Valdes. Recognition and isomorphism of two dimensional partial orders. In *International Colloquium on Automata, Languages and Programming*, pages 676 – 686, 1983.

[63] S. Stefanakos and T. Erlebach. Routing in all-optical ring networks revisited. In *Proceedings of the Ninth International Symposium on Computers and Communications 2004 Volume 2*, pages 288 – 293, Washington, DC, USA, 2004. IEEE Computer Society.

[64] Alan C. Tucker. An efficient test for circular-arc graphs. *SIAM J. Comput.*, 9(1):1 – 24, 1980.

[65] Douglas B. West. *Introduction to Graph Theory (2nd Edition)*. Prentice Hall, August 2000.